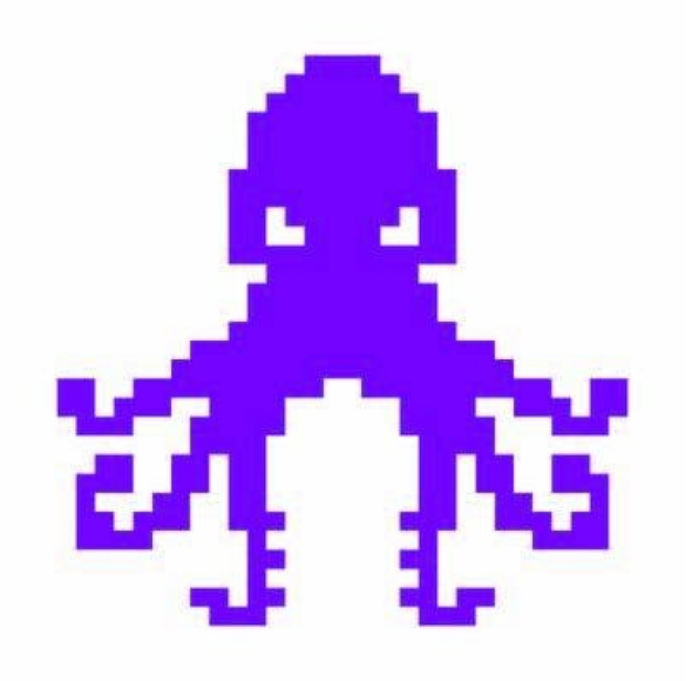
Crack the hash level 2



01/07/2022

Hash identification

We need to triying to identify which kind of hash it is.

Haiti is a CLI tool to identify the hash type og a given hash

1. 48bb6e862e54f2a795ffc4e541caed4d

haiti 741ebf5166b9ece4cca88a3868c44871e8370707cf19af3ceaa4a6fba006f224ae03f39153492853
RIPEMD-320 [JtR: dynamic_150]
Umbraco HMAC-SHA1 [HC: 24800]

Format:RIPEMD-320

2.1aec7a56aa08b25b596057e1ccbcb6d768b770eaa0f355ccbd56aee5040e02ee

haiti 1aec7a56aa08b25b596057e1ccbcb6d768b770eaa0f355ccbd56aee5040e02ee
SHA-256 [HC: 1400] [JtR: raw-sha256]
GOST R 34.11-94 [HC: 6900] [JtR: gost]
SHA3-256 [HC: 17400] [JtR: dynamic_380]
Keccak-256 [HC: 17800] [JtR: raw-keccak-256]
Snefru-256 [JtR: snefru-256]
RIPEMD-256 [JtR: dynamic_140]
Haval-256 (3 rounds) [JtR: haval-256-3]
Haval-256 (4 rounds) [JtR: dynamic_290]
Haval-256 (5 rounds) [JtR: dynamic_300]
GOST CryptoPro S-Box
Skein-256 [JtR: skein-256]
Skein-512(256)
PANAMA [JtR: dynamic_320]
BLAKE2-256
MD6-256
Umbraco HMAC-SHA1 [HC: 24800]

Format: various

3. What is Keccak-256 Hashcat code?
17800

4. What is Keccak-256 John the Ripper code?
Raw-Keccak-256

Worldlist

For hash cracking you will often need some custom or specialized dictionaries called wordlists.

SecLists is a collection of multiple types of lists used during security assessments, collected in one place. List types include usernames, passwords, URLs, sensitive data patterns, fuzzing payloads, web shells, and many more.

Wordlistctl is a script to fetch, install, update and search wordlist archives from websites offering wordlists with more than 6300 wordlists available.

Rawsec's CyberSecurity Inventory is an inventory of tools and resources about CyberSecurity. The Cracking category will be especially useful to find wordlist generator tools.

Note: On the exercise below we will see how to use wordlistctl to download a list, for the example I took rockyou which is a famous wordlist but if you use TryHackMe AttackBox or Kali Linux you should already have it under /usr/share/wordlists/, so you don't need to download it again, this is just an example to show you how wordlistctl works.

RockYou is a famous wordlist contains a large set of commonly used password sorted by frequency.

5. To search for this wordlist with wordlistclt run:

python3 ./wordlistctl.py search rockyou

to download any list and decompressed

./wordlistctl.py fetch -l "dogs" -d fetch_term

6. Which option do you need to add to the previous command to search into local archives instead of remote ones?

python3 ./wordlistctl.py search -l

7. Search the rockyou list

python3 wordlistctl.py search -l rockyou

--==[wordlistctl by blackarch.org]==--

> /usr/share/wordlists/rockyou.txt (139.92 Mb)

8. You can search for a wordlist about a specific subject (eg. facebook)wordlistctl search facebook**or list all wordlists from a category (eg. fuzzing)**wordlistctl list -g fuzzing**.**

python3 wordlistctl.py search facebook

--==[wordlistctl by blackarch.org]==--

0 > facebook-app (1.76 Mb)

1 > facebook-bot (1.64 Kb)

2 > facebook-first (40.72 Mb)

3 > facebook-firstnames (36.58 Mb)

4 > facebook-last (51.59 Mb)

5 > facebook-lastnames (46.46 Mb)

6 > facebook-names-unique (1.5 Gb)

7 > facebook-lastfirst (98.54 Mb)

8 > facebook-firstlast (171.41 Mb)

9 > facebook-f (154.93 Mb)

10 > facebook-phished (25.09 Kb)

