

# Secure Communications

## Week 8

## Hashing

### Sections

#### A. Hashing

##### A.1 <http://asecuritysite.com/encryption/md5>

"Falkirk"	48E935332AADEC763F2C 82CDB4601A25	<div><div>Hash Example (Hex)</div><div><div>Encryption Home(Home)</div><div>Full Link</div><div>Message:</div><div>Hex input: ( )</div><div>Generate Hash</div><div>The results are then:</div><div>MD5: 48E935332AADEC763F2C82CDB4601A25</div><div>SHA-1: A608B0317364808B1308B011CF02E0C8C6DF</div><div>SHA-256: F0D78F0C1049EE1EC069A0001EE5FC7863CA4EA029407D0FEE8BCA9F1</div><div>SHA-384: 8F04A2AF10D63FA2A00D10F07E03AC4C0CB863F320AC0CF1FDD18EAC033030F7F</div><div>SHA-512: 3C1C08BAC4010B3A07D032874F806900CE48806F17FF23A40B7AF00E10F0B4179FD</div></div></div>
"Edinburgh"	03CF54D8CE19777B1273 2B8C50B3B66F	<div><div>Hash Example (Hex)</div><div><div>Encryption Home(Home)</div><div>Full Link</div><div>Message:</div><div>Hex input: ( )</div><div>Generate Hash</div><div>The results are then:</div><div>MD5: 03CF54D8CE19777B12732B8C50B3B66F</div><div>SHA-1: EA374DA1963E28BC1414807065001A803A03E</div><div>SHA-256: 4F0C4580DA1A540B201E3E198278B910E4A518A50858850805090C081</div><div>SHA-384: 5D4A76381D405197E031889E1EC19CA7178C03F5A6E78F8033A48896C86089E3CC00BE</div><div>SHA-512: 416A51D08F205F1D4550E0C005036A0C0080206A4C200006A7C0B90E018F0789F1D705</div></div></div>
"Glasgow"	D586293D554981ED611A B7B01316D2D5	<div><div>Hash Example (Hex)</div><div><div>Encryption Home(Home)</div><div>Full Link</div><div>Message:</div><div>Hex input: ( )</div><div>Generate Hash</div><div>The results are then:</div><div>MD5: D586293D554981ED611A B7B01316D2D5</div><div>SHA-1: C0F780E0811F070E0A8178C07E19E48F12</div><div>SHA-256: 9E0F708F7F8B9F7E70C439010CC091E030E140C90890C40CE81</div><div>SHA-384: F0A05F8A0714D10B70C00186A5070018B0F578CA13AC2EEA1C4F7A809F9F8EC130FE</div><div>SHA-512: 97C0E4214C278E1F70B1E23A34A3AC3C063A000913821D54F8A7A07A083D042180</div></div></div>
"Stirling"	EE19033300A54DF2FA41 DB9881B4B723	<div><div>Hash Example (Hex)</div><div><div>Encryption Home(Home)</div><div>Full Link</div><div>Message:</div><div>Hex input: ( )</div><div>Generate Hash</div><div>The results are then:</div><div>MD5: EE19033300A54DF2FA1DB9881B4B723</div><div>SHA-1: CA3B4E7D7C2EAD0A7DE19148033A8E0DF4F</div><div>SHA-256: 8EAC00DE20A20F474E803880A7A144BA26F05A81C18D1A7074B21DC3</div><div>SHA-384: 78C2B935C90B47CE75E25F80687C08ACD07D03FC0BC0C0F086F0E815A2B9E0C37</div><div>SHA-512: 3EAC2C8F5A906742F3F8089FCD47D0F000817A0AB07C3AE4B33608E2A0F5AFA0A08F</div></div></div>

## A.2 `echo -n <plaintext>' | openssl md5`

```

[~(b00167321# kali)-[~]
$ echo -n 'Falkirk' | openssl md5
MD5(stdin)= 48e935332aadc763f2c82cdb4601a25
[~(b00167321# kali)-[~]
$ echo -n 'Edinburgh' | openssl md5
MD5(stdin)= 03cf54d8ce19777b12732b8c50b3b66f
[~(b00167321# kali)-[~]
$ echo -n 'Glasgow' | openssl md5
MD5(stdin)= d586293d554981ed611ab7b01316d2d5
[~(b00167321# kali)-[~]
$ echo -n 'Stirling' | openssl md5
MD5(stdin)= ee19033300a54df2fa41db9881b4b723
[~(b00167321# kali)-[~]
$

```

"Falkirk"	48E935332AADEC763F2C82CDB4601A25
"Edinburgh"	03CF54D8CE19777B12732B8C50B3B66F
"Glasgow"	D586293D554981ED611AB7B01316D2D5
"Stirling"	EE19033300A54DF2FA41DB9881B4B723

## A.3 Determine the number of hex characters for the hash signatures defined

Hash Algorithm	Output (bits)	Formula for hex chars = bits ÷ 4	Hex Chars
MD5	128 bits	$128 \div 4 = 32$	32
SHA-1	160 bits	$160 \div 4 = 40$	40
SHA-256	256 bits	$256 \div 4 = 64$	64
SHA-384	384 bits	$384 \div 4 = 96$	96
SHA-512	512 bits	$512 \div 4 = 128$	128

Each hexadecimal character represents 4 bits (0–F → 16 values → 4 bits).

