Hashing

A hash algorithm is a one-way algorithm that creates a fixed length output. The input to a hashing algorithm can be arbitrarily sized. Hashes have various applications and are used throughout many technologies. While hash functions are used in cryptography applications, they can also simplify data lookup with data types called hash tables.

Often in cyber security applications, they are used to sign or verify data. Hashes can verify a successful download from the internet, ensure a file in a court case hasn't been changed, confirm data packets in internet traffic transmission, or validate passwords on an internet server without storing the password in memory.

Not all hash algorithms are created equal. For example, the SHA256 algorithm is a standard used in many cryptographic applications. As of now, no known attacks have compromised the algorithm. However, a popular algorithm, MD5, has demonstrated collisions and security vulnerabilities. A collision occurs when two different inputs create the same output. For a good hashing algorithm, collisions should be statistical anomalies.

Hashing Exercise

For this lab we will be using Windows PowerShell to perform common hash functions on some test files. PowerShell is an improved windows command prompt, with backward functionality and added commands. Windows 7 and 10 should natively have PowerShell. We will only be using basic commands, but if you are unfamiliar with windows commands to navigate directories here's an exhaustive command list:

https://docs.microsoft.com/en-us/powershell/scripting/getting-started/cookbooks/managing-current-location?view=powershell-6

Steps:

1) Navigate to a directory you can make a few test files in. To create a new text file you can type the command:

   notepad <filename.txt>

2) Create two files with different names, but the same content. (I typed the sentence "This is a test." in both). Save in notepad and your files will now be listed in your current directory.

3) Run hashing algorithms on both files. Try MD5 and SHA256. Other options supported by this command are: SHA1, SHA256, SHA384, SHA512, MACTripleDES, MD5, and RIPMD160. Use this command:

   get-filehash <filename.txt> -Algorithm <hash_type>

4) Are the hashes the same for each file? What are the sizes of output for each type of algorithm?

5) Now change the content of one of the files by a single character. (I added a second period ". ") Does the hash result change? By how much?

6) Hashing isn't limited to text files. Find another file on your filesystem and attempt to hash it.

7) Now, lets look at a collision. MD5, once considered secure, has been proven to have many collisions. The strings below were found on the MD5 Wikipedia page. Can you notice the difference in the strings?

8) Copy and paste each string into a text file and save. Run the MD5 algorithm on them and observe the output. What is the output? Why may this pose problems for cryptography?

NOTE: If you want to remove our test files type this command in the current directory:

Remove-item <filename.txt>

MD5 Collision Strings:

d131dd02c5e6eec4 693d9a0698aff95c 2fcab58712467eab 4004583eb8fb7f89

55ad340609f4b302 83e488832571415a 085125e8f7cdc99f d91dbdf280373c5b

d8823e3156348f5b ae6dacd436c919c6 dd53e2b487da03fd 02396306d248cda0

e99f33420f577ee8 ce54b67080a80d1e c69821bcb6a88393 96f9652b6ff72a70

d131dd02c5e6eec4 693d9a0698aff95c 2fcab50712467eab 4004583eb8fb7f89

55ad340609f4b302 83e4888325f1415a 085125e8f7cdc99f d91dbd7280373c5b

d8823e3156348f5b ae6dacd436c919c6 dd53e23487da03fd 02396306d248cda0

e99f33420f577ee8 ce54b67080280d1e c69821bcb6a88393 96f965ab6ff72a70