11 > facebook-pastebay (500 B)

python3 wordlistctl.py list -g fuzzing

--==[wordlistctl by blackarch.org]==--

[+] available wordlists:

0 > php (3.48 Kb)

1 > frontpage (233.00 B)

2 > 1-4_all_letters_a-z (2.36 Mb)

3 > 3-digits-000-999 (4.00 Kb)

4 > 4-digits-0000-9999 (50.00 Kb)

5 > 5-digits-00000-99999 (600.00 Kb)

6 > 6-digits-000000-999999 (7.00 Mb)

7 > MSSQL-Enumeration (716.00 B)

8 > MSSQL (1.06 Kb)

9 > MySQL-Read-Local-Files (210.00 B)

10 > MySQL-SQLi-Login-Bypass (374.00 B)

11 > MySQL (108.00 B)

12 > NoSQL (566.00 B)

9. What is the name of the first wordlist in the usernames category?

python3 wordlistctl.py list -g usernames

--==[wordlistctl by blackarch.org]==--

[+] available wordlists:

0 > CommonAdminBase64 (1.05 Kb)

1 > multiplesources-users-fabian-fingerle (164.59 Kb)

2 > familynames-usa-top1000 (7.12 Kb)

3 > femalenames-usa-top1000 (6.94 Kb)

4 > malenames-usa-top1000 (6.68 Kb)

CommonAdminBase64

Cracking tools, modes & rules

Finally you'll need a cracking tool, the 2 very common ones are:

Hashcat

John the Ripper (jumbo version)

There are several modes of cracking you can use:

- **Wordlist mode, which consist in trying all words contained in a dictionary. For example, a list of common passwords, a list of usernames, etc.**
- **Incremental mode, which consist in trying all possible character combinations as passwords. This is powerful but much more longer especially if the password is long.**
- **Rule mode, which consist in using the wordlist mode by adding it some pattern or mangle the string. For example adding the current year, or appending a common special character.**

There are 2 ways of performing a rule based bruteforce:

1) Generating a custom wordlist and using the classic wordlist mode with it.

2) Using a common wordlist and tell the cracking tool to apply some custom mangling rules on it.

The second option is much more powerful as you wont waste gigabytes by storing tons of wordlists and waste time generating ones you will use only one time. Rather having a few interesting lists and apply various mangling rules that you can re-use over different wordlist.

John the Ripper already include various mangling rules but you can create your owns and apply them the wordlist when cracking:

```
john hash.txt --wordlist=/usr/share/wordlists/passwords/rockyou.txt rules=norajCommon02
```

You can consult John the Ripper Wordlist rules syntax for creating your own rules.

I'll give you the main ideas of mutation rules, of course several can be combined together.

Border mutation - commonly used combinations of digits and special symbols can be added at the end or at the beginning, or both

Freak mutation - letters are replaced with similarly looking special symbols

Case mutation - the program checks all variations of uppercase/lowercase letters for any character

Order mutation - character order is reversed

Repetition mutation - the same group of characters are repeated several times

Vowels mutation - vowels are omitted or capitalized

Strip mutation - one or several characters are removed

Swap mutation - some characters are swapped and change places

Duplicate mutation - some characters are duplicated

Delimiter mutation - delimiters are added between characters

Depending of your distribution, the John configuration may be located at /etc/john/john.conf **and/or** /usr/share/john/john.conf**. To locate the JtR install directory run locate john.conf, then create john-local.conf in the same directory (in my case**/usr/share/john/john-local.conf**) and create our rules in here.**

Let's use the top 10 000 most used password list from SecLists(/usr/share/seclists/Passwords/Common-Credentials/10k-most-common.txt**) and generate a simple border mutation by appending all 2 digits combinations at the end of each password. Let's edit** sudo nano /etc/john/john.conf **and add a new rule:**

```
sudo nano /etc/john/john.conf
```

```
[List.Rules:THM01]
```

```
$[0-9]$[0-9]
```

10. Now let's crack the SHA1 hash 2d5c517a4f7a14dcb38329d228a7d18a3b78ce83, we just have to write the hash in a text file and to specify the hash type, the wordlist and our rule name.

```
john --format=raw-sha1 --wordlist=/usr/share/seclists/Passwords/Common-Credentials/10k-most-common.txt --rules=THM01 hash.txt
```

Using default input encoding: UTF-8

Loaded 1 password hash (Raw-SHA1 [SHA1 256/256 AVX2 8x])

Warning: no OpenMP support for this hash type, consider --fork=8

Press 'q' or Ctrl-C to abort, almost any other key for status

```
moonligh56       (?)
```

Custom wordlist generation

As I said in the previous task mangling rules avoid to waste storage space and time but there are some cases where generating a custom wordlist could be a better idea:

- You will often re-use the wordlist, generating one will save computation power rather than using a mangling rule
- You want to use the wordlist with several tools
- You want to use a tool that support wordlists but not mangling rules
- You find the custom rule syntax of John too complex

Let's say we know the password we want to crack is about dogs. We can download a list of dog racessudo ./wordlistctl.py fetch -l "dogs" -d fetch_term (/usr/share/wordlists/misc/dogs.txt). Then we can use Mentalist to generate some mutations.

Download the list and decompressed

./wordlistctl.py fetch -l "dogs" -d fetch_term

search the path of the wordlist

./wordlistctl.py search -l dogs

--==[wordlistctl by blackarch.org]==--

> /usr/share/wordlists/misc/dogs.txt (2.41 Kb)

We can load our dog wordlist in Mentalist, add some Case, Substitution, Append/Prepend rules.

Here we will toggle the case of one char of two and replace all s with a dollar sign.