So, the number of hex characters in a hash directly reflects its bit length:

$$\text{Number of hex characters} = \text{hash bit length} \div 4$$

**That means:**

Longer bit lengths → longer hashes → more secure, since there are more possible combinations.

**Relation:** The number of hexadecimal characters increases linearly with the bit-length of the hash – each hex digit encodes 4 bits of data.

#### A.4 openssl passwd -apr1 -salt <salt> <password>

```

kali~# openssl passwd -apr1 -salt waZS/8Tm napier
$apr1$waZS/8Tm$jDZmiZBct/c2hysERcZ3m1

kali~# openssl passwd -apr1 -salt mKfrJquI Ankle123
$apr1$mKfrJquI$Kx0CL9krmqhCu0SHKqp5Q0

kali~# openssl passwd -apr1 -salt Jbe/hClb inkwell
$apr1$Jbe/hClb$/k3A4kjpJyC06BUUaPRKs0

kali~# openssl passwd -apr1 -salt 0GyPhsLi password
$apr1$0GyPhsLi$jTTzW0HNS4CI5ZEoyFLjB.

kali~# openssl passwd -1 -salt rq0IRBBM napier
$1$rq0IRBBM$R2pOQH9egTTVN1N1st2U7.

kali~#

```

bill:\$apr1\$waZS/8Tm\$jDZmiZBct/c2hysERcZ3m1	napier
mike:\$apr1\$mKfrJquI\$Kx0CL9krmqhCu0SHKqp5Q0	Ankle123
fred:\$apr1\$Jbe/hClb\$/k3A4kjpJyC06BUUaPRKs0	inkwell
ian:\$apr1\$0GyPhsLi\$jTTzW0HNS4CI5ZEoyFLjB.	password
jane:\$1\$rqOIRBBM\$R2pOQH9egTTVN1N1st2U7.	napier

#### A.5 openssl md5 <file>

```

kali~# openssl md5 1.txt
MD5(1.txt)= 5d41402abc4b2a76b9719d911017c592

kali~# openssl md5 2.txt
MD5(2.txt)= e3fc91b12a36c2334ebb5b6caa2d75b

kali~# openssl md5 3.txt
MD5(3.txt)= fea0f1f6fede90bd0a925b4194deac11

kali~# openssl md5 4.txt
MD5(4.txt)= d89b56f81cd7b82856231e662429bcf2

kali~#

```

MD5(1.txt)= 5d41402abc4b2a76b9719d911017c592	Match
----------------------------------------------	-------

MD5(2.txt)= 69faab6268350295550de7d587bc323d	Not Match
MD5(3.txt)= fea0f1f6fede90bd0a925b4194deac11	Match
MD5(4.txt)= d89b56f81cd7b82856231e662429bcf2	Match

A.6 `cat <file> | openssl md5`

```
[~(b00167321# kali)-[~]
$ cat letter_of_rec.ps | openssl md5
md5(stdin)= a25f7f0b29ee0b3960c860730533a4b9
[~(b00167321# kali)-[~]
$ cat order.ps | openssl md5
md5(stdin)= a25f7f0b29ee0b3960c860730533a4b9
[~(b00167321# kali)-[~]
$
```

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)

### B.1 Run the hashcat benchmark

MD5	<code>hashcat -b -m 0</code>	80839.8 KH/s	
SHA-1	<code>hashcat -b -m 100</code>	23960.0 KH/s	
SHA-256	<code>hashcat -b -m 1400</code>	15664.7 KH/s	
APR1	<code>hashcat -b -m 1600</code>	10532 H/s	

