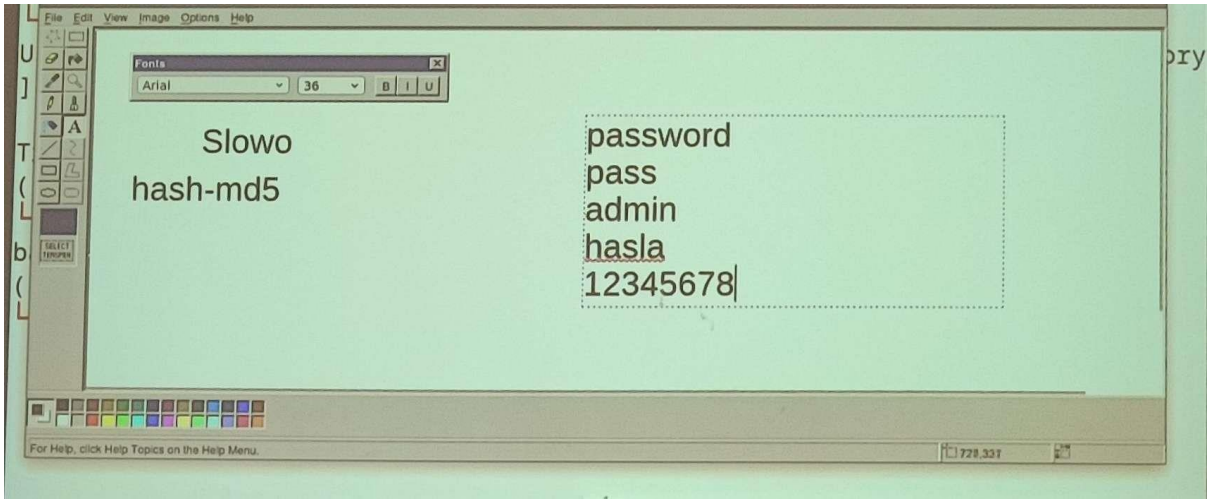
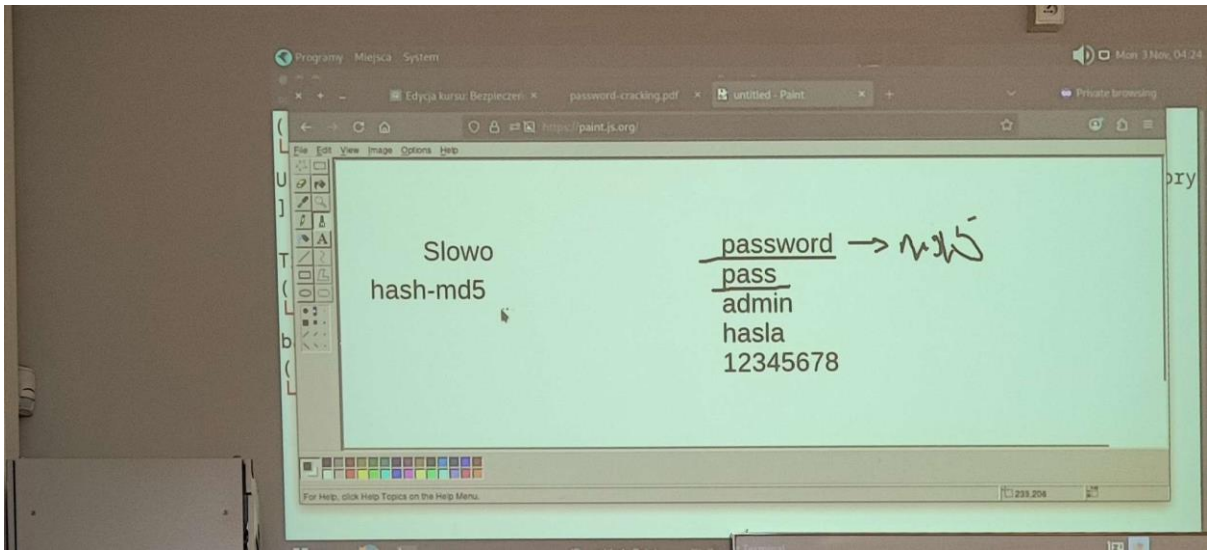

File Edit View Image Options Help
Fonts
Arial
36
B I U
Slowo
hash-md5
password
pass
admin
hasla
12345678
For Help, click Help Topics on the Help Menu.


Programy Miejsca System
Edycja kursu: Bezpieczeń
password-cracking.pdf
untitled - Paint
Private browsing
https://paint.js.org/
File Edit View Image Options Help
Slowo
hash-md5
password
pass
admin
hasla
12345678
For Help, click Help Topics on the Help Menu.

```
(.python_env) ┌─[admin@parrot]─[~]
└──╼ $ls
ciphertext.enc  juice-shop       privkey.pem                 Templates       zad3.txt
ciphertext.txt  key              priv.pem                    Untitled.ipynb  zad5.dec
curl_response   key2             pubkey.pem                  Wideo           zad5.enc
data.enc        key3             public_key_hAOYpPfuP3k.pem  zad1.dec        zad5.txt
decrypted.txt   key5             public_key__-s46RaqOt8.pem  zad1.enc        zad6.dec
Desktop         key6             public.pem                  zad1.txt        zad6.enc
Dokumenty       keys.pem         Publiczny                   zad2.dec        zad6.txt
ecpriv.pem      Muzyka           pub.pem                     zad2.enc        zad7.dec
ecpub.pem       obrazek.png      signature.b64               zad2.txt        zad7.enc
encrypt         Obrazy           slowniki.zip                zad3_4.priv
getprivkey      Pobrane          slowo.sig                   zad3_4.pub
haslo           private_key.pem  slowo.txt                   zad3.dec
IV              private.pem      snap                        zad3.enc
(.python_env) ┌─[admin@parrot]─[~]
└──╼ $unzip slowniki.zip
Archive:  slowniki.zip
   creating: slowniki/
   creating: slowniki/ex41/
  inflating: slowniki/ex41/wordlist.txt
   creating: slowniki/ex416/
  inflating: slowniki/ex416/input.txt
   creating: slowniki/ex417/
  inflating: slowniki/ex417/input.txt
   creating: slowniki/ex418/
```

```
  inflating: slowniki/ex419/input.txt
   creating: slowniki/ex42/
  inflating: slowniki/ex42/wordlist.txt
   creating: slowniki/ex420/
  inflating: slowniki/ex420/input.txt
   creating: slowniki/ex421/
  inflating: slowniki/ex421/wordlist.txt
   creating: slowniki/ex43/
  inflating: slowniki/ex43/wordlist.txt
   creating: slowniki/ex44/
  inflating: slowniki/ex44/wordlist.txt
   creating: slowniki/ex45/
  inflating: slowniki/ex45/wordlist.txt
   creating: slowniki/ex46/
  inflating: slowniki/ex46/wordlist.txt
(.python_env) ┌─[admin@parrot]─[~]
└──╼ $ls
ciphertext.enc  juice-shop       privkey.pem                 snap            zad3.enc
ciphertext.txt  key              priv.pem                    Templates       zad3.txt
curl_response   key2             pubkey.pem                  Untitled.ipynb  zad5.dec
data.enc        key3             public_key_hAOYpPfuP3k.pem  Wideo           zad5.enc
decrypted.txt   key5             public_key__-s46RaqOt8.pem  zad1.dec        zad5.txt
Desktop         key6             public.pem                  zad1.enc        zad6.dec
Dokumenty       keys.pem         Publiczny                   zad1.txt        zad6.enc
ecpriv.pem      Muzyka           pub.pem                     zad2.dec        zad6.txt
ecpub.pem       obrazek.png      signature.b64               zad2.enc        zad7.dec
encrypt         Obrazy           slowniki                    zad2.txt        zad7.enc
getprivkey      Pobrane          slowniki.zip                zad3_4.priv
haslo           private_key.pem  slowo.sig                   zad3_4.pub
IV              private.pem      slowo.txt                   zad3.dec
(.python_env) ┌─[admin@parrot]─[~]
└──╼ $cd slowniki/
(.python_env) ┌─[admin@parrot]─[~/slowniki]
└──╼ $ls
ex41  ex416  ex417  ex418  ex419  ex42  ex420  ex421  ex43  ex44  ex45  ex46
(.python_env) ┌─[admin@parrot]─[~/slowniki]
└──╼ $
```

```
(.python_env) ┌─[admin@parrot]─[~/slowniki]
└──╼ $echo -n "9517951fae507f064c8b5313e2ab8617" > hash.txt
(.python_env) ┌─[admin@parrot]─[~/slowniki]
└──╼ $hashcat --help
hashcat (v6.2.6) starting in help mode

Usage: hashcat [options]... hash|hashfile|hccapxfile [dictionary|mask|directory]...

- [ Options ] -

 Options Short / Long           | Type | Description
| Example
================================+======+=================================================
+======================
 -m, --hash-type                | Num  | Hash-type, references below (otherwise autodetect)
| -m 1000
 -a, --attack-mode              | Num  | Attack-mode, see references below
| -a 3
 -V, --version                  |      | Print version
|
 -h, --help                     |      | Print help
```

```
- [ Attack Modes ] -

  # | Mode
 ===+======
  0 | Straight
  1 | Combination
  3 | Brute-force
  6 | Hybrid Wordlist + Mask
  7 | Hybrid Mask + Wordlist
  9 | Association

- [ Built-in Charsets ] -

  ? | Charset
```

```
- [ Hash modes ] -

      # | Name                                                       | Category
  ======+============================================================+======================
==============
    900 | MD4                                                        | Raw Hash
      0 | MD5                                                        | Raw Hash
    100 | SHA1                                                       | Raw Hash
   1300 | SHA2-224                                                   | Raw Hash
   1400 | SHA2-256                                                   | Raw Hash
  10800 | SHA2-384                                                   | Raw Hash
   1700 | SHA2-512                                                   | Raw Hash
  17300 | SHA3-224                                                   | Raw Hash
  17400 | SHA3-256                                                   | Raw Hash
  17500 | SHA3-384                                                   | Raw Hash
  17600 | SHA3-512                                                   | Raw Hash
   6000 | RIPEMD-160                                                 | Raw Hash
    600 | BLAKE2b-512                                                | Raw Hash
  11700 | GOST R 34.11-2012 (Streebog) 256-bit, big-endian           | Raw Hash
  11800 | GOST R 34.11-2012 (Streebog) 512-bit, big-endian           | Raw Hash
   6900 | GOST R 34.11-94                                            | Raw Hash
  17010 | GPG (AES-128/AES-256 (SHA-1($pass)))                       | Raw Hash
   5100 | Half MD5                                                   | Raw Hash
  17700 | Keccak-224                                                 | Raw Hash
  17800 | Keccak-256                                                 | Raw Hash
  17900 | Keccak-384                                                 | Raw Hash
  18000 | Keccak-512                                                 | Raw Hash
   6100 | Whirlpool                                                  | Raw Hash
  10100 | SipHash                                                    | Raw Hash
     70 | md5(utf16le($pass))                                        | Raw Hash
    170 | sha1(utf16le($pass))                                       | Raw Hash
   1470 | sha256(utf16le($pass))                                     | Raw Hash
```