Then we can process and save the newly generated wordlist.

We save has full Wordlist

It's also possible to export John/Hashcat rules.

11. Crack the following md5 hash with the wordlist generated in the previous steps.

ed91365105bba79fdab20c376d83d752

john --format=raw-md5 –wordlist=password_hash_11.txt hash_11

john --show --format=Raw-MD5 hash_11

?:mOlo$$u$

12. Now let's use CeWL to generate a wordlist from a website. It could be useful to retrieve a lot of words related to the password's topic.

For example to download all words from example.org with a depth of 2, run:

cewl -d 2 -w $(pwd)/example.txt https://example.org

The depth is the number of link level the spider will follow.

What is the last word of the list?

Information

13. With TTPassGen **we can craft wordlists from scratch.**

Create a first wordlist containing all 4 digits PIN code value.

python3 /home/solo/.local/lib/python3.10/site-packages/ttpassgen/ttpassgen.py --rule '[?d]{4:4:*}' pin.txt

mode: combination rule mode, global_repeat_mode: ?, part_size: 0 Bytes, dictlist: [], input dict file encoding: None
raw rule string: [?d]{4:4:*}, analyzed rules: ['[?d]{4:4:*}']
estimated display size: 50.0 KB, generate dict...
100%|[redacted]| 10000/10000 [00:00<00:00, 98782.24 word/s]
generate dict complete.

Generate a list of all lowercase chars combinations of length 1 to 3.

```
python3 /home/solo/.local/lib/python3.10/site-packages/ttpassgen/ttpassgen.py --rule '[?l]{1:3:*}' abc.txt
```

mode: combination rule mode, global_repeat_mode: ?, part_size: 0 Bytes, dictlist: [], input dict file encoding: None

raw rule string: [?l]{1:3:*}, analyzed rules: ['[?l]{1:3:*}']

estimated display size: 74.54 KB, generate dict...

100%| | 18278/18278 [00:00<00:00, 177419.89 word/s]

generate dict complete.

Then we can create a new wordlist that is a combination of several wordlists. Eg. combine the PIN wordlist and the letter wordlist separated by a dash.

```
python3 /home/solo/.local/lib/python3.10/site-packages/ttpassgen/ttpassgen.py --dictlist 'pin.txt,abc.txt' --rule '$0[-]{1}$1' combination.txt
```

mode: combination rule mode, global_repeat_mode: ?, part_size: 0 Bytes, dictlist: ['pin.txt', 'abc.txt'], input dict file encoding: None

raw rule string: $0[-]{1}$1, analyzed rules: ['$0', '[-]{1}', '$1']

estimated display size: 1.64 GB, generate dict...

100%| | 18278/18278 [00:00<00:00, 177419.89 word/s]

generate dict complete.

Be warned combining wordlists quickly generated huge files, here combination.txt is 1.64 GB.

```
ls -al combination.txt abc.txt pin.txt
```

.rw-r--r-- solo solo 71 KB Tue Jun 21 23:06:35 2022 ▯ abc.txt

.rw-r--r-- solo solo 1.5 GB Thu Jun 23 17:27:01 2022 ▯ combination.txt

.rw-r--r-- solo solo 49 KB Tue Jun 21 23:05:34 2022 ▯ pin.txt

14.Crack this md5 hash with combination.txt.

e5b47b7e8df2597077e703c76ee86aee

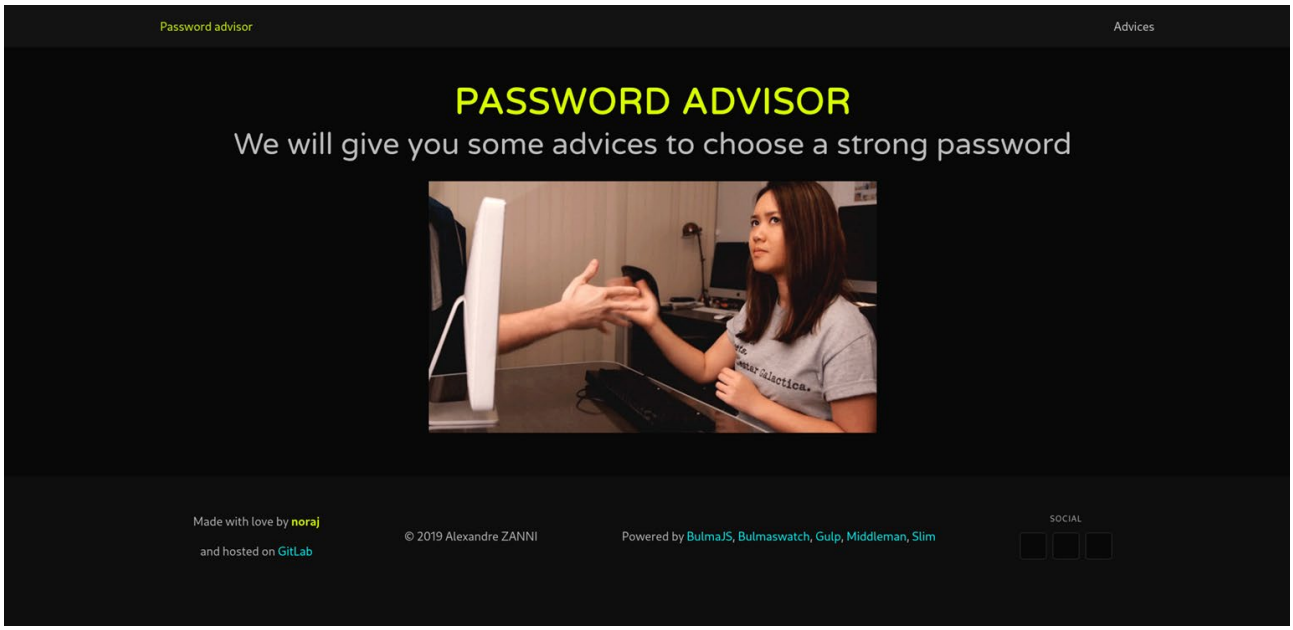
john --format=RAW-MD5 --wordlist=combination.txt hash_14