### B.2 `hashcat -m 0 hash1 words`

```

See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 90c

Host memory allocated for this attack: 512 MB (1062 MB free)

Dictionary cache built:
* Filename.: words
* Passwords.: 4
* Bytes.....: 33
* Keyspace.: 4
* Runtime....: 0 secs

The wordlist or mask that you are using is too small.
This means that hashcat cannot use the full parallel power of your device(s).
hashcat is expecting at least 2048 base words but only got 0.2% of that.
Unless you supply more work, your cracking speed will drop.
For tips on supplying more work, see: https://hashcat.net/faq/morework

Approaching final keyspace - workload adjusted.

232dd5d7274e0d662f36c575a3bd634c:napier
5f4dcc3b5aa765d61d8327deb882cf99:password
6d5875265d1979bdad1c8a8f383c5ff5:Ankle123
04013f79accfec9b673095fc6f20698d:inkwell

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 0 (MD5)
Hash.Target....: hash1
Time.Started...: Tue Nov 4 12:11:38 2025 (0 secs)
Time.Estimated.: Tue Nov 4 12:11:38 2025 (0 secs)
Kernel.Feature.: Pure Kernel (password length 0-256 bytes)
Guess.Base.....: File (words)
Guess.Queue....: L/1 (100.00%)
Speed.#01.....: 17 H/s (0.01ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered.....: 4/4 (100.00%) Digests (total), 4/4 (100.00%) Digests (new)
Progress.....: 4/4 (100.00%)
Rejected.....: 0/4 (0.00%)
Restore.Point...: 0/4 (0.00%)
Restore.Sub.#01.: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#01...: napier -> inkwell
Hardware.Mon.#01.: Util: 50%

Started: Tue Nov 4 12:11:07 2025
Stopped: Tue Nov 4 12:11:40 2025

$

```

232dd5d7274e0d662f36c575a3bd634c	napier
5f4dcc3b5aa765d61d8327deb882cf99	password

6d5875265d1979bdad1c8a8f383c5ff5	Ankle123
04013f78accfec9b673005fc6f20698d	inkwell

B.3 hashcat -m 0 hash2 fruits

```
See the above message to find out about the exact limits.
Watchdog: Temperature abort trigger set to 90c
Host memory allocated for this attack: 512 MB (2129 MB free)

Dictionary cache hit:
* Filename.: fruits
* Passwords.: 30
* Bytes.....: 234
* Keyspace...: 30

The wordlist or mask that you are using is too small.
This means that hashcat cannot use the full parallel power of your device(s).
Hashcat is expecting at least 2048 base words but only got 1.5% of that.
Unless you supply more work, your cracking speed will drop.
For tips on supplying more work, see: https://hashcat.net/faq/morework

Approaching final keyspace - workload adjusted.
1f3870bc274f6c49b3e31a0c6728957f:apple
fe01d67a082d4fa0f3ac08429b142eccd:orange
72b3024f237a228a75730123efe77c41:banana
8893dc16b1b2534bab7b03272155a2bb:pear
88956d493572d538878ce1578567b91a:peach

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 0 (MD5)
Hash.Target.....: hash2
Time.Started....: Tue Nov  4 12:33:02 2025 (0 secs)
Time.Estimated..: Tue Nov  4 12:33:02 2025 (0 secs)
Kernel.Feature..: Pure Kernel (password length 0-256 bytes)
Guess.Base.....: File (fruits)
Guess.Queue.....: 1/1 (100.00%)
Speed.#01.....: 16886 H/s (0.03ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered.....: 5/5 (100.00%) Digests (total), 5/5 (100.00%) Digests (new)
Progress.....: 30/30 (100.00%)
Rejected.....: 0/30 (0.00%)
Restore.Point...: 0/30 (0.00%)
Restore.Sub.#01.: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#01..: lenon -> melon
Hardware.Mon.#01.: Util: 35%

Started: Tue Nov  4 12:33:01 2025
Stopped: Tue Nov  4 12:33:04 2025

(h00167321# kali)-[~]
└─$
```

FE01D	orange
1F387	apple
72B30	banana
8893D	pear
88956	peach

## B.4 hashcat -m 1400 file.txt words.txt

```
Matchdog: Temperature abort trigger set to 90c
Host memory allocated for this attack: 512 MB (1841 MB free)

Dictionary cache built:
* Filename.: words.txt
* Passwords.: 5
* Bytes.....: 38
* Keyspace...: 5
* Runtime....: 0 secs

The wordlist or mask that you are using is too small.
This means that hashcat cannot use the full parallel power of your device(s).
Hashcat is expecting at least 2048 base words but only got 0.2% of that.
Unless you supply more work, your cracking speed will drop.
For tips on supplying more work, see: https://hashcat.net/faq/morework

Approaching final keyspace - workload adjusted.

106a5842fc5fce6f663176285ed1516dbb1e3d15c05abab12fdca46d60b539b7:help
Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 1400 (SHA2-256)
Hash.Target.....: 106a5842fc5fce6f663176285ed1516dbb1e3d15c05abab12fd...b539b7
Time.Started....: Tue Nov 4 12:44:57 2025 (0 secs)
Time.Estimated...: Tue Nov 4 12:44:57 2025 (0 secs)
Kernel.Feature...: Pure Kernel (password length 0-256 bytes)
Guess.Base.....: File (words.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#01.....: 2694 H/s (0.01ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 5/5 (100.00%)
Rejected.....: 0/5 (0.00%)
Restore.Point...: 0/5 (0.00%)
Restore.Sub.#01..: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#01...: napier -> help
Hardware.Mon.#01.: Util: 23%

Started: Tue Nov 4 12:44:30 2025
Stopped: Tue Nov 4 12:44:59 2025

└─(b09167321# kali)-[~]
└─$ hashcat --show -m 1400 file.txt
106a5842fc5fce6f663176285ed1516dbb1e3d15c05abab12fdca46d60b539b7:help
└─(b09167321# kali)-[~]
└─$
```

```
hashcat --show -m 1400 file.txt
```

```
106a5842fc5fce6f663176285ed1516dbb1e3d15c05abab12fdca46d60b539b7:help
```

