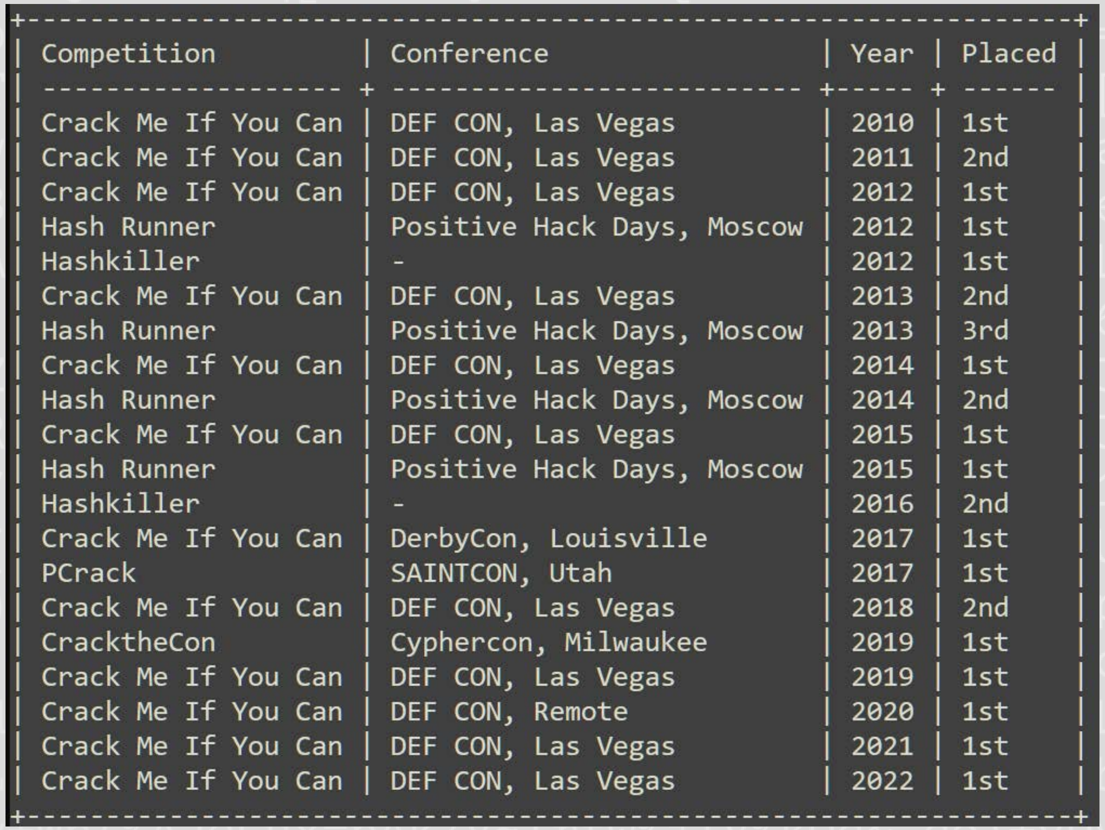
Follow the white rabbit down the rabbit hole

$whoami

Yiannis aka T0XlC, team Hashcat.

… i crack weird passwords for fun



```
+-----------------------------------------------------------------------------------+
| Competition        | Conference                    | Year | Placed |
| ------------------ + ------------------------ +----- + ------ |
| Crack Me If You Can | DEF CON, Las Vegas           | 2010 | 1st    |
| Crack Me If You Can | DEF CON, Las Vegas           | 2011 | 2nd    |
| Crack Me If You Can | DEF CON, Las Vegas           | 2012 | 1st    |
| Hash Runner         | Positive Hack Days, Moscow   | 2012 | 1st    |
| Hashkiller          | -                            | 2012 | 1st    |
| Crack Me If You Can | DEF CON, Las Vegas           | 2013 | 2nd    |
| Hash Runner         | Positive Hack Days, Moscow   | 2013 | 3rd    |
| Crack Me If You Can | DEF CON, Las Vegas           | 2014 | 1st    |
| Hash Runner         | Positive Hack Days, Moscow   | 2014 | 2nd    |
| Crack Me If You Can | DEF CON, Las Vegas           | 2015 | 1st    |
| Hash Runner         | Positive Hack Days, Moscow   | 2015 | 1st    |
| Hashkiller          | -                            | 2016 | 2nd    |
| Crack Me If You Can | DerbyCon, Louisville         | 2017 | 1st    |
| PCrack              | SAINTCON, Utah               | 2017 | 1st    |
| Crack Me If You Can | DEF CON, Las Vegas           | 2018 | 2nd    |
| CracktheCon         | Cyphercon, Milwaukee         | 2019 | 1st    |
| Crack Me If You Can | DEF CON, Las Vegas           | 2019 | 1st    |
| Crack Me If You Can | DEF CON, Remote              | 2020 | 1st    |
| Crack Me If You Can | DEF CON, Las Vegas           | 2021 | 1st    |
| Crack Me If You Can | DEF CON, Las Vegas           | 2022 | 1st    |
+-----------------------------------------------------------------------------------+
```