?:1551-li

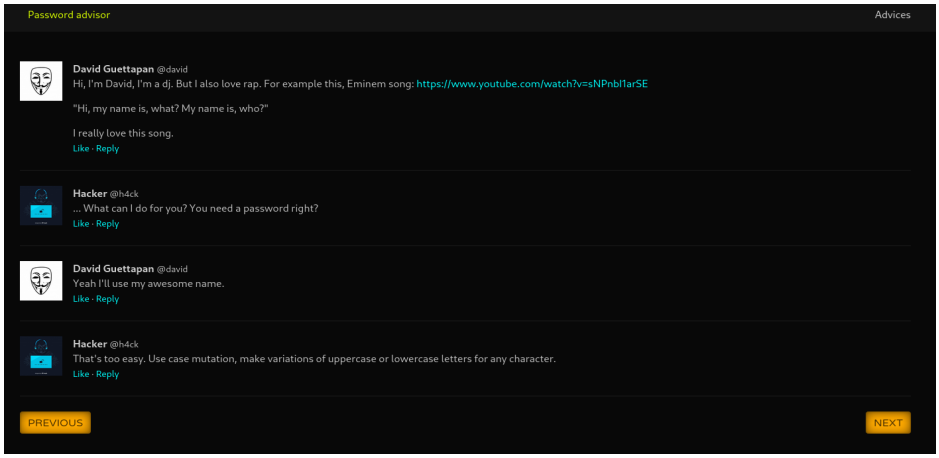
It's time to crack hashes

You will have to crack several hashes. For each hash you will be given a short scenario that will help you to create a mangling rules, build a wordlist or finding some specialized data you'll need to crack the hash.

The scenarios are located on the website: Password advisor (http://MACHINE_IP), each piece of advice matches one of the following hashes (in the same order).



Advice n°1 b16f211a8ad7f97778e5006c7cecdf31



haiti b16f211a8ad7f97778e5006c7cecdf31

SHA-1 [HC: 100] [JtR: raw-sha1]

echo b16f211a8ad7f97778e5006c7cecdf31 > advice_1

/home/solo/Desktop/wordlistctl/wordlistctl.py search "malename"

--==[wordlistctl by blackarch.org]==--

0 > femalenames-usa-top1000 (6.94 Kb)

1 > malenames-usa-top1000 (6.68 Kb)

2 > top_1000_usa_femalenames_english (6.78 Kb)

3 > top_1000_usa_malenames_english (6.52 Kb)

sudo /home/solo/Desktop/wordlistctl/wordlistctl.py fetch -l "malenames-usa-top1000" -d fetch_term

sudo /home/solo/Desktop/wordlistctl/wordlistctl.py search -l malenames-usa-top1000

--==[wordlistctl by blackarch.org]==--

> /usr/share/wordlists/usernames/malenames-usa-top1000.txt (6.68 Kb)

sudo nano /etc/john/john.conf

[List.Rules:advice1]

c$1$2$3$4$[$%&*-_+=#@~!]

john --format=RAW-MD5 --wordlist=malenames-usa-top1000.txt --rules=advice1 advice_1

john --show --format=Raw-MD5 advice_1

Zachariah1234*

another option is create a wordlist with ttpassgen

https://pypi.org/project/ttpassgen/

https://github.com/tp7309/TTPassGen

[?l] = abcdefghijklmnopqrstuvwxyz

[?s] = !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~

{n} = number of time that letters or symbols appears

create a list which contains number and symbols, the number will appear in a sequence of 5

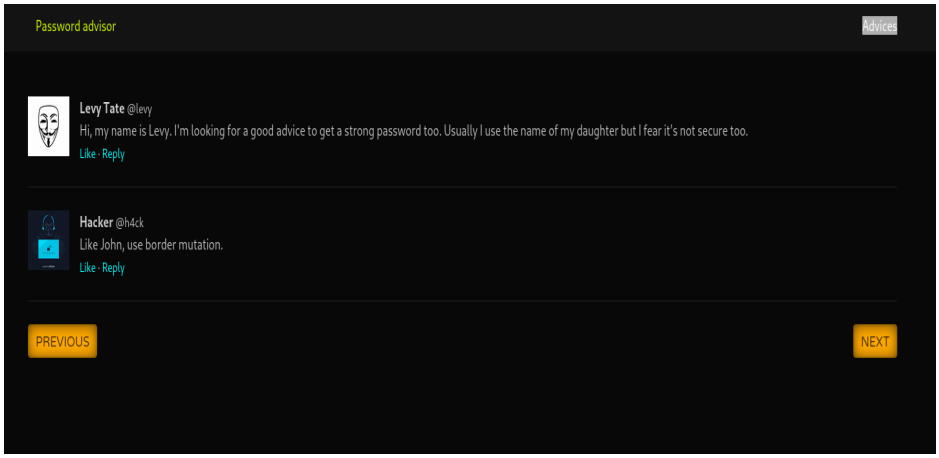
```
python3 /home/solo/.local/lib/python3.10/site-packages/ttpassgen/ttpassgen.py -r [?l]{2}[12345]{4}[?s] symbols.txt
```

then, we are going to combination malenames-usa-top1000.txt and symbols.txt

$0$1 = combination of the two files

```
python3 /home/solo/.local/lib/python3.10/site-packages/ttpassgen/ttpassgen.py --dictlist 'malenames-usa-top1000.txt,symbols.txt' --rule '$0$1' combination.txt
```