## B.5 hashcat -m 1000 file.txt words.txt

```
See the above message to find out about the exact limits.

Matchdog: Temperature abort trigger set to 90c
Host memory allocated for this attack: 512 MB (1855 MB free)

Dictionary cache hit:
* Filename.: words.txt
* Passwords.: 5
* Bytes.....: 38
* Keyspace...: 5

The wordlist or mask that you are using is too small.
This means that hashcat cannot use the full parallel power of your device(s).
Hashcat is expecting at least 2048 base words but only got 0.2% of that.
Unless you supply more work, your cracking speed will drop.
For tips on supplying more work, see: https://hashcat.net/faq/morework

Approaching final keyspace - workload adjusted.

0333c27eb4b9401d91fef02a9f74840e:help
Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 1000 (NTLM)
Hash.Target.....: 0333c27eb4b9401d91fef02a9f74840e
Time.Started....: Tue Nov 4 12:56:54 2025 (0 secs)
Time.Estimated...: Tue Nov 4 12:56:54 2025 (0 secs)
Kernel.Feature...: Pure Kernel (password length 0-256 bytes)
Guess.Base.....: File (words.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#01.....: 13804 H/s (0.09ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 5/5 (100.00%)
Rejected.....: 0/5 (0.00%)
Restore.Point...: 0/5 (0.00%)
Restore.Sub.#01..: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#01...: napier -> help
Hardware.Mon.#01.: Util: 50%

Started: Tue Nov 4 12:56:25 2025
Stopped: Tue Nov 4 12:56:56 2025

└─(b09167321# kali)-[~]
└─$ hashcat --show -m 1000 file.txt
0333c27eb4b9401d91fef02a9f74840e:help
└─(b09167321# kali)-[~]
└─$
```

```
hashcat --show -m 1000 file.txt
```

```
0333c27eb4b9401d91fef02a9f74840e:help
```

```
hashcat -m 1400 file.txt teams.txt
```

```

Filename...: teams.txt
Passwords...: 4
Bytes.....: 38
Speed.....: 4
Runtime....: 0 secs

The wordlist or mask that you are using is too small.
This means that hashcat cannot use the full parallel power of your device(s).
It is depending at least 2240 base words but only got 0.2% of that.
Unless you supply more work, your cracking speed will drop.
For tips on supplying more work, see: https://hashcat.net/faq/morework

Approaching final keypace - workload adjusted.

c3545f9503029f7c24041f17eb80da0c25bdc1770e1dd41710c592c8929ba37ee9:celtic
bf6862460a29657f55a2860407969c3af183c889021309091c815f6c6b240a:nothereu11
bc5f19a8b45e72eb49cf00b3dbd173cbf914835281fdd6d745a2b680e17d50:aberdenn
c3a16a68c94ac8298c9c2329539a4a1130b6fed247a50424b704019f1d968:livingston

Session.....: hashcat
Status.....: Cracked
Hash_Mode.....: 1400 (Shn2k-256)
Hash_Target....: File.txt
Time_Started...: Tue Nov 4 13:03:36 2025 (0 secs)
Time_Estimated...: Tue Nov 4 13:03:36 2025 (0 secs)
Kernel_Feature...: Pure Kernel (password length 0-256 bytes)
Guess_Base.....: File (teams.txt)
Guess_Queue....: 1/1 (100.00%)
Speed_H01.....: 1514 H/s (0.01ms) @ Accel: 1/4 Loops: 1 Thr: 1 Digests
Recovered.....: 4/4 (100.00%) Digests (total): 4/4 (100.00%) Digests (new)
Progress.....: 4/4 (100.00%)
Rejected.....: 0/4 (0.00%)
Restore_Point...: 0/4 (0.00%)
Restore_Sub_H01...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates_Engine: Device Generator
Candidates_H01...: celtic -> livingston
Hardware_Mon_H01...: Util: 39%

Started: Tue Nov 4 13:03:32 2025
Stopped: Tue Nov 4 13:03:38 2025

[000167321] kali)-$
$ hashcat -show m 1400 f1ls.txt
c3545f9503029f7c24041f17eb80da0c25bdc1770e1dd41710c592c8929ba37ee9:celtic
bf6862460a29657f55a2860407969c3af183c889021309091c815f6c6b240a:nothereu11
bc5f19a8b45e72eb49cf00b3dbd173cbf914835281fdd6d745a2b680e17d50:aberdenn
c3a16a68c94ac8298c9c2329539a4a1130b6fed247a50424b704019f1d968:livingston

[000167321] kali)-$
$