Łamanie haseł - Dictionary attack

4.1 *Pod adresem* `http://127.0.0.1:4001` *działa prosty serwer HTTP udostępniający dwa endpointy:* `http://127.0.0.1:4001/hash` *oraz* `http://127.0.0.1:4001/submit`.

(a) Uruchom serwer za pomocą poniższego polecenia:

```
docker run -p 4001:4001 --name ex1 docker.io/mazurkatarzyna/pass-cracking-ex1:latest
podman run -p 4001:4001 --name ex1 docker.io/mazurkatarzyna/pass-cracking-ex1:latest
```

Jeśli Docker Hub nie odpowiada, użyj obrazu zapasowego:

```
docker run -p 4001:4001 --name ex1 ghcr.io/mazurkatarzynaumcs/pass-cracking-ex1:latest
podman run -p 4001:4001 --name ex1 ghcr.io/mazurkatarzynaumcs/pass-cracking-ex1:latest
```

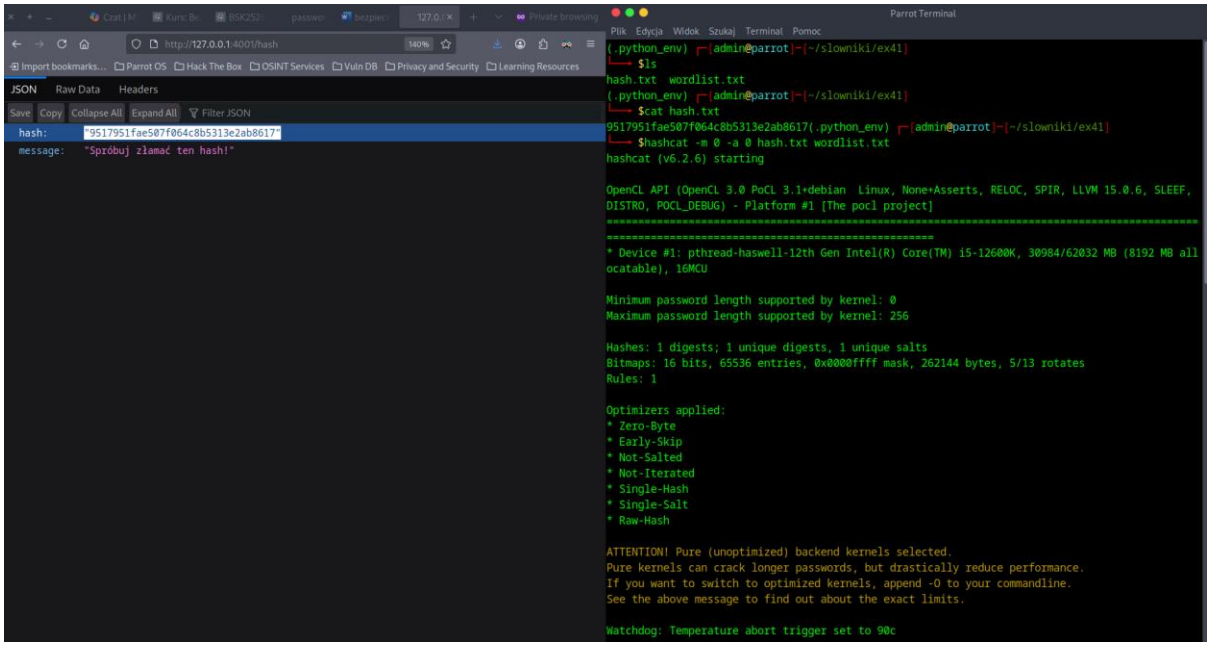
(b) Wyślij request do endpointa `http://127.0.0.1:4001/hash` używając metody HTTP `GET`. Otrzymasz w odpowiedzi **hash MD5** do złamania (jako `hash`).

(c) Wykorzystując narzędzia `hashcat` oraz `John the Ripper` i dostępne **słowniki**, spróbuj złamać hash.

(d) Za pomocą metody HTTP `POST`, wyślij pod endpoint `http://127.0.0.1:4001/submit` wartość złamanego hasha (jako `word`).

(e) W odpowiedzi serwer zwróci odpowiednią informację o sukcesie lub błędzie - zweryfikuje poprawność złamanego hasha.

```
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex41]
└─$ echo -n "9517951fae507f064c8b5313e2ab8617" > hash.txt
```



```
Host memory required for this attack: 4 MB

Dictionary cache built:
* Filename..: wordlist.txt
* Passwords.: 100
* Bytes.....: 984
* Keyspace..: 100
* 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).
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.

9517951fae507f064c8b5313e2ab8617:joseponce8

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 0 (MD5)
Hash.Target......: 9517951fae507f064c8b5313e2ab8617
Time.Started.....: Mon Nov  3 04:43:16 2025 (0 secs)
Time.Estimated...: Mon Nov  3 04:43:16 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (wordlist.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:     3055 H/s (0.04ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 100/100 (100.00%)
Rejected.........: 0/100 (0.00%)
Restore.Point....: 0/100 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: breadda1 -> nishab08
Hardware.Mon.#1..: Temp: 32c Util:  6%
```

```
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.........:     3055 H/s (0.04ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 100/100 (100.00%)
Rejected.........: 0/100 (0.00%)
Restore.Point....: 0/100 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: breadda1 -> nishab08
Hardware.Mon.#1..: Temp: 32c Util:  6%

Started: Mon Nov  3 04:43:03 2025
Stopped: Mon Nov  3 04:43:18 2025
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex41]
└──╼ $hashcat -m 0 hash.txt --show
9517951fae507f064c8b5313e2ab8617:joseponce8
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex41]
└──╼ $cat hash.txt
9517951fae507f064c8b5313e2ab8617(.python_env) ┌─[admin@parrot]─[~/slowniki/ex41]
└──╼ $curl -X POST http://127.0.0.1:4001/submit -H "Content-Type: application/json" -d '{"word
":"morkyra"}'
{"success": false, "message": "Niepoprawne słowo. Spróbuj ponownie!"}(.python_env) ┌─[admin@pa
rrot]─[~/slowniki/ex41]
└──╼ $curl -X POST http://127.0.0.1:4001/submit -H "Content-Type: application/json" -d '{"word
":"joseponce8"}'
{"success": true, "message": "Gratulacje! Poprawnie złamano hash!", "word": "joseponce8", "has
h": "9517951fae507f064c8b5313e2ab8617"}(.python_env) ┌─[admin@parrot]─[~/slowniki/ex41]
└──╼ $
```