Advice n°2 7463fcb720de92803d179e7f83070f97



haiti 7463fcb720de92803d179e7f83070f97

[HC: 0] [JtR: raw-md5]

echo 7463fcb720de92803d179e7f83070f97 > advice_2

/home/solo/Desktop/wordlistctl/wordlistctl.py search "femalename"

--==[wordlistctl by blackarch.org]==--

0 > femalenames-usa-top1000 (6.94 Kb)

1 > top_1000_usa_femalenames_english (6.78 Kb)

sudo /home/solo/Desktop/wordlistctl/wordlistctl.py fetch -l "femalenames-usa-top1000" -d fetch_term

sudo /home/solo/Desktop/wordlistctl/wordlistctl.py search -l "femalenames-usa-top1000"

--==[wordlistctl by blackarch.org]==--

> /usr/share/wordlists/usernames/femalenames-usa-top1000.txt (6.94 Kb)

sudo nano /etc/john/john.conf

[List.Rules:advice2]

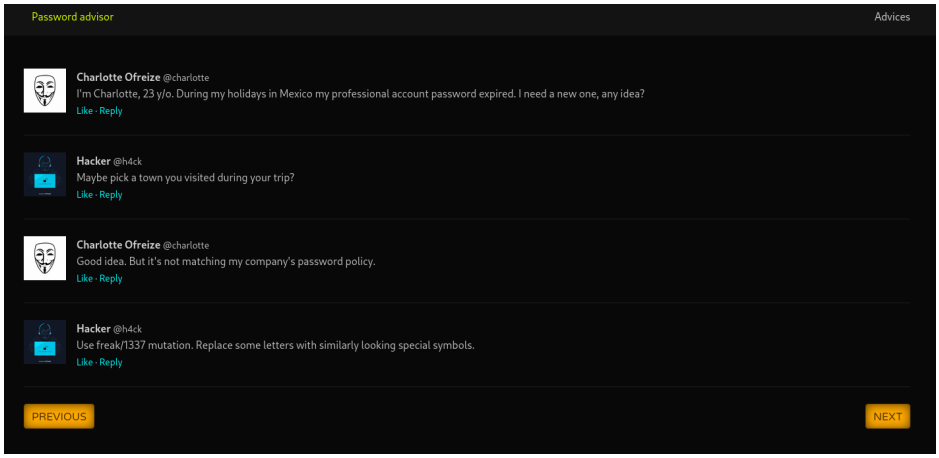
c$[0-9]$[0-9]$[$%&*-_+=#@~!]

john --format=RAW-MD5 --wordlist=femalenames-usa-top1000.txt --rules=advice2 advice_2

john --show --format=Raw-MD5 advice_2

Angelita35!

Advice n°3 f4476669333651be5b37ec6d81ef526f



haiti f4476669333651be5b37ec6d81ef526f

[HC: 0] [JtR: raw-md5]

echo 7463fcb720de92803d179e7f83070f97 > advice_3

/home/solo/Desktop/wordlistctl/wordlistctl.py search "city"

--==[wordlistctl by blackarch.org]==--

0 > htc-velocity (750.00 B)

1 > city-state-country (3.14 Mb)

sudo /home/solo/Desktop/wordlistctl/wordlistctl.py fetch -l "city-state-country.txt" -d fetch_term

sudo /home/solo/Desktop/wordlistctl/wordlistctl.py search -l "city-state-country.txt"

--==[wordlistctl by blackarch.org]==--

> /usr/share/wordlists/misc/city-state-country.txt (3.04 Mb)

Extract the name of Mexico from the list

cat city-state-country.txt | dos2unix | rg 'Mexico$' | cut -f 1 -d ',' | uniq > mexico.txt

mentalist

search freak/1337 mutatio on john.conf

sudo nano /etc/john/john.conf

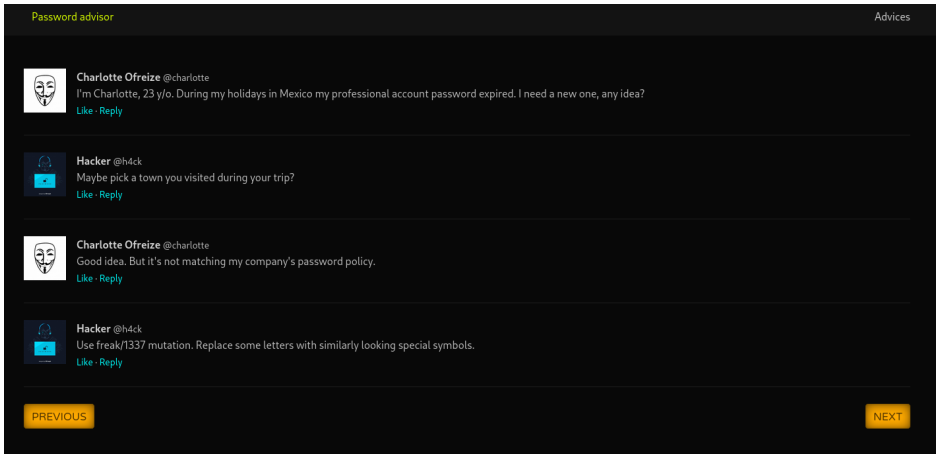
The following are some 3l33t rules

john --format=RAW-MD5 --wordlist=mentalist.txt advice_3 --rules=l33t

john --show --format=Raw-MD5 advice_3

Tl@xc@l@ncing0

Advice n°4 a3a321e1c246c773177363200a6c0466a5030afc



haiti a3a321e1c246c773177363200a6c0466a5030afc

[HC: 0] [JtR: raw-md5]

