HASHES, HASHES EVERYWHERE, BUT ALL I SEE IS PLAINTEXT...

$whoami

in.security

Will Hunt

- Co-founder of in.security
- 10+ years in cyber
- Hacker, formerly digital forensics
- Trainer/speaker @ Black Hat USA/EU, Nolacon, 44CON, Infosec Europe IICSG etc
- Assists UK government
- @Stealthsploit

Agenda

in.security

- Traditional cracking recap
- Password guidelines – old vs new
- Passphrases
- Creative / advanced cracking
- GPUs vs CPUs: A crypto-wallet story
- Foreign language cracking in h3x
- Secure p@ssw0rd advice

Before We Start

in.security

```
92f0bda3efc61c339dc50c06a264e444:discussing offline password attacks

Session..........: hashcat
Status...........: Cracked
Hash.Name........: NTLM
Hash.Target......: 92f0bda3efc61c339dc50c06a264e444
Time.Started.....: Fri Apr 26 22:54:57 2019 (1 sec)
Time.Estimated...: Fri Apr 26 22:54:58 2019 (0 secs)
Guess.Base.......: File (dictionary)
Guess.Mod........: Rules (rules/OneRuleToRuleThemAll.rule)
```

Common Techniques

in.security

Dictionary + Rules

Mask / Hybrid

Rainbow Table

Pure Brute Force

Success criteria

Algorithm complexity

Password length / complexity

Known / predictable elements

Hardware

Common Techniques

in.security

- Dictionary and rules

```
hashcat -m1000 hashes.txt rockyou.txt -r
OneRuleToRuleThemAll.rule
```

- Mask

```
hashcat -m0 hashes.txt -a3 -1 ?l?d ?1?1?1?1?1?1?1?1?1
```

- Dictionaries

https://github.com/xajkep/wordlists

- OneRuleToRuleThemAll

https://github.com/stealthsploit/Optimised-hashcat-Rule

```
 ? | Charset
===+=========
 l | abcdefghijklmnopqrstuvwxyz
 u | ABCDEFGHIJKLMNOPQRSTUVWXYZ
 d | 0123456789
 h | 0123456789abcdef
 H | 0123456789ABCDEF
 s |  !"#$%%&'()*+,-./:;<=>?@[\
 a | ?l?u?d?s
 b | 0x00 - 0xff

-1, --custom-charset1
-2, --custom-charset2
-3, --custom-charset3
-4, --custom-charset4
```

Common Techniques

in.security

- Hybrid

```
hashcat -m0 hashes.txt -a6 rockyou.txt ?d?d?d
```

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

- Pure brute force

```
hashcat -m1000 hashes.txt -a3 ?a?a?a?a?a?a?a?a?a?a?a?a --increment
```

- Rainbow tables - rarely situationally useful for single unsalted hashes

Key Space

in.security

- Key space = char set ^ length
  - Mixed alpha-numeric + special = 95 printable ASCII chars

  - 9 chars using mixed alpha-numeric + special
  - 10 chars using mixed alpha-numeric

Which password is stronger?

9 chars = 95^9 = 630,249,409,724,609,375

10 chars = 62^10 = 839,299,365,868,340,224

ASCII printable characters

32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

It's all about the length

Pure Brute Force

in.security

- 8x NVIDIA GTX 2080 Ti cracking NTLM @ 822 GH/s* (95 char set)
  - 8 char = <2.5 hours
  - 9 char = 9 days
  - 10 char = 2.3 years
  - 11 char = 219 years
  - 12 char = 20,840 years
  - 13 char = 2 million years

* https://twitter.com/hashcat/status/1095807014079512579/

Pure Brute Force

in.security

Estimated Password Recovery Times — 1x Terahash Brutalis, 44x Terahash Inmanis (448x Nvidia RTX 2080)

