

Secure Communications

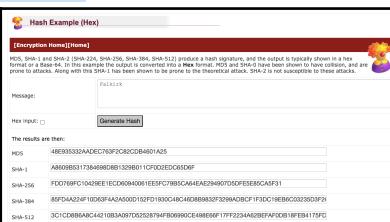
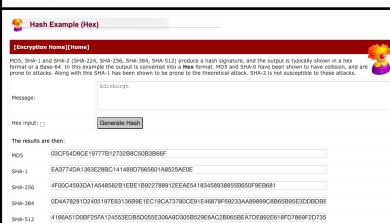
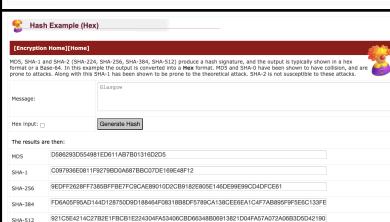
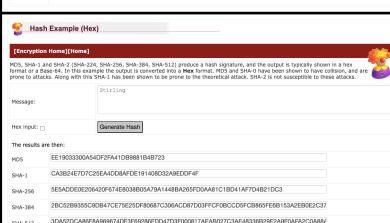
Week 8

Hashing

Sections

A. Hashing

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

“Falkirk”	48E935332AADEC763F2C 82CDB4601A25	
“Edinburgh”	03CF54D8CE19777B1273 2B8C50B3B66F	
“Glasgow”	D586293D554981ED611A B7B01316D2D5	
“Stirling”	EE19033300A54DF2FA41 DB9881B4B723	

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

```
(b00167321# kali) [~]
$ echo -n 'Falkirk' | openssl md5
MD5(stdin)= 48e935332aadec763f2c82cda4601a25

(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>

```
[b00167321# kali] ~$ openssl passwd -apr1 -salt waZS/8Tm$jDZmiZBct/c2hysERcZ3m1
$apr1waZS/8Tm$jDZmiZBct/c2hysERcZ3m1

[b00167321# kali] ~$ openssl passwd -apr1 -salt mKfrJquI ankle123
$apr1mKfrJquI$Kx0CL9krmqhCu0SHKqp5Q0

[b00167321# kali] ~$ openssl passwd -apr1 -salt Jbe/hC1b inkwell
$apr1Jbe/hC1b$K3A4kjPjyC06BUUaPRKs0

[b00167321# kali] ~$ openssl passwd -apr1 -salt 0GyPhsLi password
$apr1$0GyPhsLi$TTzW0HNS4CI5ZEoyFLjB.

[b00167321# kali] ~$ openssl passwd -1 -salt rq0IRBBN napier
$1$rq0IRBBN$R2pOQH9egTTVN1N!st2U7.

[b00167321# kali] ~$
```

bill:\$apr1\$waZS/8Tm\$jDZmiZBct/c2hysERcZ3m1	napier
mike:\$apr1\$mKfrJquI\$Kx0CL9krmqhCu0SHKqp5Q0	Ankle123
fred:\$apr1\$Jbe/hC1b\$/k3A4kjPjyC06BUUaPRKs0	inkwell
ian:\$apr1\$0GyPhsLi\$TTzW0HNS4CI5ZEoyFLjB.	password
jane:\$1\$rq0IRBBN\$R2pOQH9egTTVN1N!st2U7.	napier

A.5 openssl md5 <file>

```
[b00167321# kali] ~$ openssl md5 1.txt
MD5(1.txt)= 5d41402abc4b2a76b9719d911017c592

[b00167321# kali] ~$ openssl md5 2.txt
MD5(2.txt)= e3fc91b12a36c234eb5b66caa2d75b

[b00167321# kali] ~$ openssl md5 3.txt
MD5(3.txt)= fea0f1f6fedc90bd0a925b4194deac11

[b00167321# kali] ~$ openssl md5 4.txt
MD5(4.txt)= d89b56f81cd7b82856231e662429bcf2