echo a3a321e1c246c773177363200a6c0466a5030afc > advice_4

search case mutation on john.conf

sudo nano /etc/john/john.conf

The following are some 3l33t rules

sudo john --format=raw-sha1 --wordlist=name.txt advice_4 –rules=Jumbo

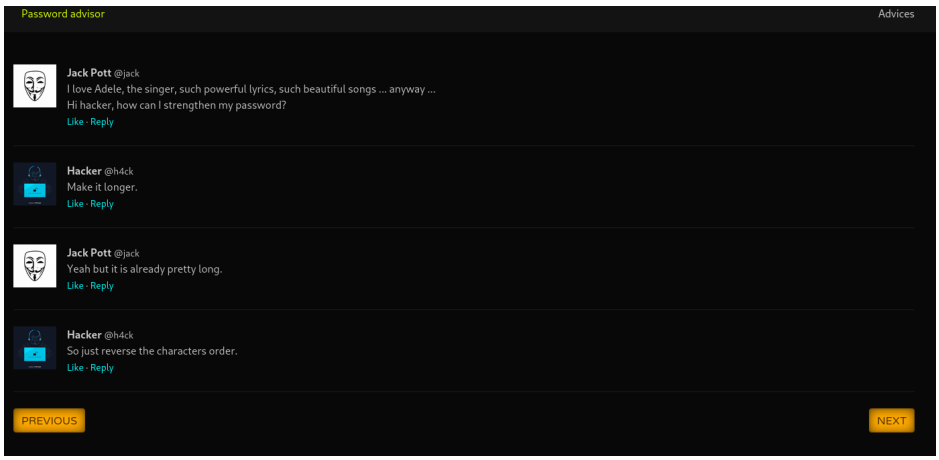
or

john --format=raw-sha1 --wordlist=name.txt advice_4 --rules=NT

john --show --format=Raw-SHA1 advice_4

DavIDgUEtTApAn

Advice n°5 d5e085772469d544a447bc8250890949



haiti d5e085772469d544a447bc8250890949

MD5 [HC: 0] [JtR: raw-md5]

echo d5e085772469d544a447bc8250890949 > advice_5

clone the repo

git clone https://github.com/initstring/lyricpass.git

search for lyric from adele

./lyricpass.py -a adele

modified the john.cong and add r for recursive

nano /etc/john/john.conf

[List.Rules:r]

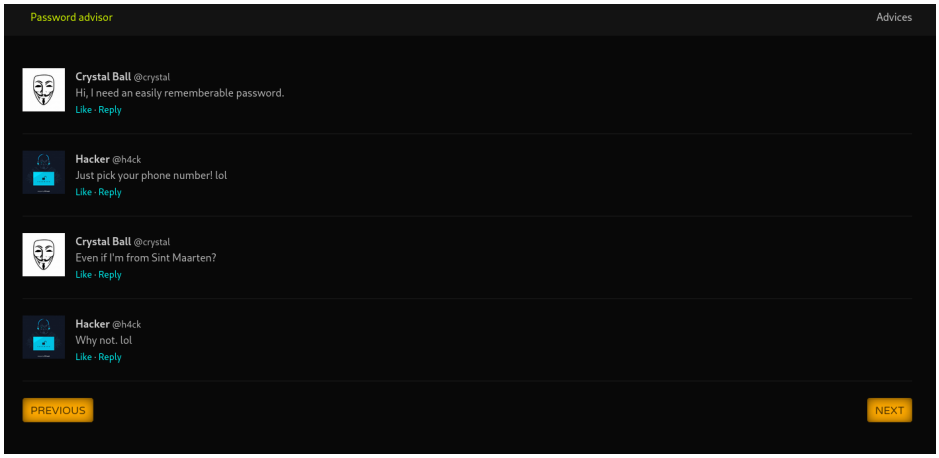
r

john --format=raw-md5 --wordlist=adele.txt --rules=r advice_5

john --show --format=raw-md5 advice_5

uoy ot miws ot em rof peed oot ro ediw oot si revir oN

Advice n°6 377081d69d23759c5946a95d1b757adc



haiti 377081d69d23759c5946a95d1b757adc

MD5 [HC: 0] [JtR: raw-md5]

echo 377081d69d23759c5946a95d1b757adc > advice_6

clone the repo

git clone https://github.com/toxydose/pnwgen.git

execute pnwgen

python3 pnwgen.py [prefix] [suffix] [length]