Full US keyboard mask attack with Terahash Hashstack

NTLM	31.82 TH/s	Instant	Instant	Instant	Instant	3 mins 29 secs	5 hrs 30 mins	3 wks 0 day	5 yrs 7 mos	538 yrs 1 mo	51.2 mil
MD5	17.77 TH/s	Instant	Instant	Instant	Instant	6 mins 14 secs	9 hrs 50 mins	1 mo 1 wk	10 yrs 1 mo	963 yrs 4 mos	91.6 mil
NetNTLMv1 / NetNTLMv1+ESS	16.82 TH/s	Instant	Instant	Instant	Instant	6 mins 35 secs	10 hrs 24 mins	1 mo 1 wk	10 yrs 8 mos	1 mil	96.8 mil
LM	15.81 TH/s	Instant	Instant	Instant	Instant						
SHA1	5.89 TH/s	Instant	Instant	Instant	Instant	18 mins 47 secs	1 day 5 hrs	3 mos 3 wks	30 yrs 7 mos	2.9 mil	276.3 mil
SHA2-256	2.42 TH/s	Instant	Instant	Instant	Instant	45 mins 39 secs	3 days 0 hr	9 mos 1 wk	74 yrs 4 mos	7.1 mil	671.9 mil
NetNTLMv2	1.22 TH/s	Instant	Instant	Instant	Instant	1 hr 30 mins	5 days 23 hrs	1 yr 6 mos	147 yrs 10 mos	14.1 mil	1335.5 mil
SHA2-512	801.9 GH/s	Instant	Instant	Instant	1 min 28 secs	2 hrs 17 mins	1 wk 2 days	2 yrs 4 mos	224 yrs 9 mos	21.4 mil	2029.7 mil
descrypt, DES (Unix), Traditional DES	647.59 GH/s	Instant	Instant	Instant	1 min 48 secs	2 hrs 50 mins	1 wk 4 days	2 yrs 11 mos	278 yrs 3 mos	26.5 mil	2513.3 mil
Kerberos 5, etype 23, TGS-REP	206.97 GH/s	Instant	Instant	Instant	5 mins 38 secs	8 hrs 54 mins	1 mo 0 wk	9 yrs 2 mos	870 yrs 10 mos	82.8 mil	7864 mil
Kerberos 5, etype 23, AS-REQ Pre-Auth	206.78 GH/s	Instant	Instant	Instant	5 mins 38 secs	8 hrs 54 mins	1 mo 0 wk	9 yrs 2 mos	871 yrs 8 mos	82.9 mil	7871.2 mil
md5crypt, MD5 (Unix), Cisco-IOS 1 (MD5)	7.61 GH/s	Instant	Instant	1 min 37 secs	2 hrs 33 mins	1 wk 3 days	2 yrs 7 mos	249 yrs 5 mos	23.7 mil	2252.6 mil	213995.1 mil
LastPass + LastPass sniffed	1.78 GH/s	Instant	Instant	6 mins 52 secs	10 hrs 52 mins	1 mo 1 wk	11 yrs 2 mos	1.1 mil	101.1 mil	9600.8 mil	912079.6 mil
macOS v10.8+ (PBKDF2-SHA512)	335.09 MH/s	Instant	Instant	36 mins 34 secs	2 days 9 hrs	7 mos 2 wks	59 yrs 7 mos	5.7 mil	538.2 mil	51127.7 mil	4857134 mil
WPA-EAPOL-PBKDF2	277.23 MH/s					9 mos 0 wk	72 yrs 0 mo	6.8 mil	650.5 mil	61799.3 mil	5870931.8 mil
TrueCrypt RIPEMD160 + XTS 512 bit	211.78 MH/s	Instant	Instant	57 mins 52 secs	3 days 19 hrs	11 mos 3 wks	94 yrs 3 mos	9 mil	851.6 mil	80899.5 mil	7685455.6 mil
7-Zip	181.51 MH/s	Instant	Instant	1 hr 7 mins	4 days 10 hrs	1 yr 1 mo	110 yrs 0 mo	10.5 mil	993.6 mil	94389.2 mil	8966975.1 mil
sha512crypt 6, SHA512 (Unix)	119.46 MH/s	Instant	1 min 5 secs	1 hr 42 mins	6 days 18 hrs	1 yr 9 mos	167 yrs 2 mos	15.9 mil	1509.7 mil	143419.6 mil	13624861.4 mil
DPAPI masterkey file v1	47.23 MH/s	Instant	2 mins 44 secs	4 hrs 19 mins	2 wks 3 days	4 yrs 5 mos	422 yrs 10 mos	40.2 mil	3818.1 mil	362723.1 mil	34458696.1 mil
RAR5	28.15 MH/s	Instant	4 mins 35 secs	7 hrs 15 mins	4 wks 0 day	7 yrs 5 mos	709 yrs 7 mos	67.4 mil	6407.6 mil	608720.6 mil	57828453.9 mil
DPAPI masterkey file v2	27.82 MH/s	Instant	4 mins 39 secs	7 hrs 20 mins	4 wks 1 day	7 yrs 6 mos	717 yrs 10 mos	68.2 mil	6482.1 mil	615797.6 mil	58500769.5 mil
RAR3-hp	20.84 MH/s	Instant	6 mins 12 secs	9 hrs 47 mins	1 mo 1 wk	10 yrs 1 mo	958 yrs 2 mos	91.1 mil	8652.3 mil	821972.3 mil	78087367.8 mil
KeePass 1 (AES/Twofish) and KeePass 2 (AES)	17.8 MH/s	Instant	7 mins 15 secs	11 hrs 28 mins	1 mo 2 wks	11 yrs 9 mos	1.1 mil	106.7 mil	10131.9 mil	962529.5 mil	91440305.8 mil
bcrypt $2*$, Blowfish (Unix)	11.37 MH/s	Instant	11 mins 21 secs	17 hrs 57 mins	2 mos 1 wk	18 yrs 5 mos	1.8 mil	167 mil	15860.3 mil	1506727.9 mil	143139150.9 mil
Bitcoin/Litecoin wallet.dat	3.55 MH/s	Instant	36 mins 18 secs	2 days 9 hrs	7 mos 2 wks	59 yrs 1 mo	5.6 mil	534.1 mil	50743.7 mil	4820655.6 mil	457962282.7 mil
	Speed	Length 4	Length 5	Length 6	Length 7	Length 8	Length 9	Length 10	Length 11	Length 12	Length 13

https://twitter.com/TerahashCorp/status/1155128018156892160

Guidelines – Old vs New

in.security

Old Guidelines

- Minimum 8 chars
- Mixed alpha-numeric and special
- Change password every 90 days

New Guidelines

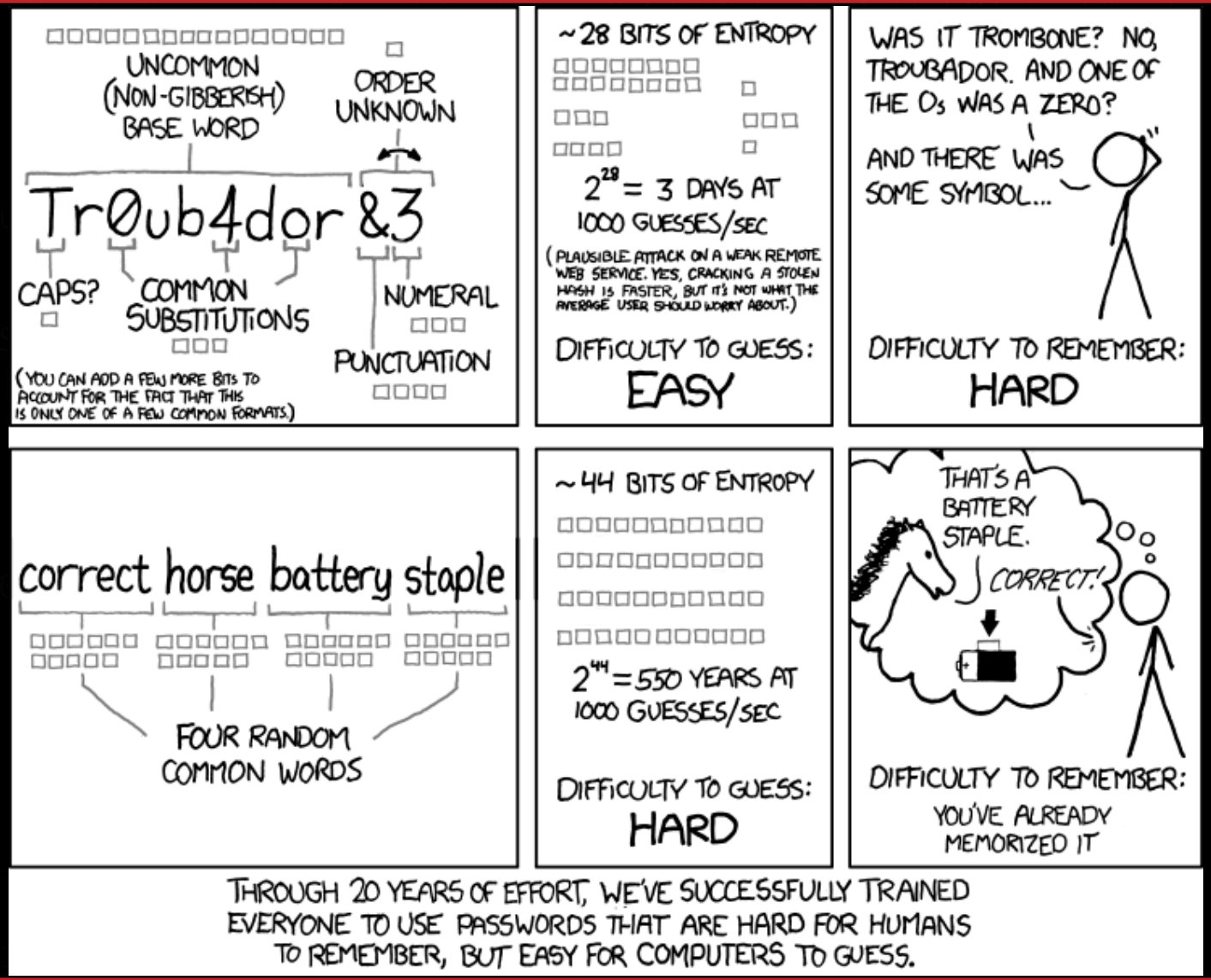
- NIST SP 800-63B*
- "..at least 64 characters" (still min 8)
- Complexity recommended not required
- No truncation, password hints or KBA
- No expiry without justification
- Compare to dictionaries / breaches