- T0XlC*.rule
- tmesis.pl

"This presentation is for awareness purposes only. The opinions expressed in this presentation are solely my own and does not necessarily represent the official position of my employer.".

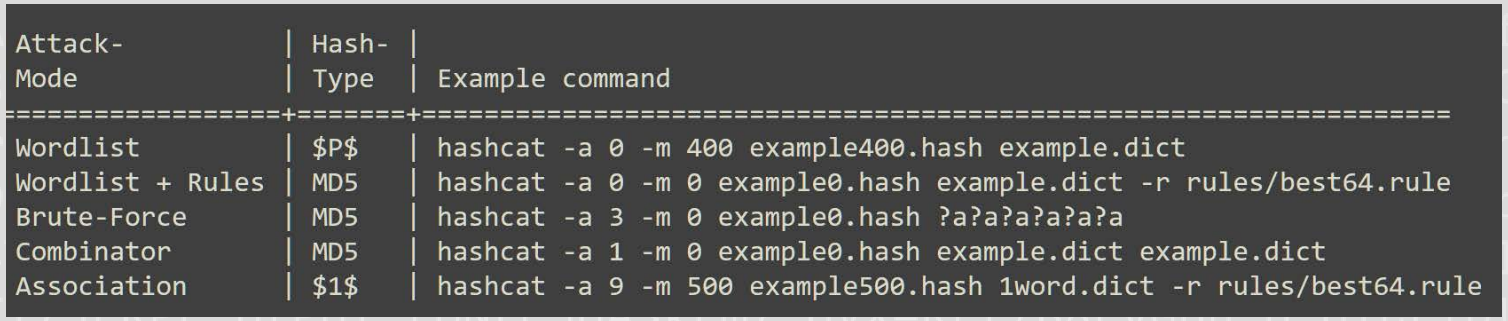
$ agenda

we are here

$ just to set the scene

- I crack hashes from public lists (Hashmob, Troyhunt’s sha1+ntlm lists etc..).
- I prefer to crack left lists (don’t care about the 98%) because if I am successful it means that the attack is new!
- I make wordlists out of EVERYTHING & I try to make rules out of everything!
- My attacks are done on a fairly small cracking rig and they are usually short in duration (from a couple of minutes to max couple of hours). That makes the attacks more applicable for both fast and slow hashes.

$ hashcat



```
Attack-           | Hash- |
Mode              | Type  | Example command
==================+=======+==================================================================
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.rule
```

473 different hash types!

