Lab 3: Hashing

Objective: The key objective of this lab is to understand the range of hashing methods used, analyse the strength of each of the methods, and in the usage of salting. Overall the most popular hashing methods are: MD5 (128-bit); SHA-1 (160-bit); SHA-256 (256-bit); SHA-3 (256-bit), bcrypt (192-bit) and PBKDF2 (256-bit). The methods of bcrypt, scrypt and PBKDF2 use a number of rounds, and which significantly reduce the hashing rate. This makes the hashing processes much slower, and thus makes the cracking of hashed passwords more difficult. We will also investigate the key hash cracking tools such as **hashcat**.

🕮 **Web link:** https://github.com/billbuchanan/esecurity/tree/master/unit03_hashing

Open up your **Ubuntu instance** within vsoc.napier.ac.uk and conduct this lab.

Demo: https://youtu.be/rnTLr6iUbf0

If required, you can check the hashing methods here: https://asecuritysite.com/encryption/js10

A Hashing

In this section we will look at some fundamental hashing methods.

No	Description	Result

A.1	Using (either on your Windows desktop or on Ubuntu):
	🕮 **Web link (Hashing):** `http://asecuritysite.com/encryption/md5`
	Match the hash signatures with their words ("Falkirk", "Edinburgh", "Glasgow" and "Stirling").
	`03CF54D8CE19777B12732B8C50B3B66F`
	`D586293D554981ED611AB7B01316D2D5`
	`48E935332AADEC763F2C82CDB4601A25`
	`EE19033300A54DF2FA41DB9881B4B723`
		`03CF5:` Is it [Falkirk][Edinburgh][Glasgow][Stirling]?
		`D5862:` Is it [Falkirk][Edinburgh][Glasgow][Stirling]?
		`48E93:` Is it [Falkirk][Edinburgh][Glasgow][Stirling]?
		`EE190:` Is it [Falkirk][Edinburgh][Glasgow][Stirling]?

A.2	Repeat Part 1, but now use openssl, such as: echo -n 'Falkirk' | openssl md5	`03CF5:` Is it [Falkirk][Edinburgh][Glasgow][Stirling]? `D5862:` Is it [Falkirk][Edinburgh][Glasgow][Stirling]? `48E93:` Is it [Falkirk][Edinburgh][Glasgow][Stirling]? `EE190:` Is it [Falkirk][Edinburgh][Glasgow][Stirling]?
A.3	Using:	MD5 hex chars:

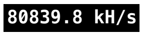
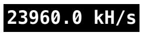
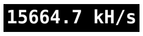
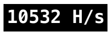
	🕮 **Web link (Hashing):**
	`http://asecuritysite.com/encryption/md5`
	
	Determine the number of hex characters for the hash signatures defined. Note: perhaps copy and paste your hash to an on-line character counter?
		32
		SHA-1 hex chars: 40
		
		SHA-256 hex chars: 64
		
		SHA-384 hex chars: 96
		
		SHA-512 hex chars: 128
		
		How does the number of hex characters relate to the length of the hash signature: The number of hexadecimal characters increases linearly with the bit-length of the hash – each hex digit encodes 4 bits of data.

A.4	For the following /etc/shadow file, determine the matching password:
	
	`bill:$apr1$waZS/8Tm$jDZmiZBct/c2hysERcZ3m1`
	`mike:$apr1$mKfrJquI$Kx0CL9krmqhCu0SHKqp5Q0`
	`fred:$apr1$Jbe/hCIb$/k3A4kjpJyC06BUUaPRKs0`
	`ian:$apr1$0GyPhsLi$jTTzW0HNS4Cl5ZEoyFLjB.`
	`jane: $1$rqOIRBBN$R2pOQH9egTTVN1Nlst2U7.`
	
	[Hint: openssl passwd -apr1 -salt *ZaZS/8TF napier*]
		The passwords are **password**, **napier**, **inkwell** and **Ankle123**.
		