```

```
hashcat --show -m 1400 file.txt
```

635450503029fc2484f1d7eb80da8e25bdc1770e1dd14710c592c8929ba37ee9:celtic

bef68628460a29657f55a2860407969e3af183e889021b30091c815f6c6b248d:motherwell

bc5fb9abe8d5e72eb49cf00b3dbd173cbf914835281fadd674d5a2b680e47d50:aberdeen

6ac16a68ac94ca8298c9c2329593a4a4130b6fed2472a98424b7b4019ef1d968:livingston

```
hashcat -a 3 -m 1400 file.txt ?1?1?1?1?1?1?1?1 --increment
```

**-a 3 → attack mode 3 = brute-force**

**-m 1400 → SHA-256**

**?l → lowercase letter**

**--increment** → starts with 1 char and gradually increases length (so 1-8 letters)

[illegible]

```
[root@kali ~]# cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 64 | xargs sha1sum
```

```
hashcat --show -m 1400 file.txt
```

4dc2159bba05da394c3b94c6f54354db1f1f43b321ac4bbdfc2f658237858c70:hair

0282d9b79f42c74c1550b20ff2dd16aafc3fe5d8ae9a00b2f66996d0ae882775:face

47c215b5f70eb9c9b4bcb2c027007d6cf38a899f40d1d1da6922e49308b15b69:eye

```

Status.....: Exhausted
Hash.Mode.....: 1400 (SHA2-256)
Hash.Target.....: file.txt
Time.Started....: Tue Nov  4 16:57:25 2025 (0 secs)
Time.Estimated...: Tue Nov  4 16:57:25 2025 (0 secs)
Kernel.Feature...: Pure Kernel (password length 0-256 bytes)
Guess.Mask.....: ?l[1]
Guess.Queue.....: 1/8 (12.50%)
Speed.#01.....: 1294 H/s (0.01ms) @ Accel:1024 Loops:16 Thr:1 Vec:8
Recovered.....: 0/3 (0.00%) Digests (total), 0/3 (0.00%) Digests (new)
Progress.....: 26/26 (100.00%)
Rejected.....: 0/26 (0.00%)
Restore.Point....: 1/1 (100.00%)
Restore.Sub.#01...: Salt:0 Amplifier:16-26 Iteration:0-16
Candidate.Engine..: Device Generator
Candidates.#01...: i → x
Hardware.Mon.#01.: Util: 27%

Status.....: Exhausted
Hash.Mode.....: 1400 (SHA2-256)
Hash.Target.....: file.txt
Time.Started....: Tue Nov  4 16:57:25 2025 (0 secs)
Time.Estimated...: Tue Nov  4 16:57:25 2025 (0 secs)
Kernel.Feature...: Pure Kernel (password length 0-256 bytes)
Guess.Mask.....: ?l?l[2]
Guess.Queue.....: 2/8 (25.00%)
Speed.#01.....: 80954 H/s (2.20ms) @ Accel:1024 Loops:16 Thr:1 Vec:8
Recovered.....: 0/3 (0.00%) Digests (total), 0/3 (0.00%) Digests (new)
Progress.....: 676/676 (100.00%)
Rejected.....: 0/676 (0.00%)
Restore.Point....: 26/26 (100.00%)
Restore.Sub.#01...: Salt:0 Amplifier:16-26 Iteration:0-16
Candidate.Engine..: Device Generator
Candidates.#01...: ia → xq
Hardware.Mon.#01.: Util: 73%

Approaching final keyspace - workload adjusted.
47c215b5f70eb9c9b4bcb2c027007d6cf38a899f40d1da6922e49308b15b69:eye

Session.....: hashcat
Status.....: Exhausted
Hash.Mode.....: 1400 (SHA2-256)
Hash.Target.....: file.txt
Time.Started....: Tue Nov  4 16:57:26 2025 (0 secs)
Time.Estimated...: Tue Nov  4 16:57:26 2025 (0 secs)
Kernel.Feature...: Pure Kernel (password length 0-256 bytes)
Guess.Mask.....: ?l?l?l[3]
Guess.Queue.....: 3/8 (37.50%)
Speed.#01.....: 9077.0 KH/s (0.60ms) @ Accel:1024 Loops:16 Thr:1 Vec:8
Recovered.....: 1/3 (33.33%) Digests (total), 1/3 (33.33%) Digests (new)
Progress.....: 17576/17576 (100.00%)
Rejected.....: 0/17576 (0.00%)
Restore.Point....: 676/676 (100.00%)
Restore.Sub.#01...: Salt:0 Amplifier:16-26 Iteration:0-16
Candidate.Engine..: Device Generator
Candidates.#01...: iar → xqx
Hardware.Mon.#01.: Util: 57%

0282d9b79f42c74c1550b20ff2dd16aafc3fe5d8ae9a00b2f66996d0ae882775:face
4dc2159bba05da394c3b94c6f54354db1f1f43b321ac4bbdfc2f658237858c70:hair

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 1400 (SHA2-256)
Hash.Target.....: file.txt
Time.Started....: Tue Nov  4 16:57:26 2025 (0 secs)
Time.Estimated...: Tue Nov  4 16:57:26 2025 (0 secs)
Kernel.Feature...: Pure Kernel (password length 0-256 bytes)
Guess.Mask.....: ?l?l?l?l[4]
Guess.Queue.....: 4/8 (50.00%)
Speed.#01.....: 2189.1 KH/s (7.61ms) @ Accel:1024 Loops:16 Thr:1 Vec:8
Recovered.....: 3/3 (100.00%) Digests (total), 3/3 (100.00%) Digests (new)
Progress.....: 192512/456976 (42.13%)
Rejected.....: 0/192512 (0.00%)
Restore.Point....: 6144/17576 (34.96%)
Restore.Sub.#01...: Salt:0 Amplifier:0-16 Iteration:0-16
Candidate.Engine..: Device Generator
Candidates.#01...: snfs → nolc
Hardware.Mon.#01.: Util: 73%