tl;dr – passphrases over passwords

* https://pages.nist.gov/800-63-3/sp800-63b.html

Guidelines – Old vs New

in.security



https://xkcd.com/936/

SOME RANDOM WORDS YOU SAY?

EXCELLENT...

Attacking Passphrases

in.security

- Combinator attacks - dictionary combined with a dictionary
  - Google top 10,000 words https://github.com/first20hours/google-10000-english/blob/master/google-10000-english.txt
  - Google top 20,000 words https://github.com/first20hours/google-10000-english/blob/master/20k.txt
- @netmux has a great guide - https://www.netmux.com/blog/cracking-12-character-above-passwords

```
hashcat -m0 hashes.txt -a1 20k.txt 20k.txt
```

Moar Words

in.security

- Combinator - https://github.com/hashcat/hashcat-utils

```
./combinator 20k.txt 20k.txt > 20k-combined
```

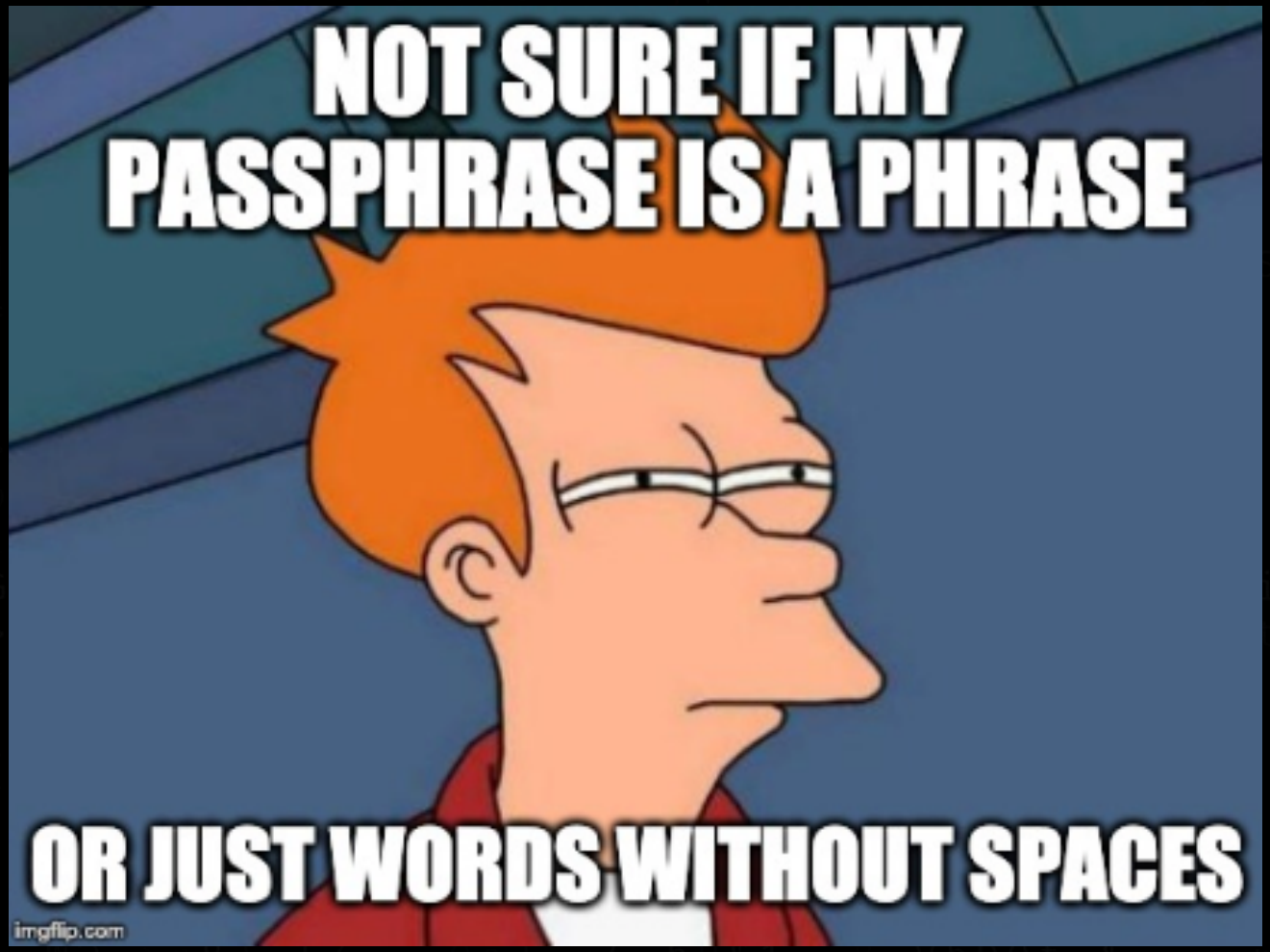
```
▶ head -5 20k-combined
thethe
theof
theand
theto
thea
```