./pnwgen.py +1721 "" 7

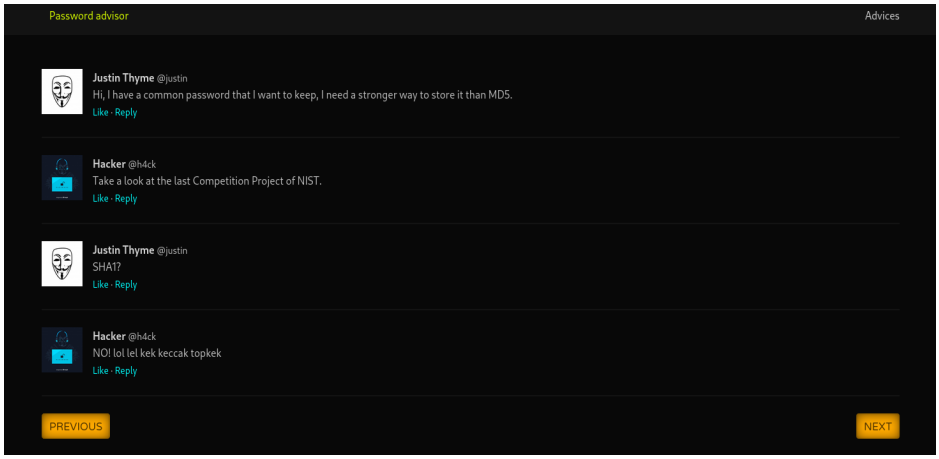
john --format=raw-md5 --wordlist=wordlist.txt advice_6

john --show --format=raw-md5 advice_6

+17215440375

Advice n°7

ba6e8f9cd4140ac8b8d2bf96c9acd2fb58c0827d556b78e331d1113fcbfe425ca9299fe917f6015978f7e1644382d1ea45fd581aed6298acde2fa01e7d83cdbd



haiti ba6e8f9cd4140ac8b8d2bf96c9acd2fb58c0827d556b78e331d1113fcbfe425ca9299fe917f6015978f7e1644382d1ea45fd581aed6298acde2fa01e7d83cdbd

SHA3-512 [HC: 17600] [JtR: raw-sha3]

echo ba6e8f9cd4140ac8b8d2bf96c9acd2fb58c0827d556b78e331d1113fcbfe425ca9299fe917f6015978f7e1644382d1ea45fd581aed6298acde2fa01e7d83cdbd > advice_7

kek keccak topkek

sha3

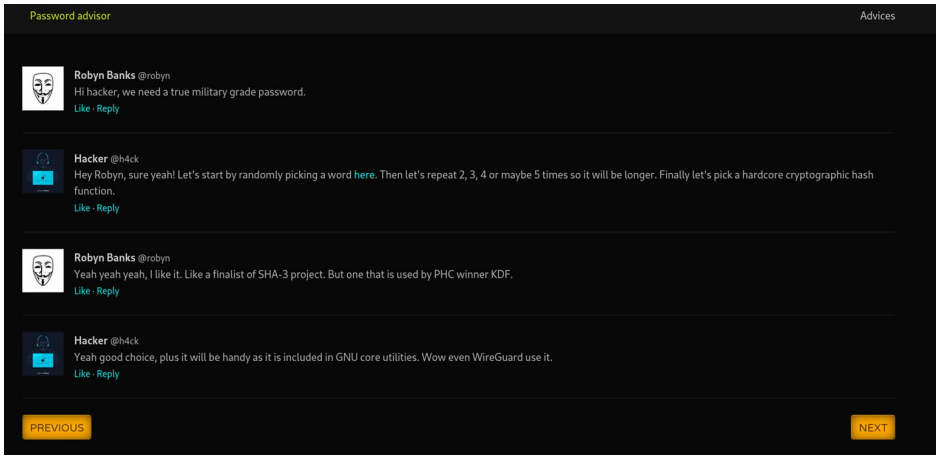
john --format=raw-sha3 --wordlist=/usr/share/wordlists/rockyou.txt advice_7