```

## Number of tests for each sequence tried:

a → z: 26/26

aa → zz: 676/676

aaa → zzz: 17576/17576

aaaa → zzzz: 192512/456976

## Number of tests (combinations) for lowercase-only sequences:

a → z (1 chars): 26

aa → zz (2 chars):  $26^2 = 676$

aaa → zzz (3 chars):  $26^3 = 17,576$

aaaa → zzzz (4 chars):  $26^4 = 456,976$

8 letters:  $26^8 = 208,827,064,576$

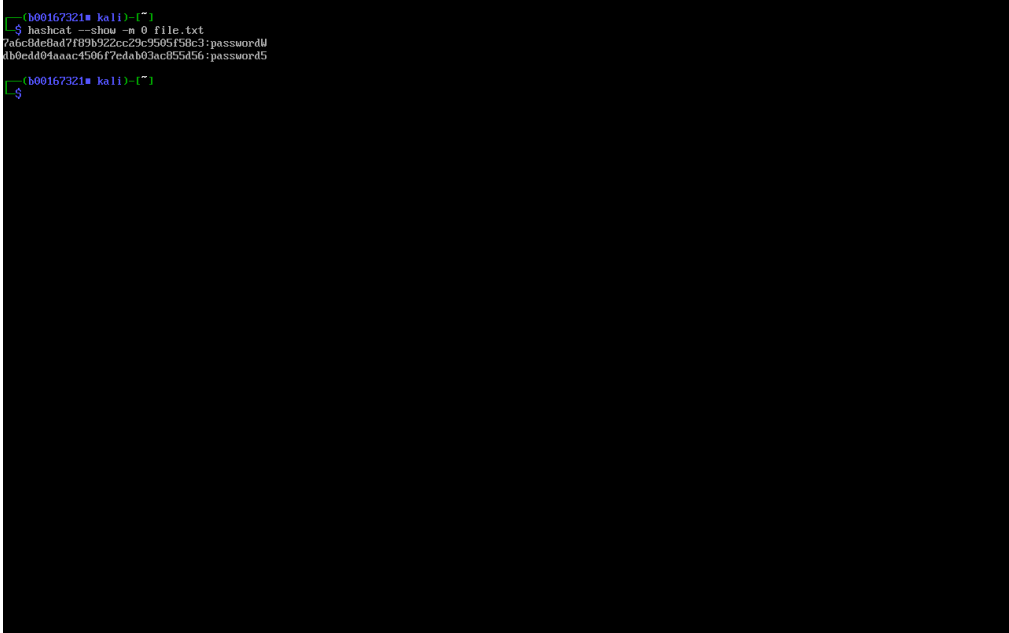
(General rule: for length  $n$  there are  $26^n$  combinations when using only lowercase letters.)