4.2 *Pod adresem* `http://127.0.0.1:4002` *działa prosty serwer HTTP udostępniający dwa endpointy:* `http://127.0.0.1:4002/hash` *oraz* `http://127.0.0.1:4002/submit`.

(a) Uruchom serwer za pomocą poniższego polecenia:

```
docker run -p 4002:4002 --name ex2 docker.io/mazurkatarzyna/pass-cracking-ex2:latest
podman run -p 4002:4002 --name ex2 docker.io/mazurkatarzyna/pass-cracking-ex2:latest
```

Jeśli Docker Hub nie odpowiada, użyj obrazu zapasowego:

```
docker run -p 4002:4002 --name ex2 ghcr.io/mazurkatarzynaumcs/pass-cracking-ex2:latest
podman run -p 4002:4002 --name ex2 ghcr.io/mazurkatarzynaumcs/pass-cracking-ex2:latest
```

(b) Wyślij request do endpointa `http://127.0.0.1:4002/hash` używając metody HTTP `GET`. Otrzymasz w odpowiedzi **hash SHA-1** do złamania (jako `hash`).

(c) Wykorzystując narzędzia `hashcat` oraz `John the Ripper` i dostępne **słowniki**, spróbuj złamać hash.

(d) Za pomocą metody HTTP `POST`, wyślij pod endpoint `http://127.0.0.1:4002/submit` wartość złamanego hasha (jako `word`).

(e) W odpowiedzi serwer zwróci odpowiednią informację o sukcesie lub błędzie - zweryfikuje poprawność złamanego hasha.

```
(.python_env) ┌─[admin@parrot]─[~/slowniki]
└──╼ $cd ex42
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex42]
└──╼ $hashcat --help
hashcat (v6.2.6) starting in help mode

Usage: hashcat [options]... hash|hashfile|hccapxfile [dictionary|mask|directory]...

- [ Options ] -

 Options Short / Long          | Type | Description
| Example
================================+======+============================================================
+========================
 -m, --hash-type               | Num  | Hash-type, references below (otherwise autodetect)
| -m 1000
 -a, --attack-mode             | Num  | Attack-mode, see references below
| -a 3
 -V, --version                 |      | Print version
|
 -h, --help                    |      | Print help
|
     --quiet                   |      | Suppress output
|
     --hex-charset             |      | Assume charset is given in hex
```

```
     # | Name                                                       | Category
 ======+============================================================+=========================
==============
   900 | MD4                                                        | Raw Hash
     0 | MD5                                                        | Raw Hash
   100 | SHA1                                                       | Raw Hash
  1300 | SHA2-224                                                   | Raw Hash
  1400 | SHA2-256                                                   | Raw Hash
 10800 | SHA2-384                                                   | Raw Hash
  1700 | SHA2-512                                                   | Raw Hash
 17300 | SHA3-224                                                   | Raw Hash
 17400 | SHA3-256                                                   | Raw Hash
 17500 | SHA3-384                                                   | Raw Hash
 17600 | SHA3-512                                                   | Raw Hash
  6000 | RIPEMD-160                                                 | Raw Hash
   600 | BLAKE2b-512                                                | Raw Hash
 11700 | GOST R 34.11-2012 (Streebog) 256-bit, big-endian           | Raw Hash
 11800 | GOST R 34.11-2012 (Streebog) 512-bit, big-endian           | Raw Hash
  6900 | GOST R 34.11-94                                            | Raw Hash
 17010 | GPG (AES-128/AES-256 (SHA-1($pass)))                       | Raw Hash
  5100 | Half MD5                                                   | Raw Hash
 17700 | Keccak-224                                                 | Raw Hash
 17800 | Keccak-256                                                 | Raw Hash
 17900 | Keccak-384                                                 | Raw Hash
 18000 | Keccak-512                                                 | Raw Hash
  6100 | Whirlpool                                                  | Raw Hash
 10100 | SipHash                                                    | Raw Hash
    70 | md5(utf16le($pass))                                        | Raw Hash
   170 | sha1(utf16le($pass))                                       | Raw Hash
  1470 | sha256(utf16le($pass))                                     | Raw Hash
 10870 | sha384(utf16le($pass))                                     | Raw Hash
  1770 | sha512(utf16le($pass))                                     | Raw Hash
   610 | BLAKE2b-512($pass.$salt)                                   | Raw Hash salted and/or
terated
   620 | BLAKE2b-512($salt.$pass)                                   | Raw Hash salted and/or
```

```
------------------+-------+-------------------------------------------------------------
  Wordlist         | $P$   | hashcat -a 0 -m 400 example400.hash example.dict
  Wordlist + Rules | MD5   | hashcat -a 0 -m 0 example0.hash example.dict -r rules/best64.rule
  Brute-Force      | MD5   | hashcat -a 3 -m 0 example0.hash ?a?a?a?a?a?a
  Combinator       | MD5   | hashcat -a 1 -m 0 example0.hash example.dict example.dict
  Association      | $1$   | hashcat -a 9 -m 500 example500.hash 1word.dict -r rules/best64.ru
le

If you still have no idea what just happened, try the following pages:

* https://hashcat.net/wiki/#howtos_videos_papers_articles_etc_in_the_wild
* https://hashcat.net/faq/

If you think you need help by a real human come to the hashcat Discord:

* https://hashcat.net/discord

(.python_env) ┌─[admin@parrot]─[~/slowniki/ex42]
└──╼ $echo -n "bf9d58891e6abdbb8bcc160c19f30f2d3ce9ad4d" > hash.txt
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex42]
└──╼ $hashcat -m 100 -a 0 hash.txt wordlist.txt
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 3.1+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 15.0.6, SLEEF,
DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
================================================================================================
================================================
* Device #1: pthread-haswell-12th Gen Intel(R) Core(TM) i5-12600K, 30984/62032 MB (8192 MB all
ocatable), 16MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
```

4.5 *Pod adresem* `http://127.0.0.1:4005` *działa prosty serwer HTTP udostępniający dwa endpointy:* `http://127.0.0.1:4005/hash` *oraz* `http://127.0.0.1:4005/submit`.

(a) Uruchom serwer za pomocą poniższego polecenia:

```
docker run -p 4005:4005 --name ex5 docker.io/mazurkatarzyna/pass-cracking-ex5:latest
podman run -p 4005:4005 --name ex5 docker.io/mazurkatarzyna/pass-cracking-ex5:latest
```

Jeśli Docker Hub nie odpowiada, użyj obrazu zapasowego:

```
docker run -p 4005:4005 --name ex5 ghcr.io/mazurkatarzynaumcs/pass-cracking-ex5:latest
podman run -p 4005:4005 --name ex5 ghcr.io/mazurkatarzynaumcs/pass-cracking-ex5:latest
```

(b) Wyślij request do endpointa `http://127.0.0.1:4005/hash` używając metody HTTP `GET`. Otrzymasz w odpowiedzi **hash bcrypt** do złamania (jako `hash`).

(c) Wykorzystując narzędzia hashcat oraz John the Ripper i dostępne **słowniki**, spróbuj złamać hash.

(d) Za pomocą metody HTTP `POST`, wyślij pod endpoint `http://127.0.0.1:4005/submit` wartość złamanego hasha (jako `word`).