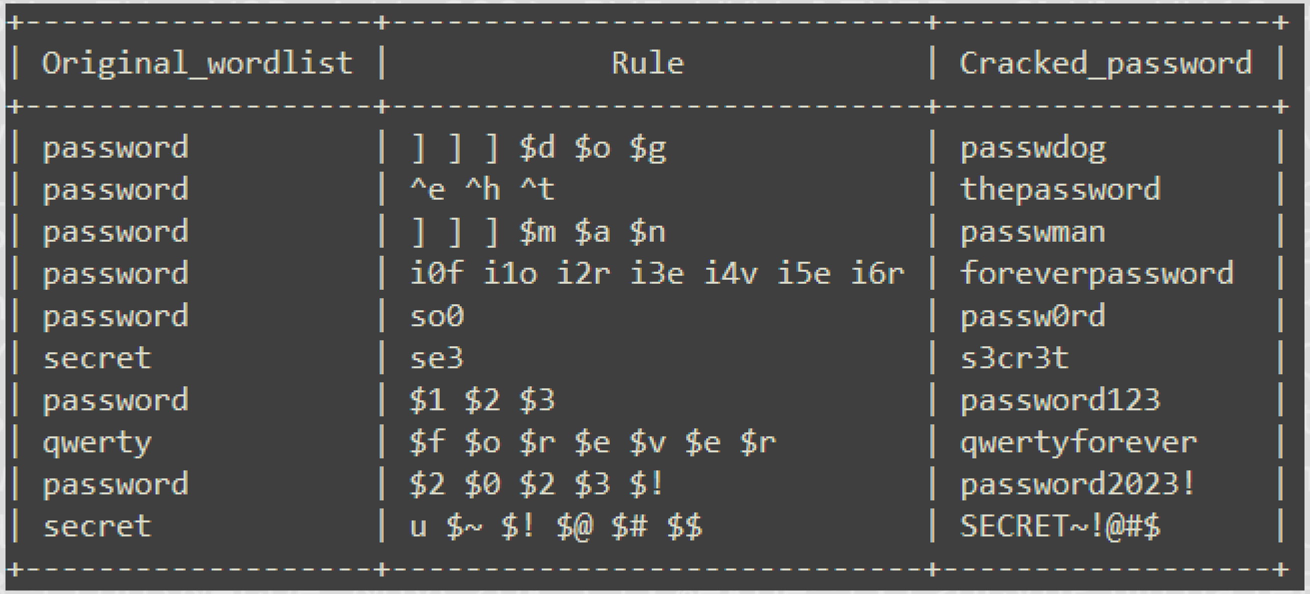
```
+------------------------------------------------------------------------------------------------+
|./hashcat –m 0 hash.txt wordlist.txt –a 0 –r rules.rule --debug-mode=4 --debug-file=debug.log –o cracked.txt |
+------------------------------------------------------------------------------------------------+
```

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

1 | Finding-Rule
2 | Original-Word
3 | Original-Word:Finding-Rule
4 | Original-Word:Finding-Rule:Processed-Word
5 | Original-Word:Finding-Rule:Processed-Word:Wordlist

$ hashcat

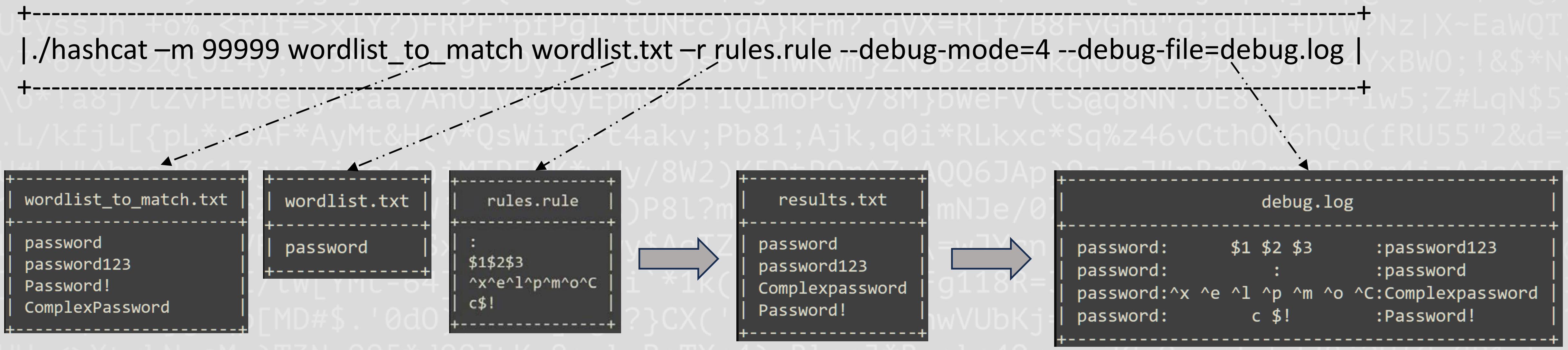
1 | Finding-Rule
2 | Original-Word
3 | Original-Word:Finding-Rule
4 | Original-Word:Finding-Rule:Processed-Word
5 | Original-Word:Finding-Rule:Processed-Word:Wordlist



```
+-------------------+-----------------------------+------------------+
| Original_wordlist |            Rule             | Cracked_password |
+-------------------+-----------------------------+------------------+
| password          | ] ] ] $d $o $g              | passwdog         |
| password          | ^e ^h ^t                    | thepassword      |
| password          | ] ] ] $m $a $n              | passwman         |
| password          | i0f i1o i2r i3e i4v i5e i6r | foreverpassword  |
| password          | so0                         | passw0rd         |
| secret            | se3                         | s3cr3t           |
| password          | $1 $2 $3                    | password123      |
| qwerty            | $f $o $r $e $v $e $r        | qwertyforever    |
| password          | $2 $0 $2 $3 $!              | password2023!    |
| secret            | u $~ $! $@ $# $$            | SECRET~!@#$      |
+-------------------+-----------------------------+------------------+
```