- 3 or 4 words

```
hashcat -m0 hashes.txt -a1 20k-combined 20k-combined.txt
```

```
▶ hashcat -a1 20k-combined 20k.txt
 --stdout | more
thethethe
thetheof
thetheand
thetheto
```

```
▶ hashcat -a1 20k-combined 20k-com
bined --stdout | more
thethethethe
thethetheof
thethetheand
thethetheto
```


NOT SURE IF MY PASSPHRASE IS A PHRASE
OR JUST WORDS WITHOUT SPACES
imgflip.com

Delimiters

in.security

```
awk '{print $0" "}' 20k.txt > 20k-space
```

```
▶ head -2 20k-space
the
of
```

```
▶ awk '{print length}' 20k-space| head -2
4
3
```

```
./combinator 20k-space 20k.txt > 20k-combined-mid-space
```

```
▶ head -2 20k-combined-mid-space
the the
the of
```

```
▶ awk '{print length}' 20k-combined-mid-space| head -2
7
6
```

3 and 4 Words

in.security

Combinator rules

```
-j, --rule-left          | Rule | Single rule applied to each word from left wordlist  | -j 'c'
-k, --rule-right         | Rule | Single rule applied to each word from right wordlist | -k '^-'
```

```
hashcat -m0 hashes -a1 20k-combined-mid-space -j '$ ' 20k.txt
```

```
the the the
the the of
```

```
hashcat -m0 hashes -a1 20k-combined-mid-space -j '$ ' 20k-combined-mid-space
```

```
the the the the
the the the of
```

Combinator All The Rules

in.security

Add another space and pipe to hashcat

```
awk '{print $0" "}' 20k-combined-mid-space > 20k-combined-mid-end-space
```

```
▶ head -2 20k-combined-mid-end-space
the the
the of
```

```
▶ awk '{print length}' 20k-combined-mid-end-space| head -2
8
7
```

```
./combinator 20k-combined-mid-end-space 20k-combined-mid-space | hashcat
-m1000 hashes.txt -r rules/OneRuleToRuleThemAll.rule -w4
```

- Optimised kernels (-O) increases speed but limits length
  - E.g. MD5 optimised = 31 / NTLM optimised = 27

Expander

in.security

- https://github.com/hashcat/hashcat-utils/
- Splits candidates into single chars, mutates & reconstructs
- Recompile with LEN_MAX 8 and always unique output

```
./expander < wordlist.txt | sort -u
```

```
GNU nano 2.0.   File: expander.c

#include <ctype.h>
#include <errno.h>
#include <fcntl.h>
#include "utils.c"

#define LEN_MIN 1
#define LEN_MAX 4
```

```
▶ echo hashcat | ./expander
  | sort -u | head -10
a
as
ash
ashc
at
ath
c
ca
cat
cath
```

```
▶ echo hashcat | ./expander8
  | sort -u | head -10
a
as
ash
ashc
ashca
ashcat
ashcath
at
ath
athas
```

Fingerprint Attack

in.security

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

1) Expand previously cracked passwords
2) Combo the resulting file with itself
3) Update (expand) wordlist
4) Repeat and rinse

```
cut -d: -f2- < hashcat.potfile | ./expander | sort -u > word.list
hashcat -m1000 hashes.txt --remove -a1 word.list word.list -o word.list2
cut -d: -f2- < word.list2 | ./expander | sort -u > word.list3
hashcat -m1000 hashes.txt --remove -a1 word.list3 word.list3 -o word.list4
```

PRINCE

in.security

PRobability INfinite Chained Elements (PRINCE)

- princeprocessor - Jens Steube "Atom" @hashcat
- Single dictionary input --> builds chains of 1 to *n* words
- E.g. if princeprocessor was guessing 4 char candidates...
  - 4 letter word
  - 2 letter word + 2 letter word
  - 1 letter word + 3 letter word
  - 3 letter word + 1 letter word
  - 1 letter word + 1 letter word + 2 letter word
  - 1 letter word + 2 letter word + 1 letter word
  - 2 letter word + 1 letter word + 1 letter word
  - 1 letter word + 1 letter word + 1 letter word + 1 letter word

https://github.com/hashcat/princeprocessor

PRINCE Attacks

in.security

- De-dupe dictionary before running prince

PRINCE Attack

```
./pp.bin rockyou.txt --pw-min=8 | hashcat -m1000 hashes.txt
```

PRINCEPTION - @jmgosney

```
./pp.bin rockyou.txt --pw-min=8 | ./pp.bin --pw-min=8 |
hashcat -m1000 hashes.txt -g 300000
```

- Delimiter request open:
  https://github.com/hashcat/princeprocessor/issues/49

Passphrases with PRINCE

in.security

- Traditional delimited combinator

```
-j, --rule-left          | Rule | Single rule applied to each word from left wordlist  | -j 'c'
-k, --rule-right         | Rule | Single rule applied to each word from right wordlist | -k '^-'
```

```
hashcat -m0 hashes -a1 20k-combined-mid-space -j '$ ' 20k-combined-mid-space
```

- pp variant (thanks @TychoTithonus & @Chick3nman512)

```
./pp.bin --elem-cnt-min=4 --elem-cnt-max=4 --pw-min=8 20k-space |
hashcat -j ] -m1000 hashes.txt -w3 -O -r prince_optimized.rule
```