		Bill's password: napier
		
		Mike's password: Ankle123
		
		Fred's password: inkwell
		
		Ian's password: password
		
		Jane's password: napier

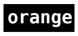
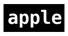
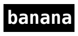
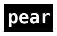
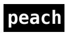
A.5	From Ubuntu, download the following:
	
	🕮 **Web link (Files):**
	`http://asecuritysite.com/files02.zip`
	
	(a quick way to download is `wget asecuritysite.com/files02.zip`) and the files should have the following MD5 signatures:
	
	`MD5(1.txt)= 5d41402abc4b2a76b9719d911017c592`
	`MD5(2.txt)= 69faab6268350295550de7d587bc323d`
	`MD5(3.txt)= fea0f1f6fede90bd0a925b4194deac11`
	`MD5(4.txt)= d89b56f81cd7b82856231e662429bcf2`
		Which file(s) have been modified? 2.txt
		
		`MD5 (2. txt) = e3fc91b12a36c2334ebb5b66caa2d75b`

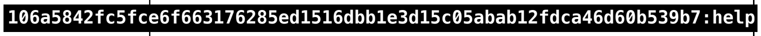
A.6	From Ubuntu, download the following ZIP file:
	
	🕮 **Web link (PS Files):**
	`http://asecuritysite.com/letters.zip`
	(a quick way to download is `wget asecuritysite.com/letters.zip`)
	On your Ubuntu instance, you should be able to view the files by double clicking on them in the file explorer (as you should have a PostScript viewer installed).
	
	`cat letter_of_rec.ps | openssl md5`
		Do the files have different contents? Yes
		
		Now determine the MD5 signature for them. What can you observe from the result? The two different files have the same md5 signature

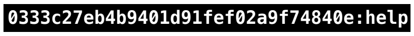
B Hash Cracking (Hashcat)

No	Description	Result
B.1	Run the hashcat benchmark (eg hashcat –b -m 0), and complete the following:	Hash rate for MD5:  80839.8 kH/s Hash rate for SHA-1:  23960.0 kH/s Hash rate for SHA-256:  15664.7 kH/s Hash rate for APR1:  10532 H/s

B.2	On Ubuntu, next create a word file (**words**) with the words of "napier", "password" "Ankle123" and "inkwell"
	Using hashcat crack the following MD5 signatures (hash1):
	`232DD5D7274E0D662F36C575A3BD634C`
	`5F4DCC3B5AA765D61D8327DEB882CF99`
	`6D5875265D1979BDAD1C8A8F383C5FF5`
	`04013F78ACCFEC9B673005FC6F20698D`
	Command used: `hashcat -m 0 hash1 words`
		`232DD...634C`
		Is it [napier][password][Ankle123][inkwell]?
		`5F4DC...CF99` Is it [napier][password][Ankle123][inkwell]?
		`6D587...5FF5` Is it [napier][password][Ankle123][inkwell]?
		`04013...698D` Is it [napier][password][Ankle123][inkwell]?

B.3	Using the method used in the first part of this tutorial, find the following for names of fruits (the fruits are all in lowercase):
	`FE01D67A002DFA0F3AC084298142ECCD`
	`1F3870BE274F6C49B3E31A0C6728957F`
	`72B302BF297A228A75730123EFEF7C41`
	`8893DC16B1B2534BAB7B03727145A2BB`
	`889560D93572D538078CE1578567B91A`
		`FE01D:`  orange
		`1F387:`  apple
		`72B30:`  banana
		`8893D:`  pear
		`88956:`  peach

B.4	Put this SHA-256 value in a file named file.txt:
	`106a5842fc5fce6f663176285ed1516dbb`
	`1e3d15c05abab12fdca46d60b539b7`
	By adding a word of "help" in a word file of words.txt, prove that the following cracks the hash (where file.txt contains the hashed value):
	`hashcat -m 1400 file.txt words.txt`
		 106a5842fc5fce6f663176285ed1516dbb1e3d15c05abab12fdca46d60b539b7:help

B.5	The following is an NTLM hash, for "help":
	`0333c27eb4b9401d91fef02a9f74840e`
	Prove that the following can crack the hash (where file.txt contains the hashed value):
	`hashcat -m 1000 file.txt words.txt`
		 0333c27eb4b9401d91fef02a9f74840e:help