## What happens when you take the “--increment” flag away?

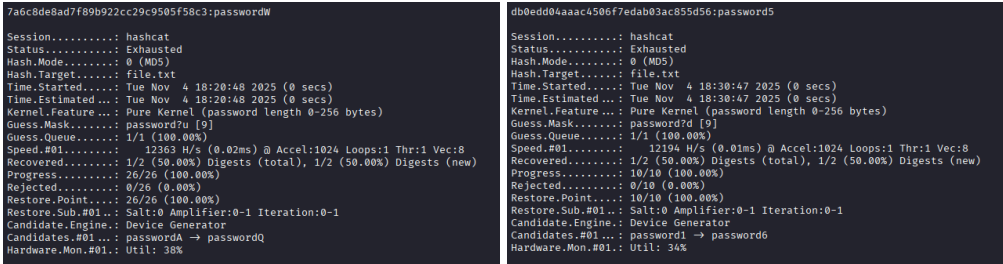
If to omit `--increment`, Hashcat only tries the exact length of the mask given (here, 8 letters).

So it would skip all shorter guesses (a–z, aa–zzz, etc.) and jump straight to 8-character words.

B.8 hashcat -a 3 -m 0 file.txt password?l  
hashcat -a 3 -m 0 file.txt password?u  
hashcat -a 3 -m 0 file.txt password?d



hashcat --show -m 0 file.txt
7a6c8de8ad7f89b922cc29c9505f58c3:passwordW
db0edd04aaac4506f7edab03ac855d56:password5



Number of tests: 26/26 & 10/10

### 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, crypt 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/escurity/tree/master/unit03\\_hashing](https://github.com/billbuchanan/escurity/tree/master/unit03_hashing)

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

Demo: <https://youtu.be/mTLr6Ubf0>

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):  <b>Web link (Hashing):</b> <a href="http://asecuritysite.com/encryption/md5">http://asecuritysite.com/encryption/md5</a>  Match the hash signatures with their words ("Falkirk", "Edinburgh", "Glasgow" and "Stirling").  03CF5408CE1977781273288C5083866F 05862930554981ED611A87013160205 48E93332A8C76392C8C084601A25 EE19033300A54DF2FA1D89881B40723	03CF5: Is it [Falkirk][Edinburgh][Glasgow][Stirling]?  D5862: Is it [Falkirk][Edinburgh][Glasgow][Stirling]?  48E93: Is it [Edinburgh][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 [Edinburgh][Edinburgh][Glasgow][Stirling]?  EE190: Is it [Falkirk][Edinburgh][Glasgow][Stirling]?
A.3	Using:	MD5 hex chars:

1

	<b>Web link (Hashing):</b> <a href="http://asecuritysite.com/encryption/md5">http://asecuritysite.com/encryption/md5</a>  Determine the number of hex characters for the hash signatures defined. Note: perhaps copy and paste your hash to an on-line character counter?  SHA-1 hex chars: 32 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 represents 4 bits of data	
A.4	For the following (/etc/shadow file, determine the matching password:  b1ll:5apr15mZS/8Tm5j02m12Rct/c2hy5eRc23m1 mike:5apr15mZS/8Tm5j02m12Rct/c2hy5eRc23m1 fred:5apr152be/hc1D5/k3A4k.jp3yC06BUuAPKs0 ian:5apr150gyPm15j1T7Zm0m5G4C52zyr1j8. jane: 51\$rg0tR8BN5R2p0qH9egTTVN1N1st2U7.  [Hint: openssl passwd -apr1 -salt ZzS/8TF napier]	The passwords are <b>password</b> , <b>napier</b> , <b>inkwell</b> and <b>Ankle123</b> .  Bill's password: <b>napier</b> Mike's password: <b>Ankle123</b> Fred's password: <b>inkwell</b> Ian's password: <b>password</b> Jane's password: <b>napier</b>
A.5	From Ubuntu, download the following:  <b>Web link (Files):</b> <a href="http://asecuritysite.com/files02.zip">http://asecuritysite.com/files02.zip</a>  (a quick way to download is wget <a href="http://asecuritysite.com/files02.zip">asecuritysite.com/files02.zip</a> ) and the files should have the following MD5 signatures:  MD5(1.txt) = 5d41402abc4b2a76b9719d911017c592 MD5(2.txt) = 69faab6268350295550de7d587bc323d MD5(3.txt) = fea0f1f6feae90b0a92504194daec11 MD5(4.txt) = d89b56f81cd7b82856231e662429bcf2	Which file(s) have been modified? <b>2.txt</b>  MD5 (2.txt) = 69faab6268350295550de7d587bc323d  The two different files have the same MD5 signature
A.6	From Ubuntu, download the following ZIP file:  <b>Web link (PS Files):</b> <a href="http://asecuritysite.com/letters.zip">http://asecuritysite.com/letters.zip</a> (a quick way to download is wget <a href="http://asecuritysite.com/letters.zip">asecuritysite.com/letters.zip</a> ) 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? <b>Yes</b>  Now determine the MD5 signature for them. What can you observe from the result?  The two different files have the same MD5 signature

