Hashing and Integrity

Creating hashes is an important part of digital forensics, as it allows any tampering or modification of evidence to be immediately visible. This lesson will cover what hashes are, how they can be retrieved, how they're used to ensure integrity of digital evidence, how they can be cracked (even though they're a one-way function!), and finish up with some practical exercises in the next lesson.

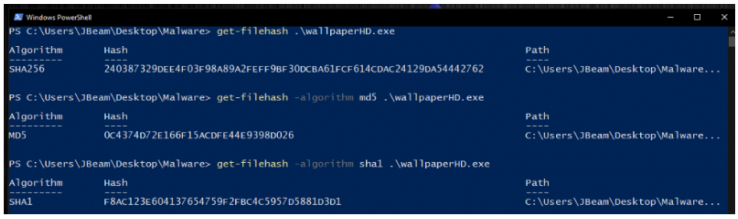
WHAT ARE HASHES?

Hash values, which come in the form of text strings, are the unique fingerprint of a file or string. If I had a text file with the letter "ABC" in it, I could generate a hash value. Now if I went back into that file and added the letter "D" to it, and retrieved the file's hash value, it will be different than the initial one. We have modified the contents, so now the fingerprint is different.

The most common hash to work with is Message Digest 5, commonly referred to as MD5. Two other common hashes include SHA1, and SHA256. Due to collisions, an event where two different data values can have the same hash value, MD5 is no longer used as a secure standard, and SHA256 is taking over as the most common algorithm to use. We have already covered how to generate MD5, SHA1, and SHA256 hashes in the Phishing Analysis domain, but we'll provide a quick overview here.

Gathering Hashes in Windows

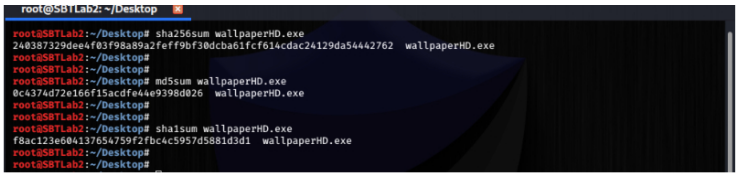
In the below screenshot we are using PowerShell on a Windows system to generate different file hashes for an executable file named "wallpaperHD.exe". By default, the command **`get-filehash <file>`** will generate a SHA256 hash. If we want to retrieve the MD5 or SHA1 values, we need to add the '-algorithm' flag to specify what hashes we want. Using **`get-filehash -algorithm md5 <file>`** we are able to retrieve the md5 hash, and the same method can be applied for SHA1.



Gathering Hashes in Linux

On a Linux system generating hashes is a lot quicker. We can use the following three commands to generate SHA256, MD5, and SHA1 hashes respectively:

- **`sha256sum <file>`**
- **`md5sum <file>`**
- **`sha1sum <file>`**



We can also retrieve the hash values of text strings using the command **`echo -n <text> | string`**, as demonstrated below.

EVIDENCE INTEGRITY

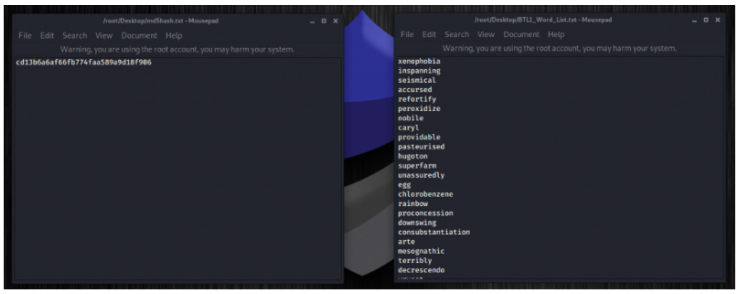
Now that you understand how hashes work, and how they're generated, you should be able to see how this applies to digital forensics, and ensuring the integrity of files or evidence. In most investigations involving a hard drive, a hash will be generated from the hard drive, and then a complete copy of the storage media will be taken at a bit-by-bit level, meaning that **everything** possible from the disk is copied to a fresh hard drive. This new hard drive then has its hash generated, to ensure that this is the exact same value as the original, proving that an exact copy was successfully generated. This allows forensic analysts or investigators to work on a copy of the evidence, instead of analyzing the actual disk which could result in loss of evidence if anything went wrong, or the court could argue that the evidence may have been tampered with, and is therefore not viable for use in court during legal proceedings.

REVERSING HASHES

Hashing functions are one-way, meaning that it is not possible to reverse the initial value directly. The only way is by employing brute force or dictionary attacks to crack the hash. You need to calculate the hash of every potential combination, or against a huge list of strings, and compare it to the hash you want to crack. You can also use free online services that have already calculated millions of hashes.

We're going to use a command-line tool called Hashcat to perform a dictionary attack against a target MD5 hash. In this attack, Hashcat will generate the hash value of the text strings in the word list, and see if the hash matches the one we're trying to crack. If they match, we've found the text string that has been hashed, and in a way, reversed the hashing process. This type of activity is typically done when penetration testers or red team members gain access to account credentials that are encrypted, and try to find the plain text versions of them.

So for this example, we have the target MD5 hash in a text file named "md5hash.txt" and our random wordlist "BTL1_Word_List.txt".

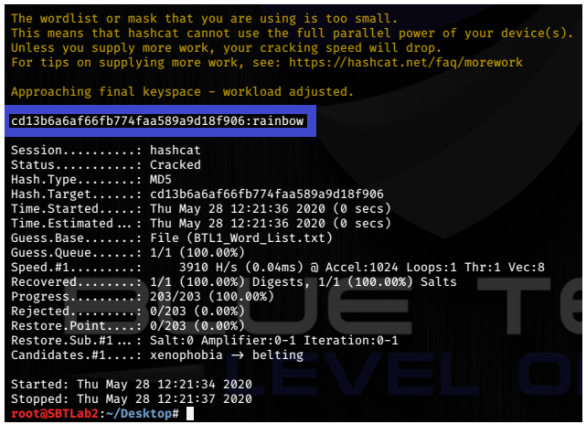


To set Hashcat to work, we need to open a terminal in the same location as our two files, in this case, the Desktop. We'll right-click the desktop and select "Open Command Prompt" and use the following command:

Using RDP and SSH
How to Start Your Exam

```
hashcat -m 0 md5hash.txt BTL1_Word_List.txt
```

- **"hashcat"** selects the tool we want to use
- **"-m 0"** selects the hash mode, 0 is md5
- **"md5hash.txt"** the file name containing the hashes to crack
- **"BTL1_Word_List.txt"** the file name of our wordlist



Once Hashcat has identified a matching hash, it will present the plain text value of the hash we wanted to crack! It's very important to remember that a dictionary attack will only ever succeed if the right text string is in the word list. In this case, the answer was Rainbow – if this word wasn't in the list, Hashcat wouldn't have identified that the hash we're cracking had the same value, and we wouldn't have got a result.

Quizzes

Lab) Hashing and Integrity

Previous Topic

Back to Lesson