john --show --format=raw-sha3 advice_7

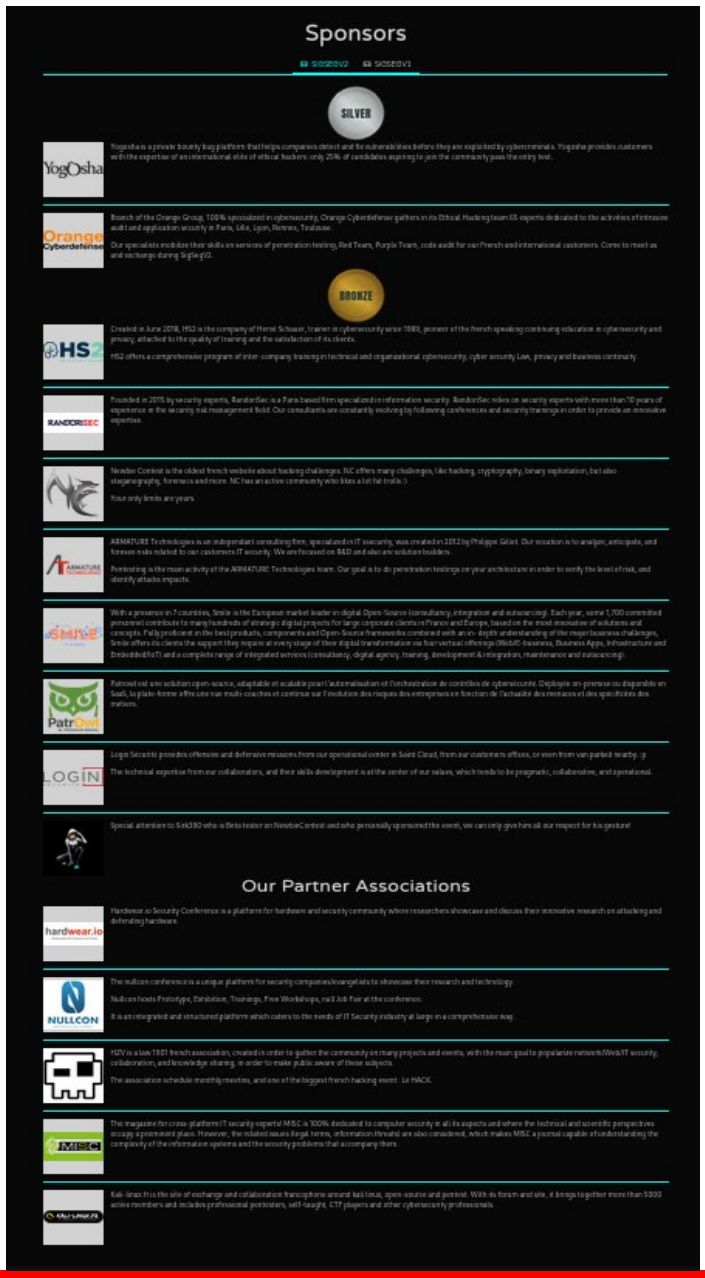
!@#redrose!@#

Advice n°8

9f7376709d3fe09b389a27876834a13c6f275ed9a806d4c8df78f0ce1aad8fb343316133e810096e09
99eaf1d2bca37c336e1b7726b213e001333d636e896617



haiti



9f7376709d3fe09b389a27876834a13c6f275ed9a806d4c8df78f0ce1aad8fb343316133e810096e0999eaf1d2bca37c336e1b7726b213e001333d636e896617

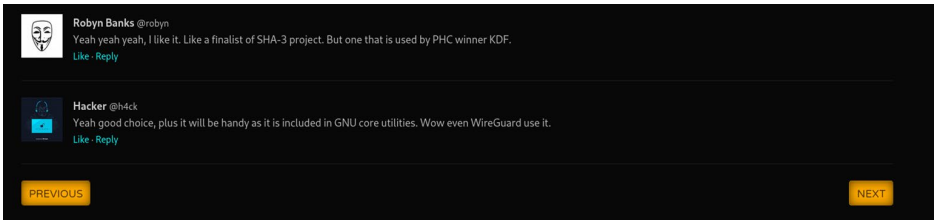
SHA-512 [HC: 1700] [JtR: raw-sha512]

SHA3-512 [HC: 17600] [JtR: raw-sha3]

SHA3-512 [HC: 17600] [JtR: dynamic_400]

Keccak-512 [HC: 18000] [JtR: raw-keccak]

BLAKE2-512 [JtR: raw-blake2]



https://en.wikipedia.org/wiki/WireGuard

https://www.wireguard.com/

wireguard use BLAKE2

echo 9f7376709d3fe09b389a27876834a13c6f275ed9a806d4c8df78f0ce1aad8fb343316133e810096e0999eaf1d2bca37c336e1b7726b213e001333d636e896617 > advice_8

use Web scrapping to get extract content and data from a website

cewl http://10.10.1.192/rtfm.re/en/sponsors/index.html -w words.txt

modified the john.cong and add r for recursive

nano /etc/john/john.conf

```
# Rules for advice 8 form TryHackMe
[List.Rules:d]
d
dd
dd
```

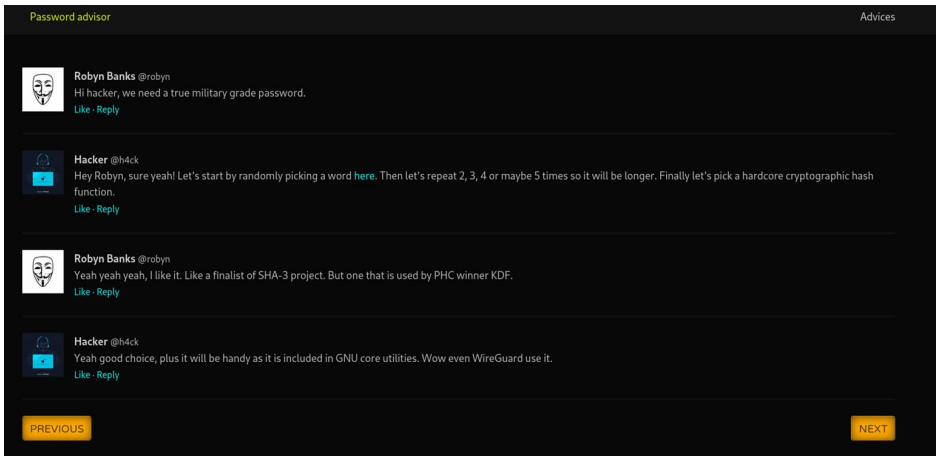
john --format=raw-blake2 --wordlist=words.txt --rule=d advice_8

john --show --format=raw-blake2 advice_8

hackinghackinghackinghacking

Advice n°9

6kI6VJ0a31.SNRsLR$Wk30X8w8iEC2FpasTo0Z5U7wke0TpfbDtSwayrNebqKjYWC4gjKoNEJxO/DkP.YFTLVFirQ5PEh4glQIHuKfA/



look for the type of hash

https://hashcat.net/wiki/doku.php?id=example_hashes

1800	sha512crypt 6, SHA512 (Unix) [2]	$6$52450745$k5ka2p8bFuSmoVT1tzOyyuaREkkKBcCNqoDKzYiJL9RaE8yMnPgh2XzzF0NDrUhgrcLwg78xs1w5pJiypEdFX/

look for sha512 in jonh

john --list=formats | grep -iF "512"

sha512crypt

echo 6kI6VJ0a31.SNRsLR$Wk30X8w8iEC2FpasTo0Z5U7wke0TpfbDtSwayrNebqKjYWC4gjKoNEJxO/DkP.YFTLVFirQ5PEh4glQIHuKfA/ > advice_9

john --format=sha512crypt --wordlist=/usr/share/wordlists/rockyou.txt advice_9

john --show --format=sha512crypt advice_9

kakashi1