(e) W odpowiedzi serwer zwróci odpowiednią informację o sukcesie lub błędzie - zweryfikuje poprawność złamanego hasha.

```
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex42]
└──╼ $cd ../ex45
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex45]
└──╼ $ls
wordlist.txt
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex45]
└──╼ $curl -X GET http://127.0.0.1:4005/hash | jq -r .hash > hash.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   133  100   133    0     0    720      0 --:--:-- --:--:-- --:--:--   722
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex45]
└──╼ $cat hash.txt
$2b$12$QI85kK7P/iBX1MjebXviyOMyPIP3YVv74pHd5Y60iCqN1mm/1hIsG
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex45]
└──╼ $
```

```
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex45]
└──╼ $hashcat --help | grep "bcrypt"
   3200 | bcrypt $2*$, Blowfish (Unix)                          | Operating System
  25600 | bcrypt(md5($pass)) / bcryptmd5                        | Forums, CMS, E-Commerce
  25800 | bcrypt(sha1($pass)) / bcryptsha1                      | Forums, CMS, E-Commerce
  28400 | bcrypt(sha512($pass)) / bcryptsha512                  | Forums, CMS, E-Commerce
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex45]
└──╼ $
```

```
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex45]
└──╼ $hashcat -m 3200 hash.txt --show
$2b$12$QI85kK7P/iBX1MjebXviyOMyPIP3YVv74pHd5Y60iCqN1mm/1hIsG:8080040393
```

```
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex45]
└──╼ $curl -X POST http://127.0.0.1:4005/submit -H "Content-Type: application/json" -d '{"word
":"8080040393"}'
{"success": true, "message": "Gratulacje! Poprawnie złamano hash!", "word": "8080040393", "has
h": "$2b$12$QI85kK7P/iBX1MjebXviyOMyPIP3YVv74pHd5Y60iCqN1mm/1hIsG"}(.python_env) ┌─[admin@parr
ot]─[~/slowniki/ex45]
```

4.7 *Pod adresem* `http://127.0.0.1:4007` *działa prosty serwer HTTP udostępniający dwa endpointy:* `http://127.0.0.1:4007/hash` *oraz* `http://127.0.0.1:4007/submit`.

(a) Uruchom serwer za pomocą poniższego polecenia:

```
docker run -p 4007:4007 --name ex7 docker.io/mazurkatarzyna/pass-cracking-ex7:latest
podman run -p 4007:4007 --name ex7 docker.io/mazurkatarzyna/pass-cracking-ex7:latest
```

Jeśli Docker Hub nie odpowiada, użyj obrazu zapasowego:

```
docker run -p 4007:4007 --name ex7 ghcr.io/mazurkatarzynaumcs/pass-cracking-ex7:latest
podman run -p 4007:4007 --name ex7 ghcr.io/mazurkatarzynaumcs/pass-cracking-ex7:latest
```

(b) Wyślij request do endpointa `http://127.0.0.1:4007/hash` używając metody HTTP `GET`. Otrzymasz w odpowiedzi **hash MD5** do złamania (jako `hash`).

(c) Wiedząc, że **hasło składa się z 3 znaków, z których każdy jest cyfrą od 0 do 9**, za pomocą narzędzia `crunch` wygeneruj słownik, którego użyjesz do złamania hasha

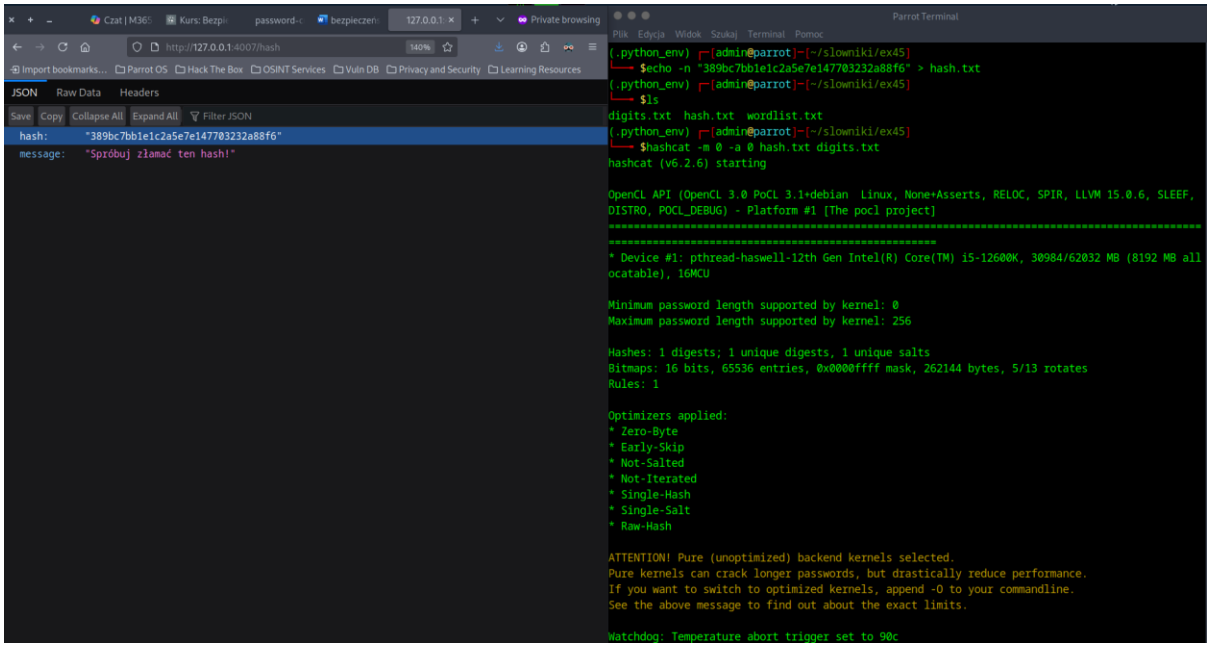
(d) Do złamania hasha użyj narzędzia `hashcat` lub `John the Ripper`.

(e) Za pomocą metody HTTP `POST`, wyślij pod endpoint `http://127.0.0.1:4007/submit` wartość złamanego hasha (jako `word`).

(f) W odpowiedzi serwer zwróci odpowiednią informację o sukcesie lub błędzie - zweryfikuje poprawność złamanego hasha.

Generowanie słownika cyferek:

```
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex45]
└──╼ $crunch
crunch version 3.6

Crunch can create a wordlist based on criteria you specify.  The output from crunch can be sen
t to the screen, file, or to another program.

Usage: crunch <min> <max> [options]
where min and max are numbers

Please refer to the man page for instructions and examples on how to use crunch.
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex45]
└──╼ $crunch 3 3 0123456789 -o digits.txt
Crunch will now generate the following amount of data: 4000 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 1000

crunch: 100% completed generating output
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex45]
└──╼ $head digits.txt
000
001
002
003
004
005
006
007
008
009
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex45]
└──╼ $
```



```
Approaching final keyspace - workload adjusted.

389bc7bb1e1c2a5e7e147703232a88f6:508

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 0 (MD5)
Hash.Target......: 389bc7bb1e1c2a5e7e147703232a88f6
Time.Started.....: Mon Nov  3 05:10:00 2025 (0 secs)
Time.Estimated...: Mon Nov  3 05:10:00 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (digits.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  3984.1 kH/s (0.03ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 1000/1000 (100.00%)
Rejected.........: 0/1000 (0.00%)
Restore.Point....: 0/1000 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: 000 -> 999
Hardware.Mon.#1..: Temp: 29c Util:  6%

Started: Mon Nov  3 05:10:00 2025
Stopped: Mon Nov  3 05:10:01 2025
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex45]
└──╼ $hashcat -m 0 hash.txt --show
389bc7bb1e1c2a5e7e147703232a88f6:508
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex45]
└──╼ $curl -X POST http://127.0.0.1:4007/submit -H "Content-Type: application/json" -d '{"word
":"508"}'
{"success": true, "message": "Gratulacje! Poprawnie złamano hash!", "word": "508", "hash": "38
9bc7bb1e1c2a5e7e147703232a88f6"}(.python_env) ┌─[admin@parrot]─[~/slowniki/ex45]
└──╼ $
```

4.12 *Pod adresem* `http://127.0.0.1:4012` *działa prosty serwer HTTP udostępniający dwa endpointy:* `http://127.0.0.1:4012/hash` *oraz* `http://127.0.0.1:4012/submit`.