- If dictionary & combo attacks fail - <u>USE PRINCE</u>

Purple Rain

in.security

...and as a last resort

- Shuffled wordlist --> different output
- https://www.netmux.com/blog/purple-rain-attack

```
▶ ./pp64.bin --pw-min=8 rockyou.txt | head -8
password
iloveyou
princess
12345678
babygirl
michelle
sunshine
football

~/Tools/Dictionaries  master ✗
▶ gshuf rockyou.txt | ./pp64.bin --pw-min=8 | head -8
rosazila
1hangout
f3f4f1f2
bangcong
lucky813
tohatchi
ionutiza
chipinde
```

```
shuf rockyou.txt | ./pp.bin --pw-min=8 | hashcat -m0 hashes --remove -w3 -O -g 300000
```

1. root@Crackers: /tools (ssh)

root@Crackers: /tool... ⌘1 ~ (zsh) ⌘2

```
root@Crackers:/tools#
```

Stats

in.security

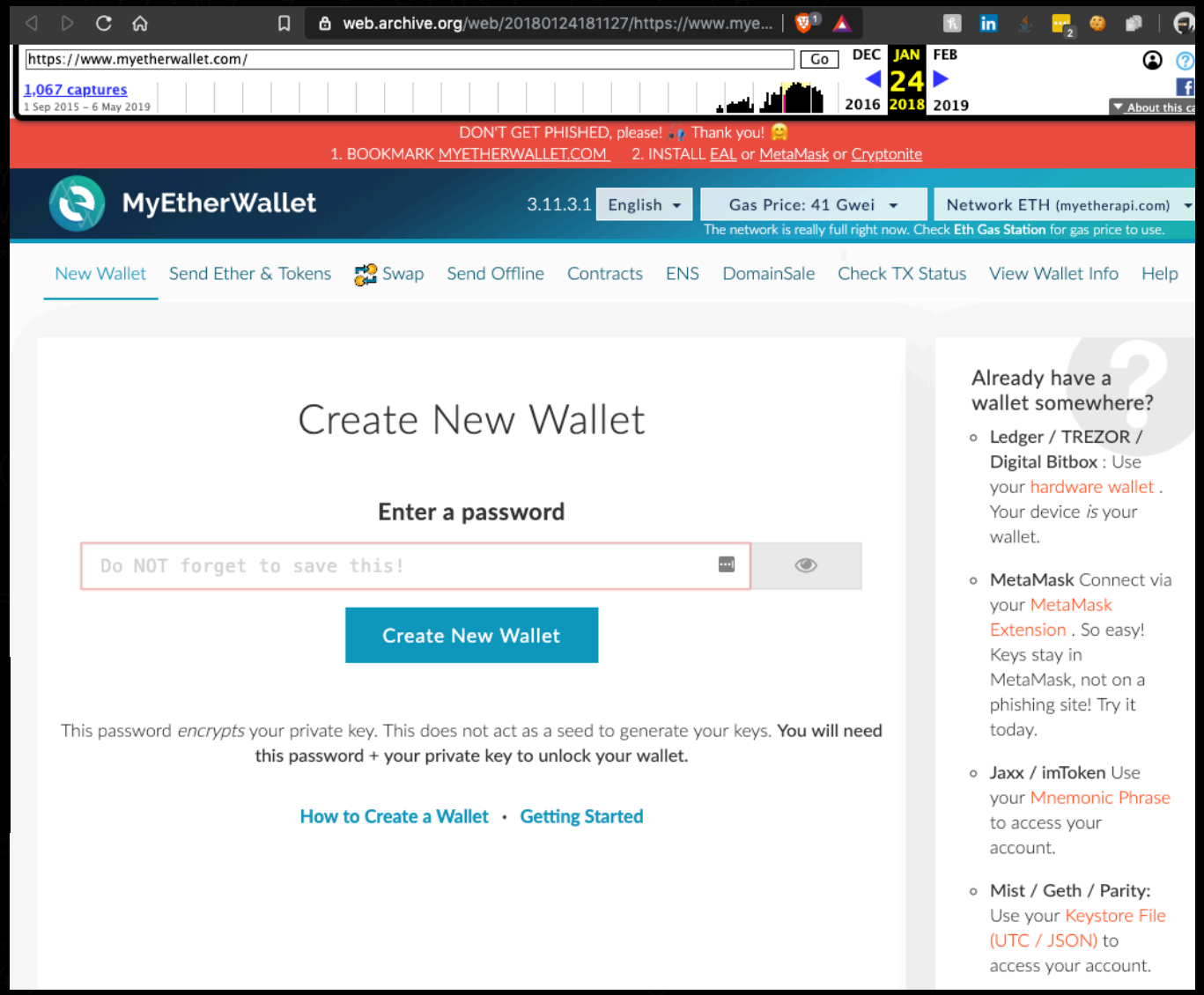
Attack (Lastfm-top500k (358863 deduped))	Cracked
Rockyou.txt + OneRuleToRuleThemAll	290337/358863 (80.9%)
5-char brute force --> Fingerprint attack 1	615/68526 (0.9%)
Fingerprint attack 2	9194/67911 (13.54%)
Fingerprint attack 3	5860/58717 (9.98%)
Fingerprint attack 4	805/52857 (1.52%)
Fingerprint attack 5	84/52052 (0.16%)
Fingerprint attack 6	9/51968 (0.02%)
Fingerprint attack 7	0/51959
2hr - Prince	82/51959 (0.16%)
2hr - Prince + prince optimized rule	3325/51878 (6.41%) – 1 in potfile
2hr - Purple rain + 300000 self-generated rules	4298/48558 (8.85%) – 6 in potfile
2hr - Purple rain + 300000 self-generated rules	1610/44270 (3.64%) – 10 in potfile
2hr - Purple rain + OneRuleToRuleThemAll	7011/42660 (16.43%) – 1 in potfile - 35649 (Total 323214 (90% cracked))

Crypto Wallet Example

in.security

- Ethereum wallets – MyEtherWallet, Geth, Mist
- JSON keystore file option
- scrypt used in testing

```
~ git:(master) ✗ hashcat --help|grep -i ethereum
16300 | Ethereum Pre-Sale Wallet, PBKDF2-HMAC-SHA256
15600 | Ethereum Wallet, PBKDF2-HMAC-SHA256
15700 | Ethereum Wallet, SCRYPT
```

