

Exemplar: Create hash values

Activity overview

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.

Scenario

In this scenario, you 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.

Let's hash some files!

Note: The lab starts with your user account, called *analyst*, already logged in to the Bash shell. This means you can start the tasks as soon as you click the **Start**

Lab button. **Disclaimer:** For optimal performance and compatibility, it is recommended to use either **Google Chrome** or **Mozilla Firefox** browsers while accessing the labs.

Start your lab

You'll need to start the lab before you can access the materials. To do this, click the green "Start Lab" button at the top of the screen.



Start Lab

After you click the **Start Lab** button, you will see a shell, where you will be performing further steps in the lab. You should have a shell like this:

```
analyst@63fced8e3bc:~$
```

When you have completed all the tasks, refer to the End your Lab section that follows the tasks for information on how to end your lab.

Task 1. 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 same 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.

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

The command to complete this step:

```
ls
```

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

2. Use the cat command to display the contents of the file1.txt file:

```
cat file1.txt
```

Note: If you enter a command incorrectly and it fails to return to the command-line prompt, you can press **CTRL+C** to stop the process and force the shell to return to the command-line prompt.

3. Use the cat command to display the contents of the file2.txt file:

```
cat file2.txt
```

4. Review the output of the two file contents:

5. analyst@4fb6d613b6b0:~\$ cat file1.txt

6. X5O!P%@AP[4\PZX54(P^)7CC)7}\$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!\$H+H*

7. analyst@4fb6d613b6b0:-\$ cat file2.txt

X5O!P%@AP[4\PZX54(P^)7CC)7}\$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!\$H+H*

Do the contents of the two files appear identical when you use the cat command?

Yes

No

Submit

Answer: Yes. The contents of the two files appear identical when you use the cat command to display the file contents.

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.

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

sha256sum file1.txt

© Anushka Sinha | github.com/rikusmiles/
You now need to follow the same step for the file2.txt file.

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

sha256sum file2.txt

7. Review the generated hashes of the contents of the two files:

8. analyst@4fb6d613b6b0:-\$ sha256sum file1.txt

9. 131f95c51cc819465fa1797f6ccacf9d494aaaff46fa3eac73ae63ffbfd8267 file1.txt

10. analyst@4fb6d613b6b0:-\$ sha256sum file2.txt

2558ba9a4cad1e69804ce03aa2a029526179a91a5e38cb723320e83af9ca017b file2.txt

Do both files produce the same generated hash value?

No

Yes

Submit

© Anushka Sinha | <https://github.com/rikusmiles/> | Original work

Answer: No. The generated hash value for file1.txt is different from the generated hash value for file2.txt, which indicates that the file contents are not identical.

Click **Check my progress** to verify that you have completed this task correctly.

Generate hashes for files

Check my progress

Task 2. Compare hashes

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

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

```
sha256sum file1.txt >> file1hash
```

You now need to complete the same step for the file2.txt file.

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

```
sha256sum file2.txt >> file2hash
```

Now, you should have two hashes written to separate files. The first hash was written to the file1hash file, and the second hash was written to the file2hash file.

You can manually display and compare the differences.

3. Use the cat command to display the hash values in the file1hash and file2hash files.

The command to complete this step:

```
cat file1hash
```

```
cat file2hash
```

4. Inspect the output and note the difference in the hash values.

Note: Although the content in `file1.txt` and `file2.txt` previously appeared identical, the hashes written to the `file1hash` and `file2hash` files are **completely** different.

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.

5. Use the `cmp` command to highlight the differences in the `file1hash` and `file2hash` files:

```
cmp file1hash file2hash
```

6. Review the output, which reports the first difference between the two files:

```
analyst@4fb6d613b6b0:~$ cmp file1hash file2hash
```

```
file1hash file2hash differ: char1, line 1
```

Note: The output of the `cmp` command indicates that the hashes differ at the first character in the first line.

Based on the hash values, is `file1.txt` different from `file2.txt`?

Yes

No

Submit

Answer: Yes, the contents of the two files are different because the hash values of each file are different.

Click **Check my progress** to verify that you have completed this task correctly.

Compare hashes

Check my progress

Conclusion

Great work!

You practiced how to

- compute hashes using `sha256sum`,
- display hashes using the `cat` command, and
- compare hashes using the `cmp` command.

These are valuable tools you can use to validate data integrity as you contribute to the control of your organization's security.

End your lab

Before you end the lab, make sure you're satisfied that you've completed all the tasks, and follow these steps:

1. Click **End Lab**. A pop-up box will appear. Click **Submit** to confirm that you're done. Ending the lab will remove your access to the Bash shell. You won't be able to access the work you've completed in it again.
2. Another pop-up box will ask you to rate the lab and provide feedback comments. You can complete this if you choose to.
3. Close the browser tab containing the lab to return to your course.
4. Refresh the browser tab for the course to mark the lab as complete.

© Anushka Sinha | github.com/rikusmiles/