(a) Uruchom serwer za pomocą poniższego polecenia:

```
docker run -p 4012:4012 --name ex12 docker.io/mazurkatarzyna/pass-cracking-ex12:latest
podman run -p 4012:4012 --name ex12 docker.io/mazurkatarzyna/pass-cracking-ex12:latest
```

Jeśli Docker Hub nie odpowiada, użyj obrazu zapasowego:

```
docker run -p 4012:4012 --name ex12 ghcr.io/mazurkatarzynaumcs/pass-cracking-ex12:latest
podman run -p 4012:4012 --name ex12 ghcr.io/mazurkatarzynaumcs/pass-cracking-ex12:latest
```

(b) Wyślij request do endpointa `http://127.0.0.1:4012/hash` używając metody HTTP `GET`. Otrzymasz w odpowiedzi **hash SHA-1** do złamania (jako `hash`).

(c) Wiedząc, że **hasło składa się z 4 znaków, gdzie każdy z nich jest małą literą od** a **do** z, złam hash.

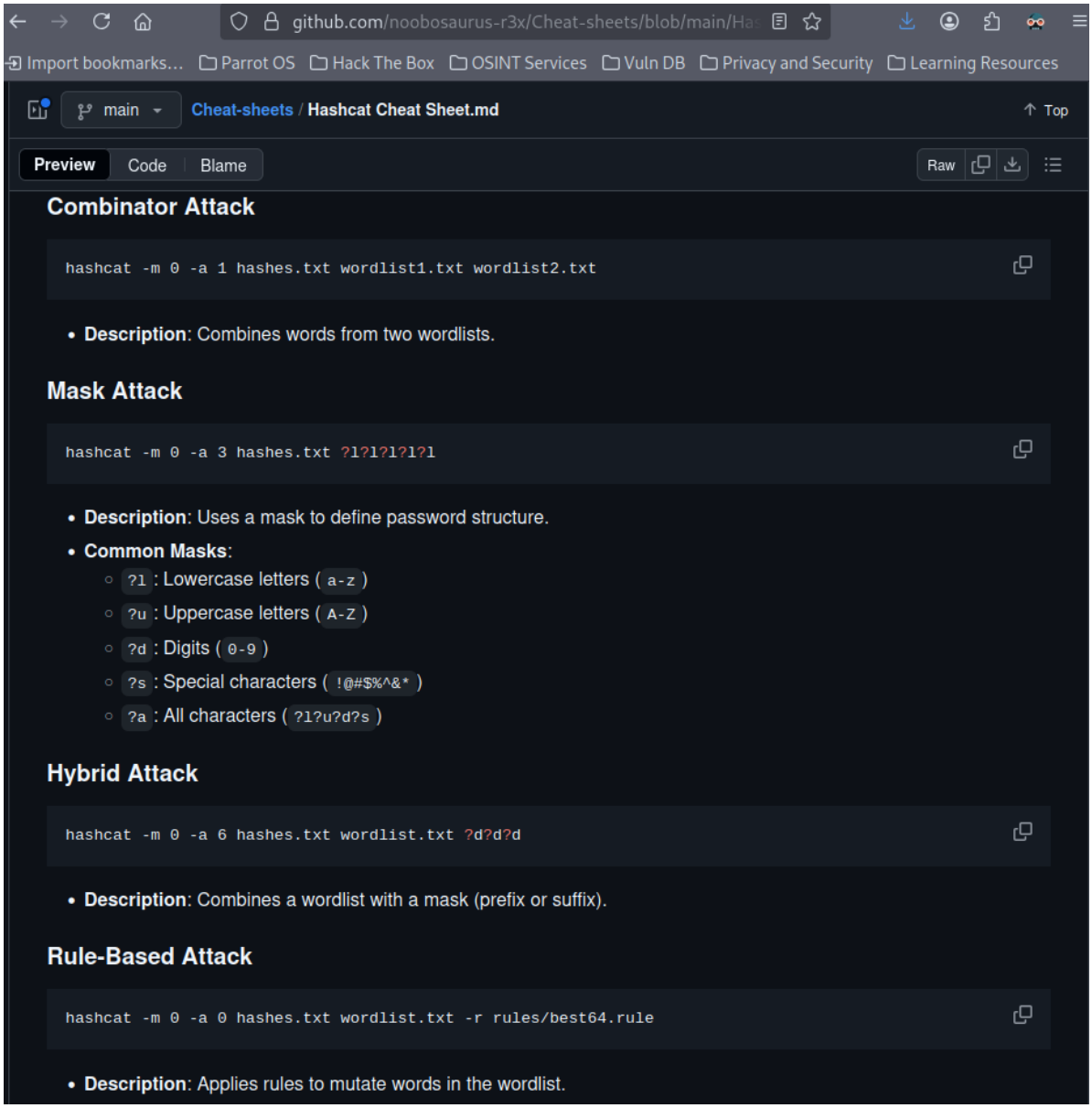
(d) Nie generuj własnych, ani nie używaj gotowych słowników. Wykorzystaj metodę bruteforce, definiując wzorzec hasła używając odpowiedniej maski. (Maski to wzorce definiujące strukturę hasła. Zamiast sprawdzać wszystkie możliwe kombinacje, możesz określić dokładny format hasła.)

(e) Do złamania hasha użyj narzędzia `hashcat` lub `John the Ripper`.

(f) Za pomocą metody HTTP `POST`, wyślij pod endpoint `http://127.0.0.1:4012/submit` wartość złamanego hasha (jako `word`).

(g) W odpowiedzi serwer zwróci odpowiednią informację o sukcesie lub błędzie - zweryfikuje poprawność złamanego hasha.


(.python_env) ┌─[✗]─[admin@parrot]─[~]
└──╼ $podman rm ex12
ex12
(.python_env) ┌─[admin@parrot]─[~]
└──╼ $podman run -p 4012:4012 --name ex12 docker.io/mazurkatarzyna/pass-cracking
-ex12:latest
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:4012
* Running on http://10.0.2.100:4012
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 145-865-179



Link w kampusie do tego

```
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex45]
└──╼ $cd ../ex46
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex46]
└──╼ $echo -n "c106ca2d7a3ab82ccc453cca716ad85cb2ad2013" > hash.txt
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex46]
└──╼ $hashcat -a 3 -m 100 hash.txt '?l?l?l?l'
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 3.1+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 15.0.6, SLEEF,
DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
================================================================================================
================================================
* Device #1: pthread-haswell-12th Gen Intel(R) Core(TM) i5-12600K, 30984/62032 MB (8192 MB all
ocatable), 16MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates

Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Hash
* Single-Salt
* Brute-Force
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.
```

```
Stopped: Mon Nov  3 05:17:38 2025
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex46]
└──╼ $hashcat -a 3 -m 100 hash.txt --show
c106ca2d7a3ab82ccc453cca716ad85cb2ad2013:kouk
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex46]
└──╼ $curl -X POST http://127.0.0.1:4012/submit -H "Content-Type: application/json" -d '{"word
":"kouk"}'
{"success": true, "message": "Gratulacje! Poprawnie złamano hash!", "word": "kouk", "hash": "c
106ca2d7a3ab82ccc453cca716ad85cb2ad2013"}(.python_env) ┌─[admin@parrot]─[~/slowniki/ex46]
└──╼ $
```

4.13 *Pod adresem* `http://127.0.0.1:4013` *działa prosty serwer HTTP udostępniający dwa endpointy:* `http://127.0.0.1:4013/hash` *oraz* `http://127.0.0.1:4013/submit`.

(a) Uruchom serwer za pomocą poniższego polecenia:

```
docker run -p 4013:4013 --name ex13 docker.io/mazurkatarzyna/pass-cracking-ex13:latest
podman run -p 4013:4013 --name ex13 docker.io/mazurkatarzyna/pass-cracking-ex13:latest
```

Jeśli Docker Hub nie odpowiada, użyj obrazu zapasowego:

```
docker run -p 4013:4013 --name ex13 ghcr.io/mazurkatarzynaumcs/pass-cracking-ex13:latest
podman run -p 4013:4013 --name ex13 ghcr.io/mazurkatarzynaumcs/pass-cracking-ex13:latest
```

(b) Wyślij request do endpointa `http://127.0.0.1:4013/hash` używając metody HTTP `GET`. Otrzymasz w odpowiedzi **hash SHA-1** do złamania (jako `hash`).