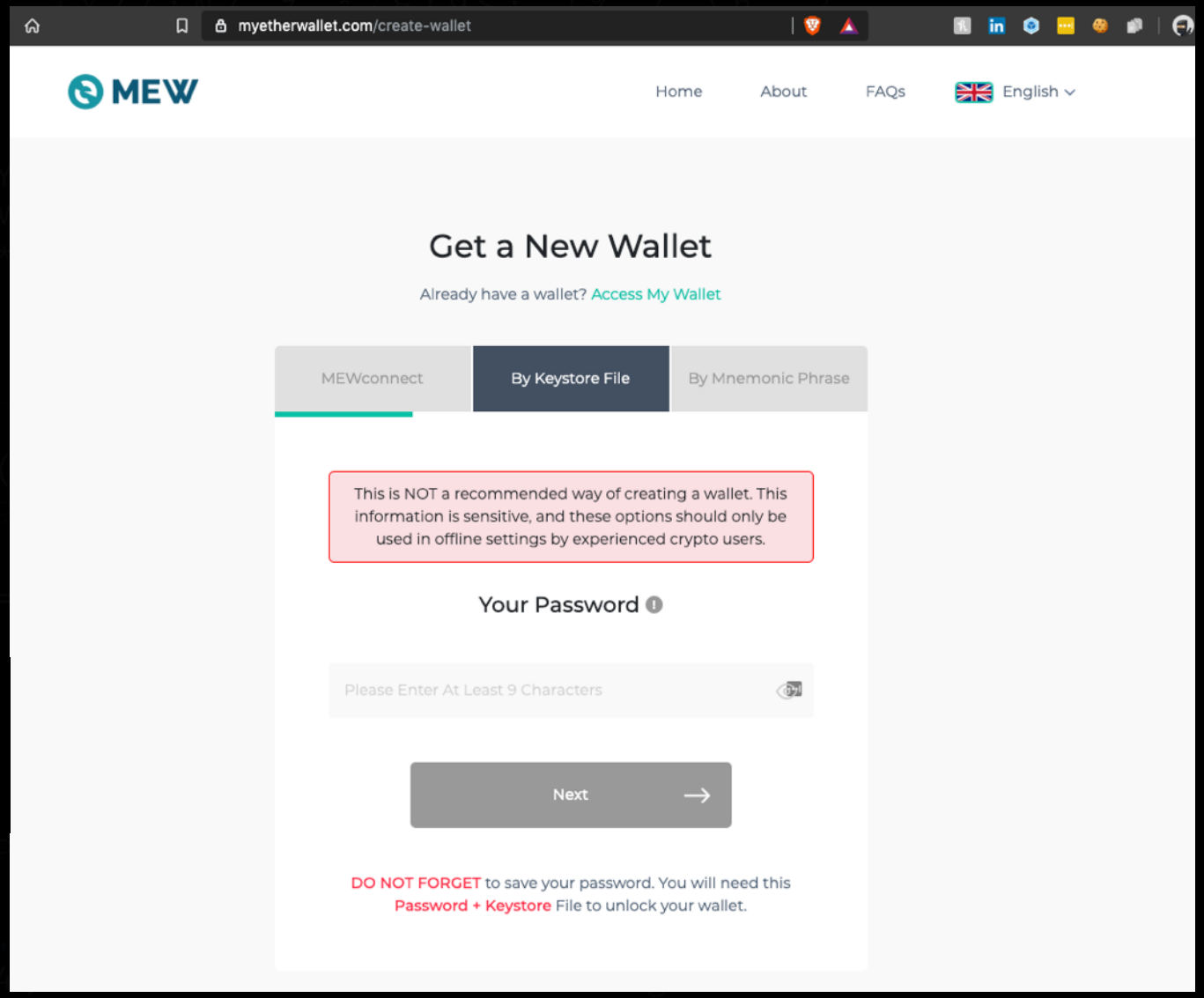


Crypto Wallet Example

in.security

- Ethereum wallets – MyEtherWallet, Geth, Mist
- JSON keystore file option
- scrypt used in testing

```
~ git:(master) ✗ hashcat --help|grep -i ethereum
16300 | Ethereum Pre-Sale Wallet, PBKDF2-HMAC-SHA256
15600 | Ethereum Wallet, PBKDF2-HMAC-SHA256
15700 | Ethereum Wallet, SCRYPT
```



Ethereum Wallets

in.security

- GPU cracking viability depends on *n*, *r*

- https://stealthsploit.com/2017/06/12/ethereum-wallet-cracking/

UTC–2017-06-10T11-51-33.675Z–f418f8185f2c1163ae953bf778acc6877b9bc203

{"version":3,"id":"5cf4711d-3f69-4636-89d0-b304a7e23b75",
"address":"f418f8185f2c1163ae953bf778acc6877b9bc203",
"Crypto":
{"ciphertext":"7f5c865554d67604394ae54d7a4f9735bdb85c90e606a672d18add1d167d793b",
"cipherparams":{"iv":"ae4e8c9c2ac201d6c2baa58ff670fd39"},
"cipher":"aes-128-ctr",
"kdf":"scrypt",
"kdfparams":
{"dklen":32,"salt":"437964c9bd1b5f63bde56560808c894792f8f670694590b776e22381e32dd33b","n":1024,"r":8,"p":1},
"mac":"96f2a849321cc04cb6c0fcee1bd4b195ca681ca28064dc45000f02e47230c5b6"}}

hashcat Format

$ethereum$s*1024*8*1*437964c9bd1b5f63bde56560808c894792f8f670694590b776e22381e32dd33b*7f5c865554d67604394ae54d7a4f9735bdb85c90e606a672d18add1d167d793b*96f2a849321cc04cb6c0fcee1bd4b195ca681ca28064dc45000f02e47230c5b6

Calculating Memory

in.security

- Determine memory requirements using *n*, *r*

Step 1: Calculate Single Computation per GPU

size_scrypt = (128 * *r*) * *n*

Step 2: Calculate Parallel Computations per GPU

Threads per compute unit(1) * no. of compute units(2)

= no. parallel computations

Step 3: Calculate RAM requirement per GPU

size_scrypt * no. parallel computations

	NVIDIA	AMD
(1)	32	64
(2)	Depends on GPU	

Calculating Memory

in.security

- In earlier post: n = 1024, r = 8

Step 1: Calculate Single Computation per GPU

size_scrypt = (128 * r) * n

(128 * 8) * 1024 = 1048576 bytes (1 MB)

Step 2: Calculate Parallel Computations per GPU

Threads per compute unit(1) * no. of compute units(2)

