Participation 01 Assignment

Validating Data Acquisitions

This lab is to be completed on our Linux CSE server (i.e., cse01).

Validating data is the most critical aspect of computer forensics that requires using a hashing algorithm. There are several data validation techniques including CRC-32, MD5, and SHA-1.

There are three rules for forensic hashes:

1. You cannot predict the hash value of a file or device.
2. No two hash values can be the same.
3. If anything changes in the file or device, the hash value must change.

Note: You must perform this exercise *exactly* as described or your answers may be different than what is expected.

**CRC-32 Data Validation**

The Cyclic Redundancy Check 32 (CRC-32) uses a mathematical algorithm that determines whether a file’s contents has changed, but is not considered a forensic hashing algorithm as it is really geared towards reliably detecting 1–2 bit error patterns that commonly occur during transmission.

To show why CRC-32 is not used as a forensic hash, log onto our Linux CSE server cse01 and create a file called **textfile1** that only contains the text "plumless", without the quotes, with no other characters (not even a carriage return) included in the file. You should do this with the command: echo plumless > textfile1

1. Now, run the crc32 Linux utility on this file (e.g., crc32 textfile1) and record the resulting hash value: \_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. Copy this file using the cp command as **textfile2**. Then, change the letter p to a b in the file using the vim or nano editor and run the crc32 utility against this new file to record the resulting hash value: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

You should have noticed that the hash value changed from the original file, which makes sense, since the file contents have changed.

Create a new file called **textfile3** that only contains the text "buckeroo", without the quotes, as before with no other characters (not even a carriage return) included in the file. You can do this with the command: echo buckeroo > textfile3

3. Run the crc32 utility on this file and record the resulting hash value: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Hopefully, you should now notice that this file (with a completely different word than the first file) has the same hash value as the first file, which violates the second and third rules for forensic hashes.

**MD5 and SHA-1 Data Validation**

Message Digest 5 (MD5) and Secure Hash Algorithm 1 (SHA-1) use mathematical formulas that translate a file into a hexadecimal code value, or a hash value such that if a bit or byte in the file is changed, the digital hash is altered. Although both MD5 and SHA-1 have collisions, they are still widely used, as the probability of a collision remains realistically small.

In this portion of this participation activity, we will learn how to use dd and MD5/SHA-1 to create, restore, and forensically verify sound files (normally, you will do this with disk images, but since this privilege is not allowed by student accounts on our CSE machine, we do this with files that we create).

1. First, create an MD5 and SHA-1 checksum of your **textfile3**. To do this, run the md5sum and sha1sum Linux utilities on the file and record your results:

MD5: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

SHA-1: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Create an image file of the textfile3 file by issuing the following command:

**dd if=textfile3 of=<EUID>A.img bs=1k**

where <EUID> should be replaced with your actual EUID (e.g., dd if=textfile3 of=mat0299A.img bs=1k).

You should now have an image file created in your current directory based on the **textfile3** file that you created earlier. Now, verify that this image file is *exactly* the same as the original file using the md5sum and sha1sum utilities. You don’t have to write the hash values again (since they should be the same as you found earlier), but if not, repeat the process and correct any mistakes you might have made until you get these values to be the same.

1. Next, restore the image file to a file using dd to copy the image file to the file textfile4 as follows:

**dd if=<EUID>A.img of=textfile4 bs=1k**

where <EUID> should be replaced with your actual EUID as before.

This command reverses the flow of the data. Again, verify the accuracy of the dd command by comparing the MD5 and SHA-1 hash values of the textfile4 file with the original textfile3 file. As before, you don’t have to write the hash values again, but make sure that the values are the same, and if not, repeat this process until they are the same.

1. We now want to test MD5 and SHA-1 against an altered image file. We can do this by simply adding a single byte of data to the image file <EUID>A.img as follows:

**echo x >> <EUID>A.img**

where <EUID> should be replaced with your actual EUID as before.

1. Finally, run the md5sum and sha1sum utilities on this altered image file and record your results:

MD5: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

SHA-1: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Notice how a difference of only 1 byte causes both the MD5 and SHA1 hash values to change drastically. This demonstrates the value of using and checking MD5 and SHA1 signatures of files when downloading them from the Internet.

You are to submit this document, with your solutions, to the **Participation 01** dropbox on Canvas by the due date and time. **No late submissions will be accepted.**