(c) Wiedząc, że **hasło składa się z 4 znaków, gdzie pierwszy z nich jest wielką literą (od `A` do `Z`), a reszta cyframi od 0 do 9**, złam hash.

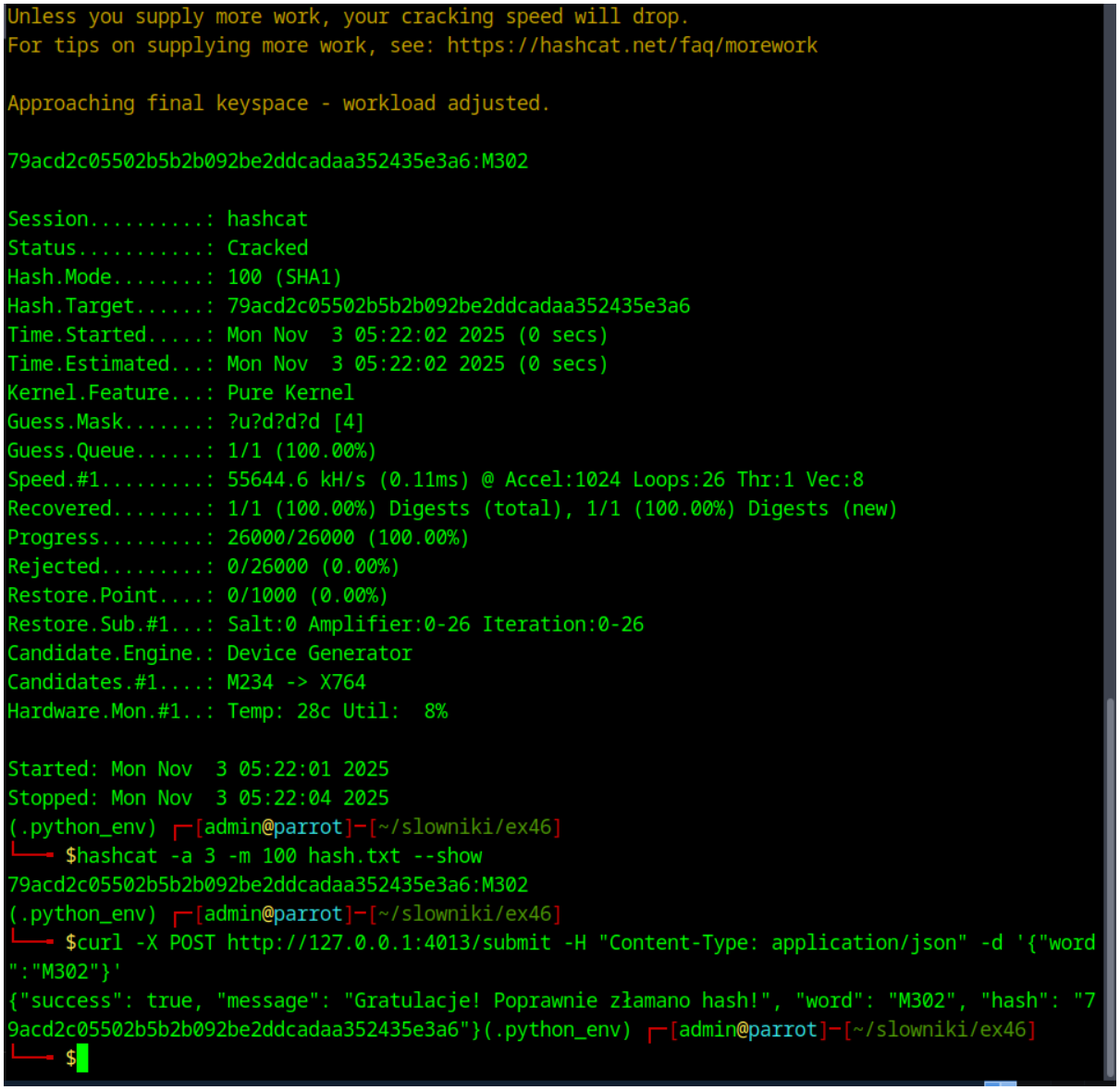
(d) Nie generuj własnych, ani nie używaj gotowych słowników. Wykorzystaj metodę bruteforce, definiując wzorzec hasła używając odpowiedniej maski. (Maski to wzorce definiujące strukturę hasła. Zamiast sprawdzać wszystkie możliwe kombinacje, możesz określić dokładny format hasła.)

(e) Do złamania hasha użyj narzędzia `hashcat` lub `John the Ripper`.

(f) Za pomocą metody HTTP `POST`, wyślij pod endpoint `http://127.0.0.1:4013/submit` wartość złamanego hasha (jako `word`).

(g) W odpowiedzi serwer zwróci odpowiednią informację o sukcesie lub błędzie - zweryfikuje poprawność złamanego hasha.

```
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex46]
└──╼ $curl -X GET http://127.0.0.1:4013/hash | jq -r .hash > hash.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   113  100   113    0     0  21312      0 --:--:-- --:--:-- --:--:-- 22600
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex46]
└──╼ $cat hash.txt
79acd2c05502b5b2b092be2ddcadaa352435e3a6
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex46]
└──╼ $hashcat -m 0 -a 3 hash.txt '?u?d?d?d'
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 3.1+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 15.0.6, SLEEF,
DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
============================================================================================
============================================
* Device #1: pthread-haswell-12th Gen Intel(R) Core(TM) i5-12600K, 30984/62032 MB (8192 MB all
ocatable), 16MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashfile 'hash.txt' on line 1 (79acd2c05502b5b2b092be2ddcadaa352435e3a6): Token length excepti
on

* Token length exception: 1/1 hashes
  This error happens if the wrong hash type is specified, if the hashes are
  malformed, or if input is otherwise not as expected (for example, if the
  --username option is used but no username is present)

No hashes loaded.

Started: Mon Nov  3 05:21:35 2025
Stopped: Mon Nov  3 05:21:35 2025
(.python_env) ┌─[✗]─[admin@parrot]─[~/slowniki/ex46]
└──╼ $hashcat -m 100 -a 3 hash.txt '?u?d?d?d'
hashcat (v6.2.6) starting
```


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.
79acd2c05502b5b2b092be2ddcadaa352435e3a6:M302
Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 100 (SHA1)
Hash.Target.....: 79acd2c05502b5b2b092be2ddcadaa352435e3a6
Time.Started.....: Mon Nov 3 05:22:02 2025 (0 secs)
Time.Estimated...: Mon Nov 3 05:22:02 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.....: ?u?d?d?d [4]
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 55644.6 kH/s (0.11ms) @ Accel:1024 Loops:26 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 26000/26000 (100.00%)
Rejected.....: 0/26000 (0.00%)
Restore.Point....: 0/1000 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-26 Iteration:0-26
Candidate.Engine.: Device Generator
Candidates.#1....: M234 -> X764
Hardware.Mon.#1..: Temp: 28c Util: 8%
Started: Mon Nov 3 05:22:01 2025
Stopped: Mon Nov 3 05:22:04 2025
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex46]
└─$ $hashcat -a 3 -m 100 hash.txt --show
79acd2c05502b5b2b092be2ddcadaa352435e3a6:M302
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex46]
└─$ $curl -X POST http://127.0.0.1:4013/submit -H "Content-Type: application/json" -d '{"word
":"M302"}'
{"success": true, "message": "Gratulacje! Poprawnie złamano hash!", "word": "M302", "hash": "7
9acd2c05502b5b2b092be2ddcadaa352435e3a6"}(.python_env) ┌─[admin@parrot]─[~/slowniki/ex46]
└─$ $

4.14 *Pod adresem* `http://127.0.0.1:4014` *działa prosty serwer HTTP udostępniający dwa endpointy:* `http://127.0.0.1:4014/hash` *oraz* `http://127.0.0.1:4014/submit`.

(a) Uruchom serwer za pomocą poniższego polecenia:

```
docker run -p 4014:4014 --name ex14 docker.io/mazurkatarzyna/pass-cracking-ex14:latest
podman run -p 4014:4014 --name ex14 docker.io/mazurkatarzyna/pass-cracking-ex14:latest
```

Jeśli Docker Hub nie odpowiada, użyj obrazu zapasowego:

```
docker run -p 4014:4014 --name ex14 ghcr.io/mazurkatarzynaumcs/pass-cracking-ex14:latest
podman run -p 4014:4014 --name ex14 ghcr.io/mazurkatarzynaumcs/pass-cracking-ex14:latest
```

(b) Wyślij request do endpointa `http://127.0.0.1:4014/hash` używając metody HTTP `GET`. Otrzymasz w odpowiedzi **hash SHA-1** do złamania (jako `hash`).