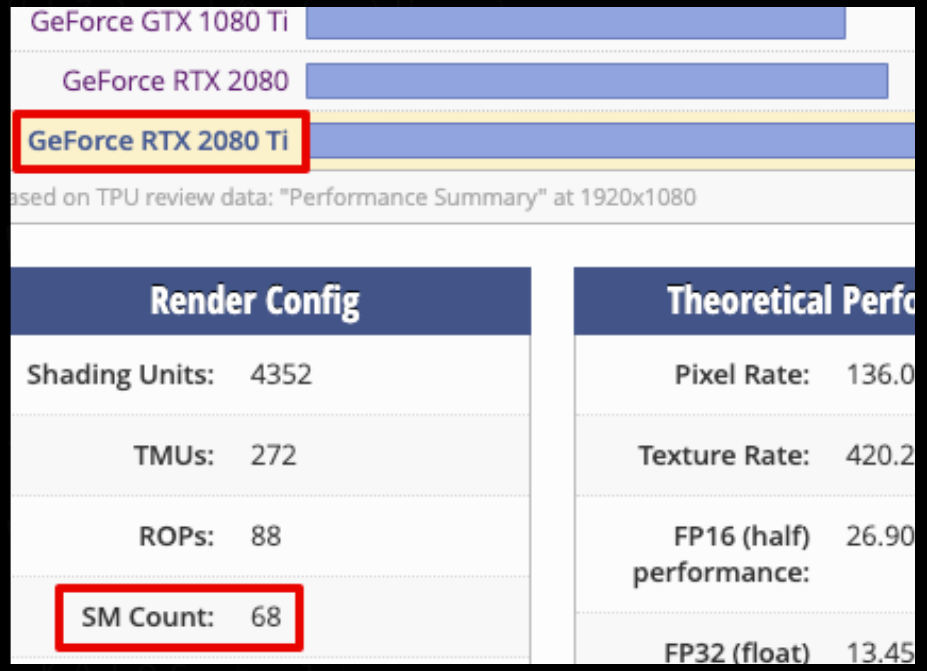
= no. parallel computations

32 * 68 = 2176

Step 3: Calculate RAM requirement per GPU

size_scrypt * no. parallel computations

1 MB * 2176 = **2176 MB per GPU**



https://www.techpowerup.com/gpu-specs/geforce-rtx-2080-ti.c3305

	NVIDIA	AMD
(1)	32	64
(2)	Depends on GPU	

Calculating Memory

in.security

```
➜ Downloads git:(master) ✗ cat UTC--2019-10-03T23-00-58.565Z--f3c1b02765004f1df0740225474fc5e3b509335a
{"version":3,"id":"dffea2fb-3eb4-461c-94a4-5f0f48eaab7b","address":"f3c1b02765004f1df0740225474fc5e3b50933
5a","crypto":{"ciphertext":"94b35c8afa0afc51e04f7381a4c9ace0d3609793a3170f5992e6ff98316db4de","cipherparam
s":{"iv":"a2bf2e38e69f3786560411f6140101e7"},"cipher":"aes-128-ctr","kdf":"scrypt","kdfparams":{"dklen":32
,"salt":"dfa15d5ede8d8010a627d9f2a0b27886dd4b14ae3fb2f9b4b5c4178248e16217","n":131072,"r":8,"p":1},"mac":"
9af060abec1d5d9b7ce185f564604b7c6de1251e64147300f307c9c967470d1c"}}%
```

size_scrypt = (128 * *r*) * *n*

(128 * 8) * 131072 = 134217728 bytes (128 MB)

Threads per compute unit(1) * no. of compute units(2) = no. parallel computations

32 * 68 = 2176

size_scrypt * no. parallel computations

128 MB * 2176 = 278528 MB = **272 GB RAM per GPU**

GPU vs CPU

in.security

```
-d, --opencl-devices          | Str  | OpenCL devices to use, separated with commas        | -d 1
-D, --opencl-device-types     | Str  | OpenCL device-types to use, separated with commas   | -D 1
```

- Cracking recent wallets on GPU will likely error, hang, blue screen etc.
- CPU to the rescue!
- https://stealthsploit.com/2018/01/04/ethereum-wallet-cracking-pt-2-gpu-vs-cpu/

```
- [ OpenCL Device Types ] -

# | Device Type
===+=============
1 | CPU
2 | GPU
3 | FPGA, DSP, Co-Processor
```

-D 1 -d <deviceID>

```
➜ hashcat git:(master) ✗ ./hashcat -I
hashcat (v5.1.0-905-g7c6970db) starting...

OpenCL Info:

Platform ID #1
  Vendor  : Apple
  Name    : Apple
  Version : OpenCL 1.2 (Feb 22 2019 20:16

Device ID #1
  Type           : CPU
  Vendor ID      : 4
  Vendor         : Intel
  Name           : Intel(R) Core(TM) i7
```

GPU vs CPU

in.security

- Convert with ethereum2john – `ethereum2john.py <wallet file>`
  https://github.com/magnumripper/JohnTheRipper/blob/bleeding-jumbo/run/ethereum2john.py

```
➜  Dictionaries git:(master) ✗ hashcat -m15700 '$ethereum$s*131072*8*1*919d207466fc4fe352fa75a6db8e2499c1f6
2f0c14db71a1da6286522ee3f270*2f8da158b01a9f144d9f89bfb04dac19643049c87488071100d068942c46c632*91bd22afeb668
d89800627dc760730709de03df38c54aea2a939bc465b90efe4' rockyou.txt -r OneRuleToRuleThemAll.rule -w3 -D1 -d1
hashcat (v5.1.0-905-g7c6970db) starting...

OpenCL Platform #1: Apple
=========================
* Device #1: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz, 4096/16384 MB allocatable, 8MCU
* Device #2: Intel(R) HD Graphics 630, skipped.
* Device #3: AMD Radeon Pro 555 Compute Engine, skipped.

Time.Started.....: Thu Jun 20 01:09:04 2019 (1 sec)
Time.Estimated...:                          ?
Guess.Base.......: File (rockyou.txt)
Guess.Mod........: Rules (OneRuleToRuleThemAll.rule)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:          ?   (285.00ms) @ Accel:1 Loops:1 Thr:1 Vec:1
```