The cracked hashed are stored in:

```
~/.hashcat/hashcat.potfile
```

What do you observe when you use the command:

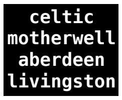
```
cat ~/.hashcat/hashcat.potfile
```

Note, hashcat doesn't show previously cracked values, so if you want it to recrack them, just use:

```
rm ~/.hashcat/hashcat.potfile
```

B.6 Now crack the following Scottish football teams (all are single words):

```
635450503029fc2484f1d7eb80da8e25bdc1770e1dd14710c592c8929ba37ee9
BEF68628460A29657F55A2860407969E3AF183E889021B30091C815F6C6B248D
bc5fb9abe8d5e72eb49cf00b3dbd173cbf914835281fadd674d5a2b680e47d50
6ac16a68ac94ca8298c9c2329593a4a4130b6fed2472a98424b7b4019ef1d968
```

Football teams: 

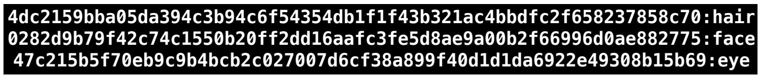
```
celtic
motherwell
aberdeen
livingston
```

B.7 Rather than use a dictionary, we can use a brute force a hashed password using a lowercase character set:

```
hashcat -a 3 -m 1400 file.txt ?l?l?l?l?l?l?l?l --increment
```

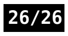
Using this style of command (look at the hash type and perhaps this is a SHA-256 hash), crack the following words:

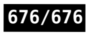
```
4dc2159bba05da394c3b94c6f54354db1f1f43b321ac4bbdfc2f658237858c70
0282d9b79f42c74c1550b20ff2dd16aafc3fe5d8ae9a00b2f66996d0ae882775
47c215b5f70eb9c9b4bcb2c027007d6cf38a899f40d1d1da6922e49308b15b69
```

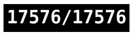
Words: 

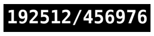
```
4dc2159bba05da394c3b94c6f54354db1f1f43b321ac4bbdfc2f658237858c70:hair
0282d9b79f42c74c1550b20ff2dd16aafc3fe5d8ae9a00b2f66996d0ae882775:face
47c215b5f70eb9c9b4bcb2c027007d6cf38a899f40d1d1da6922e49308b15b69:eye
```

Number of tests for each sequence tried:

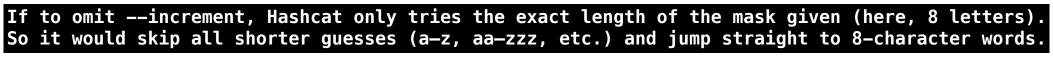
a->z:  26/26

aa->zz:  676/676

aaa->zzz:  17576/17576

aaaa->zzzz:  192512/456976

What happens when you take the "--increment" flag away?



```
If to omit --increment, Hashcat only tries the exact length of the mask given (here, 8 letters).
So it would skip all shorter guesses (a-z, aa-zzz, etc.) and jump straight to 8-character words.
```

B.8 We can focus on given letters, such as where we add a letter or a digit at the end:

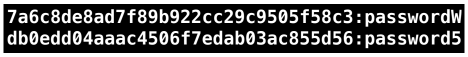
```
hashcat -a 3 -m 1000 file.txt password?l
hashcat -a 3 -m 1000 file.txt password?u
hashcat -a 3 -m 1000 file.txt password?d
```

Using these commands, crack the following:

```
7a6c8de8ad7f89b922cc29c9505f58c3
db0edd04aaac4506f7edab03ac855d56
```

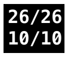
Note: Remember to try both MD5 (0) and NTLM hash (1000).

Words:



```
7a6c8de8ad7f89b922cc29c9505f58c3:passwordW
db0edd04aaac4506f7edab03ac855d56:password5
```

Number of tests for each:



```
26/26
10/10
```