Google IT Support Certificate: Course 5: IT Security - Defense Against the Digital Dark Arts

Week 2: Pelcgbybtl (Cryptography)

Quiz: Hands-on with Hashing

Introduction

In this lab, you'll have hands-on practice demonstrating hashing and hash verification using md5sum and shasum tools.
Md5sum is a hashing program that calculates and verifies 128-bit MD5 hashes. As with all hashing algorithms, theoretically, there's an unlimited number of files that will have any given MD5 hash. Md5sum is used to verify the integrity of files. Similarly, shasum is an encryption program that calculates and verifies SHA hashes. It's also commonly used to verify the integrity of files.
In this lab, you'll see that almost any change to a file will cause its MD5 hash or SHA hashes to change.

What you'll do

- **Compute** - You'll create a text file and generate hashes using the md5sum and shasum tools.
- **Inspect** - After you generate the hash digests, you'll inspect the resulting files.
- **Verify** - You'll verify the hash using the md5sum and shasum tools.
- **Modify** - You'll modify the text file and compare these results to the original hash to observe how the digest changes and how the hash verification process fails.

MD5

Let's kick things off by creating a text file containing some data. Feel free to substitute your own text data, if you want. This command creates a text file called "file.txt" with a single line of basic text in it:

You should see the following output (or something very similar) :

```
echo 'This is some text in a file, just so we have some data' > file.txt
```

Copied!

content_copy

Click *Check my progress* to verify the objective.

Create Text File

Check my progress

You'll now generate the MD5 sum for the file and store it. To generate the sum for your new file, enter this md5sum command:

```
md5sum file.txt > file.txt.md5
```

Copied!

content_copy

This creates the MD5 hash, and saves it to a new file. You can take a look at the hash by printing its contents to the screen, using this command:

```
cat file.txt.md5
```

Copied!

content_copy

This should print the hash to the terminal, which should look something like this:

```
c7a8ef893898f9a6b380eb4ec1e87113  file.txt
```

More importantly, you can also verify that the hash is correct, and that the original file hasn't been tampered with since the sum was made. To do this, enter this command and see the following output, which indicates that the hash is valid:

```
md5sum -c file.txt.md5
```

Copied!

content_copy

```
file.txt: OK
```

Click *Check my progress* to verify the objective.

md5sum

Check my progress

Verifying an invalid file

Next, we'll demonstrate the security of this process by showing how even a single-character change to the file results in a different hash. First, you'll create a copy of the text file, and insert a single space at the end of the file. Feel free to use any text-editor that you'd like. Head's up that we've included instructions for making this change in Nano. To make a copy of the file, enter this command:

```
cp file.txt badfile.txt
```

Copied!

content_copy

Then generate a new md5sum for the new file:

```
md5sum badfile.txt > badfile.txt.md5
```

Copied!

content_copy

Note that the resulting hash is **identical** to the hash for our original file.txt despite the filenames being different. This is because hashing only looks at the data, not the metadata of the file.

```
cat badfile.txt.md5
```

Copied!

content_copy

```
cat file.txt.md5
```

Copied!

content_copy

To open the text file in Nano, enter this command:

```
nano badfile.txt
```

Copied!

content_copy

This will open the file in the text editor. To add a space to the end of the file, use the arrow keys (not the mouse!) to move the cursor to the end of the line of text. Then, press the spacebar to add a space character to the end of the file. Your screen should look like this image:

```
This is some text in a file, just so we have some data
<br>
<br>
<br>
<br>
<br>
<br>
```

```
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit  ^R Read File  ^\ Replace  ^U Uncut Text  ^T To Spell  ^_ Go To Line
```

To save the file, press **ctrl+X**. You should see this message:

```
This is some text in a file, just so we have some data

<br>

<br>

<br>

<br>

<br>

<br>

Save modified buffer?  (Answering "No" will DISCARD changes.)

 Y Yes
 N No           ^C Cancel
```

Confirm by typing **Y** for **yes**, then press **Enter** to confirm.

This will take you back to the normal terminal screen. Now that you've made a very minor change to the file, try verifying the hash again. It should fail verification this time, showing that any change at all will result in a different hash. Try to verify it by entering this command again:

```
md5sum -c badfile.txt.md5
```

Copied!

content_copy

You should see a message that shows that the verification wasn't successful:

```
badfile.txt: FAILED
md5sum: WARNING: 1 computed checksum did NOT match
```

Click *Check my progress* to verify the objective.

md5sum failure

Check my progress

To see how different the hash of the edited file is, generate a new hash and inspect it:

```
md5sum badfile.txt > new.badfile.txt.md5
```

Copied!

content_copy

```
cat new.badfile.txt.md5
```

Copied!

content_copy

Check out how it's different from our previously generated hash:

```
dcd879fd2c162dbfe9a186a67902e7ce  badfile.txt
```

For reference, here are the contents of the original sum:

```
c7a8ef893898f9a6b380eb4ec1e87113  file.txt
```

Click *Check my progress* to verify the objective.

Recompute MD5 Sum

Check my progress

SHA

Let's do the same steps, but for SHA1 and SHA256 hashes using the shasum tool. Functionally, the two work in very similar ways, and their purpose is the same. But SHA1 and SHA256 offer stronger security than MD5, and SHA256 is more secure than SHA1. This means that it's easier for a malicious third party to attack a system using MD5 than one using SHA1. And because SHA256 is the strongest of the three, it's currently widely used.

SHA1

To create the SHA1 sum and save it to a file, use this command:

```
shasum file.txt > file.txt.sha1
```

Copied!

content_copy

View it by printing it to the screen, like you've done before:

```
cat file.txt.sha1
```

Copied!

content_copy

```
65639a89992784291d769e05338085d1739645c6  file.txt
```

Now, verify the hash using the command below. (Like before, this would fail if the original file had been changed.)

```
shasum -c file.txt.sha1
```

Copied!

content_copy

You should see the following output, indicating that the verification was a success:

```
file.txt: OK
```

Click *Check my progress* to verify the objective.

SHA1 Hash

Check my progress

SHA256

The same tool can be used to create a SHA256 sum. The "-a" flag specifies the algorithm to use, and defaults to SHA1 if nothing is specified. To generate the SHA256 sum of the file, use this command:

```
shasum -a 256 file.txt > file.txt.sha256
```

Copied!

content_copy

You can output the contents of this file, the same as before:

```
cat file.txt.sha256
```

Copied!

content_copy

SHA256's increased security comes from it creating a longer hash that's harder to guess. You can see that the contents of the file here are much longer than the SHA1 file:

```
7a54af37c15a82e157c8368324e7234d22778ce845219cd16172895a608030ff  file.txt
```

Finally, to verify the SHA256 sum, you can use the same command as before:

```
shasum -c file.txt.sha256
```

Copied!

content_copy

Click *Check my progress* to verify the objective.

SHA256 Hash

Check my progress

Conclusion

Congrats! You've successfully created and verified hashes using three of the most common hashing algorithms: MD5, SHA1, and SHA256. You've also demonstrated the security of these hashes by showing that even very small changes to the file result in a new hash and cause the verification to fail.