GPU vs CPU

in.security

- Convert with ethereum2john – `ethereum2john.py <wallet file>`
  https://github.com/magnumripper/JohnTheRipper/blob/bleeding-jumbo/run/ethereum2john.py

```
➜  Dictionaries git:(master) ✗ hashcat -m15700 '$ethereum$s*131072*8*1*919d207466fc4fe352fa75a6db8e2499c1f6
2f0c14db71a1da6286522ee3f270*2f8da158b01a9f144d9f89bfb04dac19643049c87488071100d068942c46c632*91bd22afeb668
d89800627dc760730709de03df38c54aea2a939bc465b90efe4' rockyou.txt -r OneRuleToRuleThemAll.rule -w3 -D1 -d1
hashcat (v5.1.0-905-g7c6970db) starting...

OpenCL Platform #1: Apple
=========================
* Device #1: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz, 4096/16384 MB
* Device #2: Intel(R) HD Graphics 630, skipped.
* Device #3: AMD Radeon Pro 555 Compute Engine, skipped.

Time.Started.....: Thu Jun 20 01:09:04 2019 (1 sec)
Time.Estimated...: Sun Nov  5 22:49:12 2975 (956 years, 138 days)
Guess.Base.......: File (rockyou.txt)
Guess.Mod........: Rules (OneRuleToRuleThemAll.rule)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:      25 H/s (285.00ms) @ Accel:1 Loops:1 Thr:1 Vec:1
```

Non-ASCII Characters

in.security

- 256 ASCII characters – single byte encoding
- UTF-8 (1-4 byte encoding) for other characters
- Find the hex and pass to hashcat with `--hex-charset`

A	41	LATIN CAPITAL LETTER A
B	42	LATIN CAPITAL LETTER B
C	43	LATIN CAPITAL LETTER C

Β	ce 92	GREEK CAPITAL LETTER BETA
Γ	ce 93	GREEK CAPITAL LETTER GAMMA
Δ	ce 94	GREEK CAPITAL LETTER DELTA

https://www.utf8-chartable.de/unicode-utf8-table.pl

Non-ASCII Characters

in.security

- E.g. Arabic alphabet uses the following UTF-8 (hex) chars
- First byte

```
d8 d9 da db
```

- Second byte

```
80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f 90 91 92 93
94 95 96 97 98 99 9a 9b 9c 9d 9e 9f a0 a1 a2 a3 a4 a5 a6 a7
a8 a9 aa ab ac ad ae af b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb
bc bd be bf
```

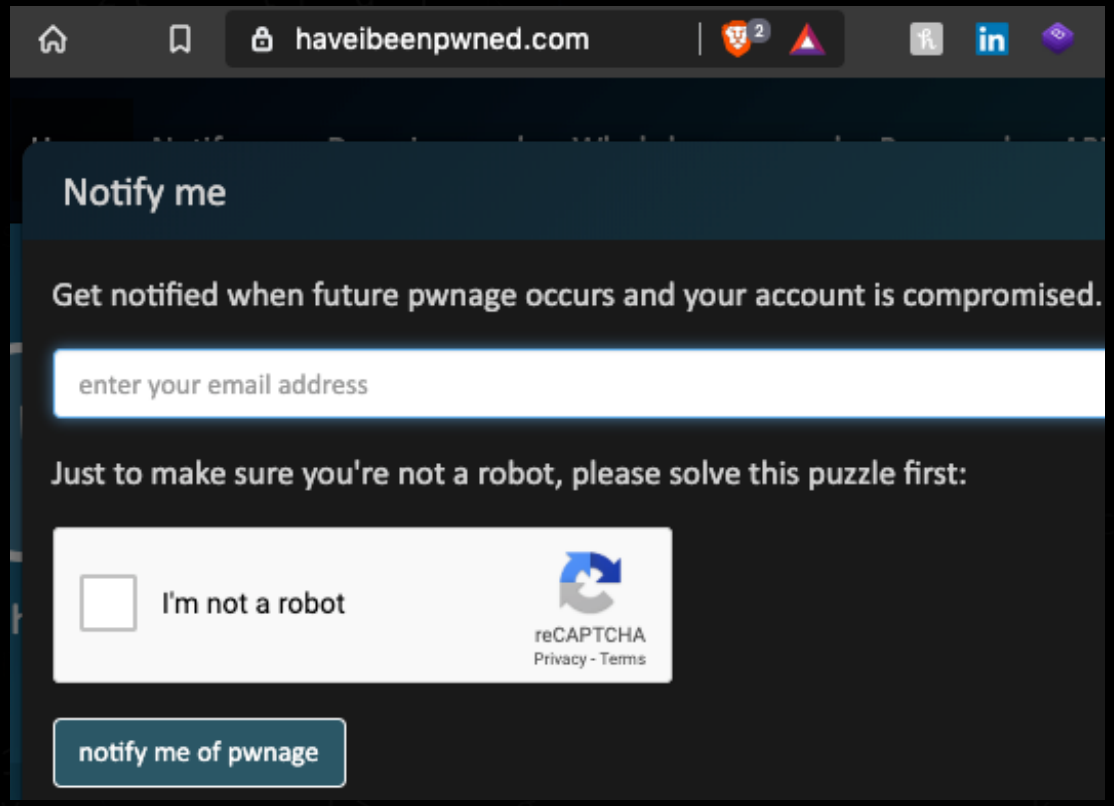
- The attack

```
hashcat -m0 a095531811f39557054b5340e5d2b182 -1 d8d9dadb -2
808182838485868788898a8b8c8d8e8f909192939495969798999a9b9c9d
9e9fa0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babb
bcbdbebf --hex-charset ?1?2?1?2?1?2?1?2?1?2 -w3 -a3
```

S3cure YOurs3lf

in.security

- Password managers – Lastpass, 1Password, Dashlane, Keepass etc
- Passphrases – 5+ random words
- Use spaces (other/differing delimiters)
- Avoid elements based on KBA
- 2FA / MFA
- Don't reuse passwords
- HIBP – https://haveibeenpwned.com/



Have a crack at our challenge
https://in.security/password-cracking-ctf/

Thank you!

https://in.security

contact@in.security

@Stealthsploit / @insecurity_ltd

in.security

We're in security to prevent insecurity