[b00167321# kali] ~$
```

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

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

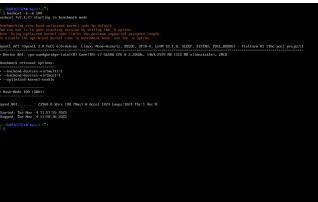
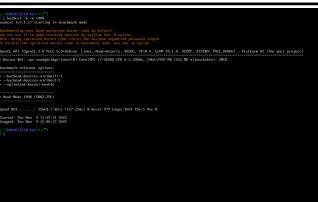
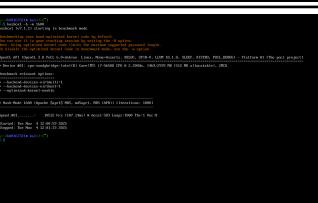
A.6 `cat <file> | openssl md5`

```
(b00167321# kali)-[*]
$ cat letter_of_rec.ps | openssl md5
MD5(stdin)= a25f7f0b29ee0b3968c866738533a4b9
(b00167321# kali)-[*]
$ cat order.ps | openssl md5
MD5(stdin)= a25f7f0b29ee0b3968c866738533a4b9
(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
* Keystpace.: 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 keystpace - workload adjusted.

232dd5d7274e0d662f36c575a3bd634c:napier
5f4dcc3b5aa765d61d8327deb882cf99

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 Kernels (password length 0-256 bytes)
Guess.Base....: File (words)
Guess.Rate....: 1/1 (100.00%)
Speed.#01....: 17 H/s (0.01ms) @ Accel:1024 Loops:1 Thr:1 Uec: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.: UHII: 50%
```

(h00167321# kali) [~]

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.
hashdog: 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!
In this case, hashcat will not 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 words, your cracking speed will drop.

For tips on supplying more work, see: https://hashcat.net/faq-morework

Approaching final keyspace - workload adjusted.

1f3870be274f6c49b3e31a0c6f728957f:apple
fe01d67a024fa0f3ac0f04290142eccd:orange
72b302bf237a223a75730123cfef7c41:banana
8893dc1b61b2534bab7b03727145a2bb:pear
009560d93572d53007bc0c157b567b91a: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.Mask....: 1/1 (1 password)
Speed.M01.....: 16896 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.: lemon -> melon
Hardware.Mon.#01.: Util: 35% 

Started: Tue Nov  4 12:33:01 2025
Stopped: Tue Nov  4 12:33:04 2025
[+] (b00167321㉿kali)-[*]
$
```

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

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

```
watchdog: 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.....: 36
* 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 (SH02-256)
Hash.Target...: 106a5842fc5fce6f663176285ed1516dbb1e3d15c05abab12fdca46d60b539b7
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.....: 2684 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.Sth #01...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#01...: napier -> help
Hardware.Mon.#01.: Util: 29%
Started: Tue Nov 4 12:44:30 2025
Stopped: Tue Nov 4 12:44:59 2025

[~]# hashcat --show -m 1400 file.txt
106a5842fc5fce6f663176285ed1516dbb1e3d15c05abab12fdca46d60b539b7:help
[~]#
```

```
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.

watchdog: 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.....: 36
* 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.....: 13004 H/s (0.00ms) @ 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.Sth #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

[~]# hashcat --show -m 1000 file.txt
0333c27eb4b9401d91fef02a9f74840e:help
[~]#
```

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

```
0333c27eb4b9401d91fef02a9f74840e:help
```

B.6 hashcat -m 1400 file.txt teams.txt

```
[*] Filename...: texts.txt
[*] Passwords : 4
[*] Bytes....: 38
[*] Keaspace...: 4
[*] Runtime...: 0 secs

The wordlist or mask that you are using is too small.
This means that hashcat cannot utilize the full parallel power of your device(s).
hashcat is expecting at least 2048 words in order to only get 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 keaspace - workload adjusted.

635450503629f2c48f41d7eb0badae25bdcc170e01d14710c59c292992a37ee9:celtic
bf6fb02f40a62965f15a206047967691a103e089921b300919c151fc6b24d0nethell
bf5fb9aeb6d72e72eb0013d4bd173cf194855261ad67d45a2b680e7fd56:aberdeen
ac16a6b4a9c9ac0298c3c2359539a41130b6fed2472a98424b74019ef1d968:livington

Session.....: hashcat
Status.....: Cracked
Hash.Mode...: 1400 (SHA2-256)
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)
Hash.Bases....: File (texts.txt)
Guess.Bases...: 1/1 (100.00%)
Recovered....: 1/1 (100.00%) Digests (0.0ms) 0 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.: Salt0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#01.: celtic -> livingston
Hardware.Mon.#01.: Util: 39%

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

(bo0167321:kali1)~`
```