$ hashcat & mode 99999 & debug

Hash-Mode 99999 (Plaintext) – match the plaintext!



```
+-------------------------------------------------------------------------------------------------------------+
|./hashcat –m 99999 wordlist_to_match wordlist.txt –r rules.rule --debug-mode=4 --debug-file=debug.log |
+-------------------------------------------------------------------------------------------------------------+
```

```
+-------------------------+
| wordlist_to_match.txt |
+-------------------------+
| password               |
| password123            |
| Password!              |
| ComplexPassword        |
+-------------------------+
```

```
+----------------+
| wordlist.txt |
+----------------+
| password     |
+----------------+
```

```
+----------------+
|  rules.rule  |
+----------------+
| :              |
| $1$2$3         |
| ^x^e^l^p^m^o^C |
| c$!            |
+----------------+
```

```
+-----------------+
|   results.txt   |
+-----------------+
| password        |
| password123     |
| Complexpassword |
| Password!       |
+-----------------+
```

```
+------------------------------------------------------------+
|                         debug.log                          |
+------------------------------------------------------------+
| password:      $1 $2 $3      :password123                  |
| password:          :         :password                     |
| password:^x ^e ^l ^p ^m ^o ^C:Complexpassword              |
| password:        c $!        :Password!                    |
+------------------------------------------------------------+
```

$ hashcat & mode 99999 & debug (all your rules)

Why attempt to “crack” plains ?

Use the “by-product” of --debug to evaluate the effectiveness of your rules !

How ?

```
+--------------------------------------------------------------------------------------------------------------------+
| cat wordlist.txt | tr –d [:digit:] | tr [:upper:] [lower] | tr –d [:punct:] | sort | uniq –ic | sort –rn > root.txt   |
+--------------------------------------------------------------------------------------------------------------------+
```

Then try to “crack”/match the wordlist.txt with the rootwords.txt.

```
+--------------------------------------------------------------------------------------------------------------------+
|./hashcat –m 99999 wordlist.txt root.txt –r rules.rule --debug-mode=4 --debug-file=debug.log                          |
+--------------------------------------------------------------------------------------------------------------------+
```

Note: hashcat will include in debug passwords that were cracked without the rule mutation.
You will need to remove the entries where the plain was not mutated by the rule.

Or ?

2795352	u
2740396	c
2609537	t
2565187	E
964383	d
942349	r
866667	]
800447	p1
773299	p2
677815	Y1
644537	Z1
625570	q
619371	[
594618	$1
541240	Z2
481198	T0
446615	^1
392044	f
380633	K
348937	}
341948	y2
335551	y3
335066	{
329374	Y2

$ mode 99999 & debug

use hashcat to generate random rules

```
./hashcat -m 99999 Bigwordlist.txt rootwords.txt
```