2

#### 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: <b>44320.0 MD/s</b> Hash rate for SHA-1: <b>23809.0 MD/s</b> Hash rate for SHA-256: <b>15864.7 MD/s</b> Hash rate for APR1: <b>8632.0 MD/s</b>
B.2	On Ubuntu, next create a word file ( <b>words</b> ) with the words of "napier", "password", "Ankle123" and "inkwell".  Using hashcat crack the following MD5 signatures (hash1):  23200507274E0D662F36C575A380634C 5F4DC...CF99 Is it [napier][password][Ankle123][inkwell]?  23200507274E0D662F36C575A380634C 5F4DC385AA765061D83270E8882CF99 6058757450137780A01C4A8F383CF95 04013F78ACFC98673005FC6F206980  Command used: hashcat -m 0 hash1 words	2320D...634C Is it [napier][password][Ankle123][inkwell]?  5F4DC...CF99 Is it [napier][password][Ankle123][inkwell]?  60587...5FF5 Is it [napier][password][Ankle123][inkwell]?  04013...698D Is it [napier][password][Ankle123][inkwell]?  FE01D: <b>crack</b> 1F387: <b>napier</b> 72830: <b>password</b> 8893D: <b>ink</b> 88956: <b>well</b>
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):  FE01D67A0020A0F3AC084298142ECCD 1F38708E274F6C49B3E31A0C6728957F 7283028F29A228A75730123FEFE7C41 8893D61681825348A07803727145A208 889560D93572D538078C1578567891A	FE01D: <b>crack</b> 1F387: <b>apple</b> 72830: <b>banana</b> 8893D: <b>pear</b> 88956: <b>peach</b>
B.4	Put this SHA-256 value in a file named file.txt:  106a5842fc5ceb663176285ed1516dbb 1e3d15c05ab017f6c4a660b539b7  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	<b>106a5842fc5ceb663176285ed1516dbb1e3d15c05ab017f6c4a660b539b7 help</b>
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	<b>0333c27eb4b9401d91fef02a9f74840e help</b>

The cracked hashes are stored in:

3

~/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 re-crack them, just use:

rm ~/hashcat/hashcat.potfile

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

635450503029FC2484F1d7eb80da8e25bdc1770e1dd14710c592c8929ba37ee9  
8EF68C8460A295775A2860407360834F181E8800183009C313FE6C02480  
bc5f9ab8e8d5e72eb49cF00b3dbd173cbf914835281fad674d5a2b680e47d50  
6ac16a68ac94ca8298c9c232953a4a4130b6fd2472a98424b7b4019ef1d968

Football teams:	<b>celtic motherwell dunfermline livingston</b>
-----------------	-------------------------------------------------------------

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 ?[?]?[?]?[?]?[?]?[?] --increment

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

4dc2159ba05da394c3b94c6f54354db1f1f43b321ac4bbdfc2f658237858c70  
0282c9b79f41c74c1550b20f2d016aafc3fe5d8aeb00b2f66996b0a882775  
47c215b5f70eb9c9b4cb2c027007d6cf38a899f40d1da6922e49308b15b69

Words:  
**4dc2159ba05da394c3b94c6f54354db1f1f43b321ac4bbdfc2f658237858c70 help  
0282c9b79f41c74c1550b20f2d016aafc3fe5d8aeb00b2f66996b0a882775 face  
47c215b5f70eb9c9b4cb2c027007d6cf38a899f40d1da6922e49308b15b69 eye**

Number of tests for each sequence tried:

a->Z: **26/26**  
aa->ZZ: **676/676**  
aaa->ZZZ: **17576/17576**  
aaaa->ZZZZ: **456512/456512**

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

**If it omits --increment, hashcat only tries the exact length of the hash 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:

4

```
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?g
```

Using these commands, crack the following:

```
7a6c8de8ad7f89b922cc29c9505f58c3  
db0edd04aac4506f7edab03ac855d56
```

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

Words: `7a6c8de8ad7f89b922cc29c9505f58c3.password`  
`db0edd04aac4506f7edab03ac855d56.password`

Number of tests for each: `10000`  
`10000`