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

635450503029fc2484f1d7eb80da8e25bdc1770e1dd14710c592c8929ba37ee9:celtic

bef68628460a29657f55a2860407969e3af183e889021b30091c815f6c6b248d:motherwell

bc5fb9abe8d5e72eb49cf00b3dbd173cbf914835281fadd674d5a2b680e47d50:aberdeen

6ac16a68ac94ca8298c9c2329593a4a4130b6fed2472a98424b7b4019ef1d968:livingston

```
B.7 hashcat -a 3 -m 1400 file.txt ?1?1?1?1?1?1?1?1?1?1 --increment
```

-a 3 → attack mode 3 = brute-force

-m 1400 → SHA-256

?1 → lowercase letter
--increment → starts with 1 char and gradually increases length (so

```
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 i/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 (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 i/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...: i → x
Hardware.Mon.#01.: Util: 73%
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?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...: snts → nolc
Hardware.Mon.#01.: Util: 75%

```

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

```
(b00167321# kali)-[~]
└─$ hashcat --show -m 0 file.txt
7a6c8de8ad7f89b922cc29c9505f58c3:passwordW
db0edd04aac4506f7edab03ac855d56:password5
└─$
```

```
hashcat --show -m 0 file.txt
```

```
7a6c8de8ad7f89b922cc29c9505f58c3:passwordW
```

```
db0edd04aac4506f7edab03ac855d56:password5
```

```
7a6c8de8ad7f89b922cc29c9505f58c3:passwordW
Session.....: hashcat
Status.....: Hashcat
Hash.Mode....: 0 (MD5)
Hash.Target...: file.txt
Time.Started...: Tue Nov 4 18:20:48 2025 (0 secs)
Time.Estimated...: Tue Nov 4 18:20:48 2025 (0 secs)
Kernel.Feature...: Pure Kernel (password length 0-256 bytes)
Guess.Mask.....: password?u [9]
Guess.Queue....: 1/1 (100.00%)
Speed.#0.....: 12363 H/s (0.02ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered.....: 1/2 (50.00%) Digests (total), 1/2 (50.00%) Digests (new)
Progress.....: 26/26 (100.00%)
Rejected.....: 0/26 (0.00%)
Restore.Point...: 26/26 (100.00%)
Restore.Sub.#01..: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#01...: passwordd → passwordQ
Hardware.Mon.#01.: Util: 38%
```

```
db0edd04aac4506f7edab03ac855d56:password5
Session.....: hashcat
Status.....: Hashcat
Hash.Mode....: 0 (MD5)
Hash.Target...: file.txt
Time.Started...: Tue Nov 4 18:30:47 2025 (0 secs)
Time.Estimated...: Tue Nov 4 18:30:47 2025 (0 secs)
Kernel.Feature...: Pure Kernel (password length 0-256 bytes)
Guess.Mask.....: password?d [9]
Guess.Queue....: 1/1 (100.00%)
Speed.#0.....: 12363 H/s (0.02ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered.....: 1/2 (50.00%) Digests (total), 1/2 (50.00%) Digests (new)
Progress.....: 10/10 (100.00%)
Rejected.....: 0/10 (0.00%)
Restore.Point...: 10/10 (100.00%)
Restore.Sub.#01..: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#01...: password1 → password6
Hardware.Mon.#01.: Util: 34%
```

Number of tests: 26/26 & 10/10


```
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  
db0edd04aaa4506f7edab03ac85d5d6
```

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

Words: **7a6c8de8ad7f89b922cc29c9505f58c3 password00
db0edd04aaa4506f7edab03ac85d5d6 password01**

Number of tests for each: **25/25
18/18**