`--debug-mode=5` `--debug-file=debug.log`	Original-Word:Finding-Rule:Processed-Word:Wordlist
`--keep-guessing`	Since we want to find good rules we don’t want to stop the attacks
`--generate-rules=NUM`	Generate some rules
`--generate-rules-func-min=NUM` `--generate-rules-func-max=NUM`	The min and max number of functions of the generated rules
`--generate-rules-func-sel=ioTlc`	The type of rule functions we want in the generated rules.

```
**cat wordlist.txt |  egrep .{12}$ | head -n 5000000
    cat wordlist | grep -vx '[0-9][0-9]*'
```

$ debug & mode 9999 & --generated rules

```
17937 s}Z 33W sEv 39^ TA $7 31]
16544 oB{ T0 @L @v 3AL
10510 } s>; l l c l
 7328 @8 @' TB } c 35* T4
 6862 T5 { T2 T8 sFb
 5698 sa, T1 { l { sPF
 3914 o8v c sm3 s83 s6s l $5 T5
 1316 T0 38I TB } TB
 1245 q l
 1026 s
  872 t d
  779 38V sJ[ o7x 39i } {
  760 d t
  747 { } { seD l T7 355 TB
  687 l d
  686 d c
  658 E d
  618 d l
  616 q c
  602 s2B s1x { 34o sCD T3 {
  579 p1 u
  557 f r
  553 d u
  551 p1 t
  542 @
  529 q r
  519 c d
  518 p2 l
```

These are statistics from <u>randomly</u> generated rules that performed really well!

$ agenda

we are here

$ Unicode Character Ranges

What is Unicode ?

an international encoding standard for use with different languages and scripts, by which each letter, digit, or symbol is assigned a unique numeric value that applies across different platforms and programs.

- Unicode 15.0 defines 327 blocks

Plane	Allocated code points	Assigned characters
0 BMP	65,520	55,634
1 SMP	26,160	23,276
2 SIP	60,912	60,873
3 TIP	9,136	9,131
14 SSP	368	337
15 SPUA-A	65,536	0 (by definition)
16 SPUA-B	65,536	0 (by definition)
Totals	293,168	**149,251 characters**