(c) Wiedząc, że **hasło składa się z 3 znaków, gdzie:**

- pierwszy znak hasła to litera ze zbioru `{a, b, c}`,
- drugi znak hasła to cyfra za zbioru `{4, 6, 8}`
- trzeci znak hasła to znak specjalny ze zbioru: `{*, %, :}`

złam hash. (Przykładowe hasła to: `a8*`, `b6%`, `c4:` ....)

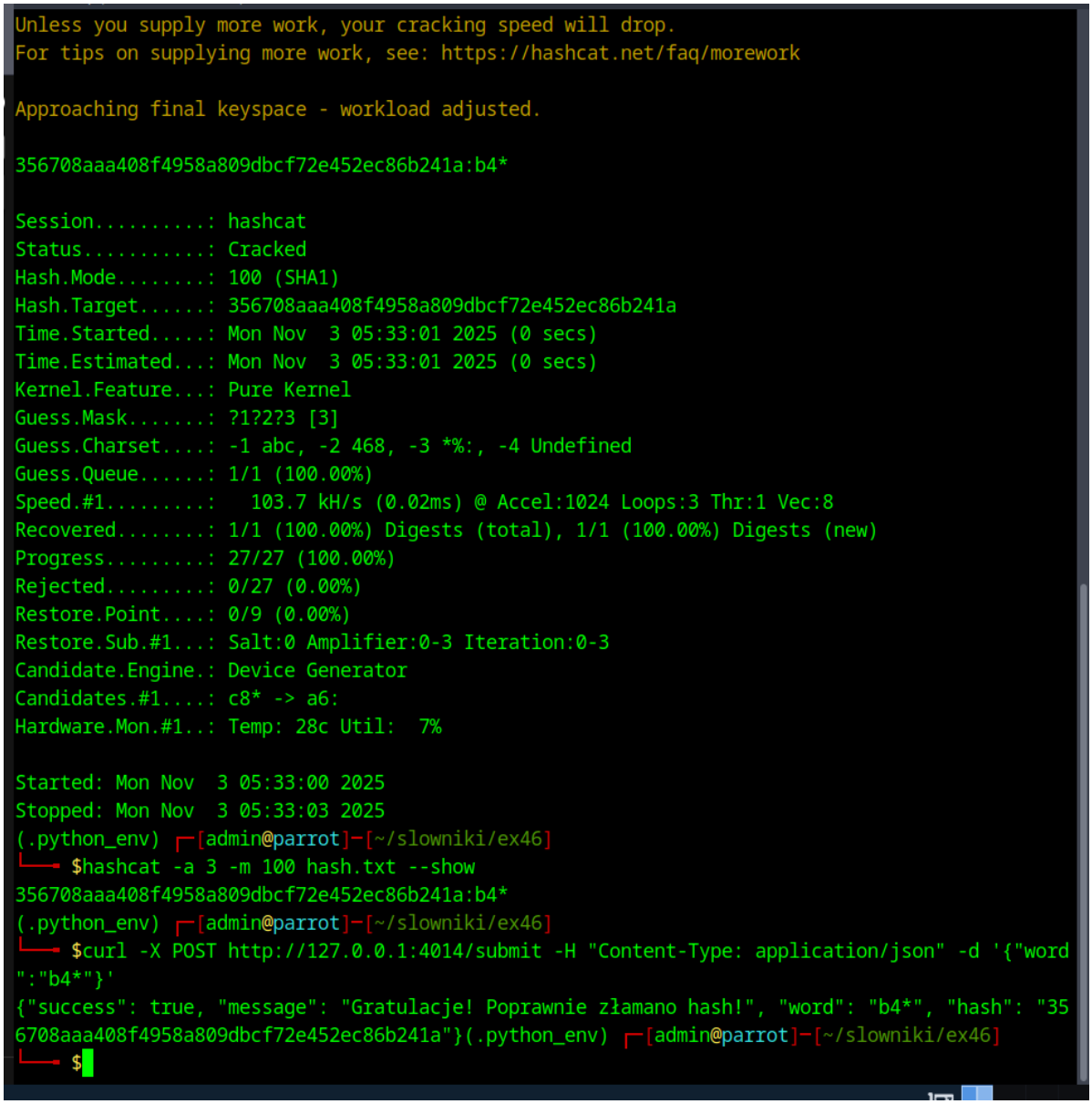
(d) Nie generuj własnych, ani nie używaj gotowych słowników. Wykorzystaj metodę bruteforce, definiując wzorzec hasła używając odpowiedniej maski. (Maski to wzorce definiujące strukturę hasła. Zamiast sprawdzać wszystkie możliwe kombinacje, możesz określić dokładny format hasła.)

(e) Do złamania hasha użyj narzędzia `hashcat` lub `John the Ripper`.

(f) Za pomocą metody HTTP `POST`, wyślij pod endpoint `http://127.0.0.1:4014/submit` wartość złamanego hasha (jako `word`).

(g) W odpowiedzi serwer zwróci odpowiednią informację o sukcesie lub błędzie - zweryfikuje poprawność złamanego hasha.

```
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex46]
└──╼ $curl -X GET http://127.0.0.1:4014/hash | jq -r .hash > hash.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   113  100   113    0     0  70846      0 --:--:-- --:--:-- --:--:--  110k
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex46]
└──╼ $hashcat -m 100 -a 3 hash.txt -1 abc -2 468 -3 '*%:' '?1?2?3'
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 3.1+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 15.0.6, SLEEF,
DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
================================================================================================
================================================
* Device #1: pthread-haswell-12th Gen Intel(R) Core(TM) i5-12600K, 30984/62032 MB (8192 MB all
ocatable), 16MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates

Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Hash
* Single-Salt
* Brute-Force
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 90c
```


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.
356708aaa408f4958a809dbcf72e452ec86b241a:b4*
Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 100 (SHA1)
Hash.Target.....: 356708aaa408f4958a809dbcf72e452ec86b241a
Time.Started.....: Mon Nov 3 05:33:01 2025 (0 secs)
Time.Estimated...: Mon Nov 3 05:33:01 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.....: ?1?2?3 [3]
Guess.Charset....: -1 abc, -2 468, -3 *%:, -4 Undefined
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 103.7 kH/s (0.02ms) @ Accel:1024 Loops:3 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 27/27 (100.00%)
Rejected.....: 0/27 (0.00%)
Restore.Point....: 0/9 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-3 Iteration:0-3
Candidate.Engine.: Device Generator
Candidates.#1....: c8* -> a6:
Hardware.Mon.#1..: Temp: 28c Util: 7%
Started: Mon Nov 3 05:33:00 2025
Stopped: Mon Nov 3 05:33:03 2025
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex46]
└─$ $hashcat -a 3 -m 100 hash.txt --show
356708aaa408f4958a809dbcf72e452ec86b241a:b4*
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex46]
└─$ $curl -X POST http://127.0.0.1:4014/submit -H "Content-Type: application/json" -d '{"word
":"b4*"}'
{"success": true, "message": "Gratulacje! Poprawnie złamano hash!", "word": "b4*", "hash": "35
6708aaa408f4958a809dbcf72e452ec86b241a"}(.python_env) ┌─[admin@parrot]─[~/slowniki/ex46]
└─$ $

Łamanie haseł - Hybrid attacks (dictionary + mask/rule)

4.16 *Pod adresem* `http://127.0.0.1:4016` *działa prosty serwer HTTP udostępniający dwa endpointy:* `http://127.0.0.1:4016/hash` *oraz* `http://127.0.0.1:4016/submit`.

(a) Uruchom serwer za pomocą poniższego polecenia:

```
docker run -p 4016:4016 --name ex16 docker.io/mazurkatarzyna/pass-cracking-ex16:latest
podman run -p 4016:4016 --name ex16 docker.io/mazurkatarzyna/pass-cracking-ex16:latest
```

Jeśli Docker Hub nie odpowiada, użyj obrazu zapasowego:

```
docker run -p 4016:4016 --name ex16 ghcr.io/mazurkatarzynaumcs/pass-cracking-ex16:latest
podman run -p 4016:4016 --name ex16 ghcr.io/mazurkatarzynaumcs/pass-cracking-ex16:latest
```

(b) Wyślij request do endpointa `http://127.0.0.1:4016/hash` używając metody HTTP `GET`. Otrzymasz w odpowiedzi **hash SHA-256** do złamania (jako `hash`).