BMP

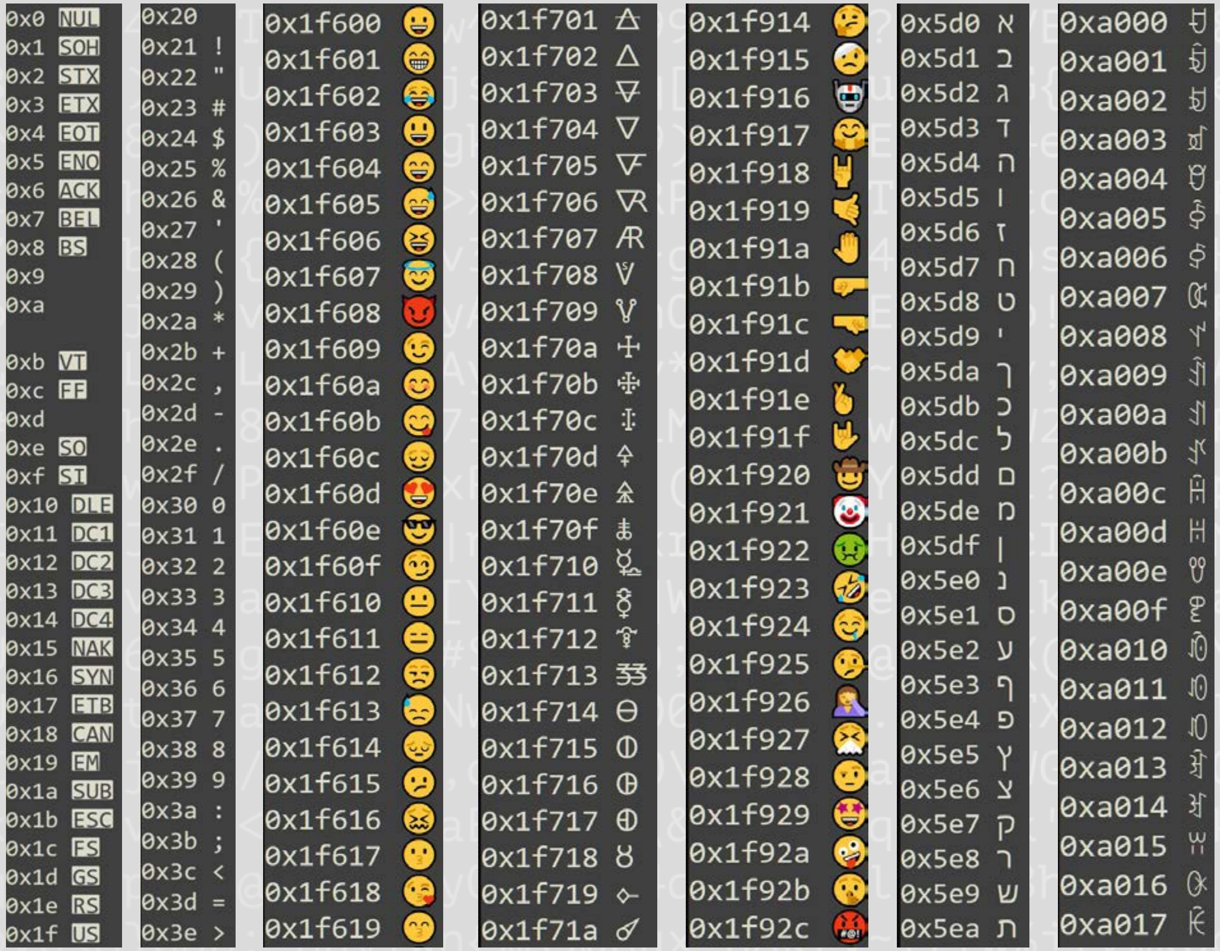
Block Range	Block Name	Block Range	Block Name
0020 — 007F	Basic Latin	2580 — 259F	Block Elements
00A0 — 00FF	Latin-1 Supplement	25A0 — 25FF	Geometric Shapes
0100 — 017F	Latin Extended-A	2600 — 26FF	Miscellaneous Symbols
0180 — 024F	Latin Extended-B	2700 — 27BF	Dingbats
0250 — 02AF	IPA Extensions	27C0 — 27EF	Miscellaneous Mathematical Symbols-A
02B0 — 02FF	Spacing Modifier Letters	27F0 — 27FF	Supplemental Arrows-A
0300 — 036F	Combining Diacritical Marks	2800 — 28FF	Braille Patterns
0370 — 03FF	Greek and Coptic	2900 — 297F	Supplemental Arrows-B
2460 — 24FF	Enclosed Alphanumerics	2F800 — 2FA1F	CJK Compatibility Ideographs Supplement
2500 — 257F	Box Drawing	E0000 — E007F	Tags

SMP

Block Range	Block Name	Block Range	Block Name
10000 — 1007F	Linear B Syllabary	**1F600–1F64F**	**Emoticons**
10080 — 100FF	Linear B Ideograms	1F650–1F67F	Ornamental Dingbats
10100 — 1013F	Aegean Numbers	1F680–1F6FF	Transport and Map Symbols
10140 — 1018F	Ancient Greek Numbers	1F700–1F77F	Alchemical Symbols
10190 — 101CF	Ancient Symbols	1F780–1F7FF	Geometric Shapes Extended
101D0 — 101FF	Phaistos Disc	1F800–1F8FF	Supplemental Arrows-C
10280 — 1029F	Lycian	1F900–1F9FF	Supplemental Symbols and Pictographs
102A0 — 102DF	Carian	1FA00–1FA6F	Chess Symbols
102E0 — 102FF	Coptic Epact Numbers	1FA70–1FAFF	Symbols and Pictographs Extended-A
10300 — 1032F	Old Italic	1FB00–1FBFF	Symbols for Legacy Computing

$ Unicode Character Ranges / 0x0....

Print them sequentially or grep the range against your wordlists.



```
0x0 NUL
0x1 SOH
0x2 STX
0x3 ETX
0x4 EOT
0x5 ENQ
0x6 ACK
0x7 BEL
0x8 BS
0x9
0xa

0xb VT
0xc FF
0xd
0xe SO
0xf SI
0x10 DLE
0x11 DC1
0x12 DC2
0x13 DC3
0x14 DC4
0x15 NAK
0x16 SYN
0x17 ETB
0x18 CAN
0x19 EM
0x1a SUB
0x1b ESC
0x1c FS
0x1d GS
0x1e RS
0x1f US
```

```
0x20
0x21 !
0x22 "
0x23 #
0x24 $
0x25 %
0x26 &
0x27 '
0x28 (
0x29 )
0x2a *
0x2b +
0x2c ,
0x2d -
0x2e .
0x2f /
0x30 0
0x31 1
0x32 2
0x33 3
0x34 4
0x35 5
0x36 6
0x37 7
0x38 8
0x39 9
0x3a :
0x3b ;
0x3c <
0x3d =
0x3e >
```

```
0x1f600 😀
0x1f601 😁
0x1f602 😂
0x1f603 😃
0x1f604 😄
0x1f605 😅
0x1f606 😆
0x1f607 😇
0x1f608 😈
0x1f609 😉
0x1f60a 😊
0x1f60b 😋
0x1f60c 😌
0x1f60d 😍
0x1f60e 😎
0x1f60f 😏
0x1f610 😐
0x1f611 😑
0x1f612 😒
0x1f613 😓
0x1f614 😔
0x1f615 😕
0x1f616 😖
0x1f617 😗
0x1f618 😘
0x1f619 😙
```

```
0x1f701 🜁
0x1f702 🜂
0x1f703 🜃
0x1f704 🜄
0x1f705 🜅
0x1f706 🜆
0x1f707 🜇
0x1f708 🜈
0x1f709 🜉
0x1f70a 🜊
0x1f70b 🜋
0x1f70c 🜌
0x1f70d 🜍
0x1f70e 🜎
0x1f70f 🜏
0x1f710 🜐
0x1f711 🜑
0x1f712 🜒
0x1f713 🜓
0x1f714 🜔
0x1f715 🜕
0x1f716 🜖
0x1f717 🜗
0x1f718 🜘
0x1f719 🜙
0x1f71a 🜚
```

```
0x1f914 🤔
0x1f915 🤕
0x1f916 🤖
0x1f917 🤗
0x1f918 🤘
0x1f919 🤙
0x1f91a 🤚
0x1f91b 🤛
0x1f91c 🤜
0x1f91d 🤝
0x1f91e 🤞
0x1f91f 🤟
0x1f920 🤠
0x1f921 🤡
0x1f922 🤢
0x1f923 🤣
0x1f924 🤤
0x1f925 🤥
0x1f926 🤦
0x1f927 🤧
0x1f928 🤨
0x1f929 🤩
0x1f92a 🤪
0x1f92b 🤫
0x1f92c 🤬
```

```
0x5d0 א
0x5d1 ב
0x5d2 ג
0x5d3 ד
0x5d4 ה
0x5d5 ו
0x5d6 ז
0x5d7 ח
0x5d8 ט
0x5d9 י
0x5da ך
0x5db כ
0x5dc ל
0x5dd ם
0x5de מ
0x5df ן
0x5e0 נ
0x5e1 ס
0x5e2 ע
0x5e3 ף
0x5e4 פ
0x5e5 ץ
0x5e6 צ
0x5e7 ק
0x5e8 ר
0x5e9 ש
0x5ea ת
```

```
0xa000 ꀀ
0xa001 ꀁ
0xa002 ꀂ
0xa003 ꀃ
0xa004 ꀄ
0xa005 ꀅ
0xa006 ꀆ
0xa007 ꀇ
0xa008 ꀈ
0xa009 ꀉ
0xa00a ꀊ
0xa00b ꀋ
0xa00c ꀌ
0xa00d ꀍ
0xa00e ꀎ
0xa00f ꀏ
0xa010 ꀐ
0xa011 ꀑ
0xa012 ꀒ
0xa013 ꀓ
0xa014 ꀔ
0xa015 ꀕ
0xa016 ꀖ
0xa017 ꀗ
```