(c) Wiedząc, że **hasło jest słowem ze słownika zawierającego najpopularniejsze hasła, do którego dodano na końcu dwie cyfry (każda od 0 do 9)**, złam hash. (Hasło to jedno z 10 pierwszych haseł **ze słownika zawierającego najpopularniejsze hasła**) z dodanymi 2 cyframi na końcu, czyli np. `shadow39` czy `sunshine00`.)

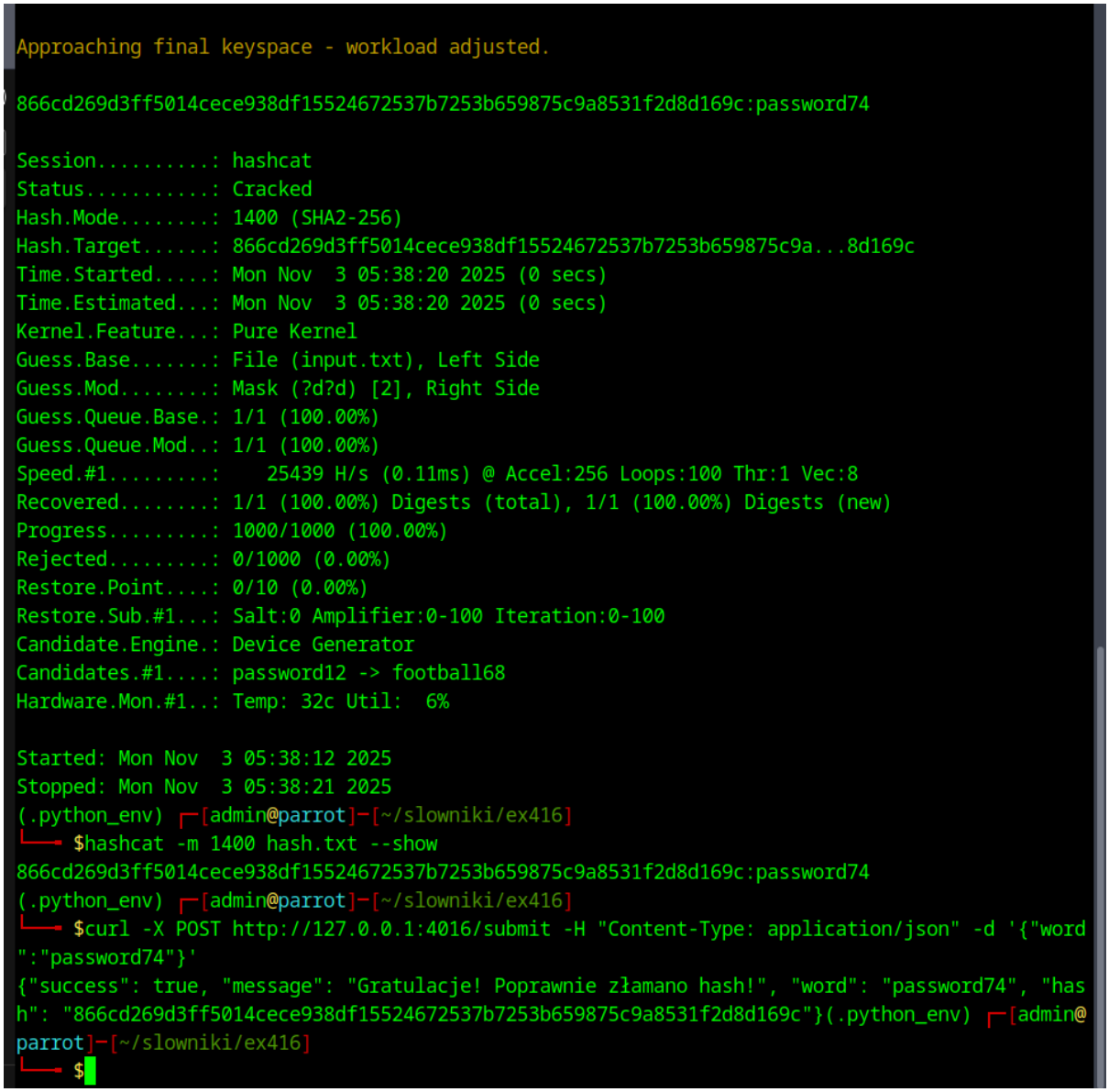
(d) W celu złamania hasha, wykorzystaj **słownik popularnych haseł**, oraz zastosuj odpowiednią maskę, dodającą do każdego słowa ze słownika 2 cyfry na końcu słowa.

(e) Do złamania hasha użyj narzędzia `hashcat` lub `John the Ripper`.

(f) Za pomocą metody HTTP `POST`, wyślij pod endpoint `http://127.0.0.1:4016/submit` wartość złamanego hasha (jako `word`).

(g) W odpowiedzi serwer zwróci odpowiednią informację o sukcesie lub błędzie - zweryfikuje poprawność złamanego hasha.

```
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex416]
└──╼ $curl -X GET http://127.0.0.1:4016/hash | jq -r .hash > hash.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   137  100   137    0     0  89250      0 --:--:-- --:--:-- --:--:--  133k
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex416]
└──╼ $cat hash.txt
866cd269d3ff5014cece938df15524672537b7253b659875c9a8531f2d8d169c
(.python_env) ┌─[admin@parrot]─[~/slowniki/ex416]
└──╼ $hashcat -a 6 -m 1400 hash.txt input.txt '?d?d'
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 3.1+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 15.0.6, SLEEF,
DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
====================================================================================================
================================================
* Device #1: pthread-haswell-12th Gen Intel(R) Core(TM) i5-12600K, 30984/62032 MB (8192 MB all
ocatable), 16MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates

Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Hash
* Single-Salt
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.
```



Jakby było najpierw plik potem maska to ‘-a 7’

4.17 *Pod adresem* `http://127.0.0.1:4017` *działa prosty serwer HTTP udostępniający dwa endpointy:* `http://127.0.0.1:4017/hash` *oraz* `http://127.0.0.1:4017/submit`.

(a) Uruchom serwer za pomocą poniższego polecenia:

```
docker run -p 4017:4017 --name ex17 docker.io/mazurkatarzyna/pass-cracking-ex17:latest
podman run -p 4017:4017 --name ex17 docker.io/mazurkatarzyna/pass-cracking-ex17:latest
```

Jeśli Docker Hub nie odpowiada, użyj obrazu zapasowego:

```
docker run -p 4017:4017 --name ex17 ghcr.io/mazurkatarzynaumcs/pass-cracking-ex17:latest
podman run -p 4017:4017 --name ex17 ghcr.io/mazurkatarzynaumcs/pass-cracking-ex17:latest
```

(b) Wyślij request do endpointa `http://127.0.0.1:4017/hash` używając metody HTTP `GET`. Otrzymasz w odpowiedzi **hash SHA-256** do złamania (jako `hash`).

(c) Wiedząc, że **hasło jest słowem ze słownika zawierającego najpopularniejsze hasła, do którego dodano na początku małą literę (każda od a do z), cyfrę (od 0 do 9), oraz znak specjalny ze zbioru znaków !, @, #, $, %, &, ***, złam hash. (Hasło to jedno z 10 pierwszych haseł **ze słownika zawierającego najpopularniejsze hasła**) z dodanymi 3 znakami na początku słowa, czyli np. `f4@shadow` czy `k9!sunshine`.)

(d) W celu złamania hasha, wykorzystaj **słownik popularnych haseł**, oraz zastosuj odpowiednią maskę, dodającą do każdego słowa ze słownika 2 cyfry na końcu słowa.

(e) Do złamania hasha użyj narzędzia `hashcat` lub `John the Ripper`.

(f) Za pomocą metody HTTP `POST`, wyślij pod endpoint `http://127.0.0.1:4017/submit` wartość złamanego hasha (jako `word`).

(g) W odpowiedzi serwer zwróci odpowiednią informację o sukcesie lub błędzie - zweryfikuje poprawność złamanego hasha.