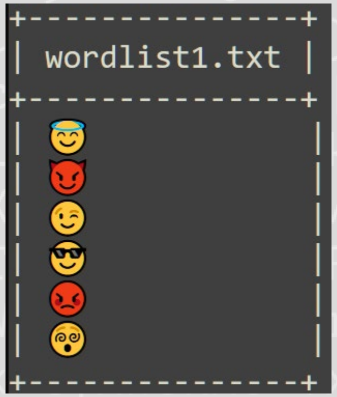
```
start_code_point="1F300"; end_code_point="1F5FF"; for ((i=16#$start_code_point; i<=16#$end_code_point; i++)); do printf "\U$(printf '%x' $i)"; done
```

```
grep -P -I "[\x{0000}-\x{007F}]" wordlist.txt > Basic_Latin.txt
grep -P -I "[\x{0250}-\x{02AF}]" wordlist.txt > IPA_Extensions.txt
grep -P -I "[\x{02B0}-\x{02FF}]" wordlist.txt > Spacing_Modifier_Letters.txt
grep -P -I "[\x{0300}-\x{036F}]" wordlist.txt > Combining_Diacritical_Marks.txt
grep -P -I "[\x{0370}-\x{03FF}]" wordlist.txt > Greek_and_Coptic.txt
grep -P -I "[\x{0400}-\x{04FF}]" wordlist.txt > Cyrillic.txt
grep -P -I "[\x{0530}-\x{058F}]" wordlist.txt > Armenian.txt
grep -P -I "[\x{0590}-\x{05FF}]" wordlist.txt > Hebrew.txt
grep -P -I "[\x{0600}-\x{06FF}]" wordlist.txt > Arabic.txt
………………
………………
grep -P -I "[\x{1F600}-\x{1F64F}]" wordlist.txt > Emoticons.txt
grep -P -I "[\x{1F680}-\x{1F6FF}]" wordlist.txt > Transport_and_Map_Symbols.txt
grep -P -I "[\x{1F700}-\x{1F77F}]" wordlist.txt > Alchemical_Symbols.txt
grep -P -I "[\x{1F780}-\x{1F7FF}]" wordlist.txt > Geometric_Shapes_Extended.txt
grep -P -I "[\x{1FA00}-\x{1FA6F}]" wordlist.txt > Chess_Symbols.txt
grep -P -I "[\x{1FB00}-\x{1FBFF}]" wordlist.txt > Symbols_for_Legacy_Computing.txt
grep -P -I "[\x{E0000}-\x{E007F}]" wordlist.txt > Tags.txt
```

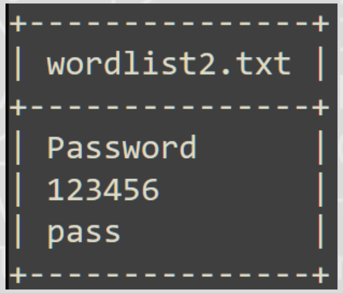
$ Unicode Character Ranges / how to handle

Attack Ideas

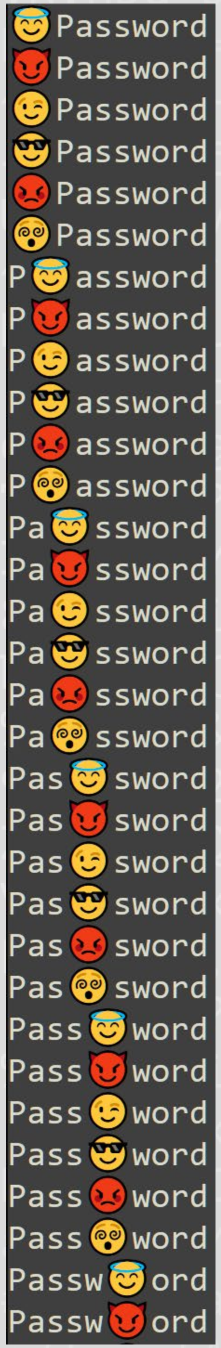
- Combinator/shuffling attack (-a 1) between two different blocks
  - -a 1 Greek.txt Emoticons.txt
- hashcat-utils
  - combinator , combinator3 Wancho.txt Emoticons.txt Kawi.txt | hashcat …
  - Python script every_day_im_shufflin.py



```
+---------------+
| wordlist1.txt |
+---------------+
| 😇            |
| 😈            |
| 😉            |
| 😎            |
| 😡            |
| 😵‍💫            |
+---------------+
```



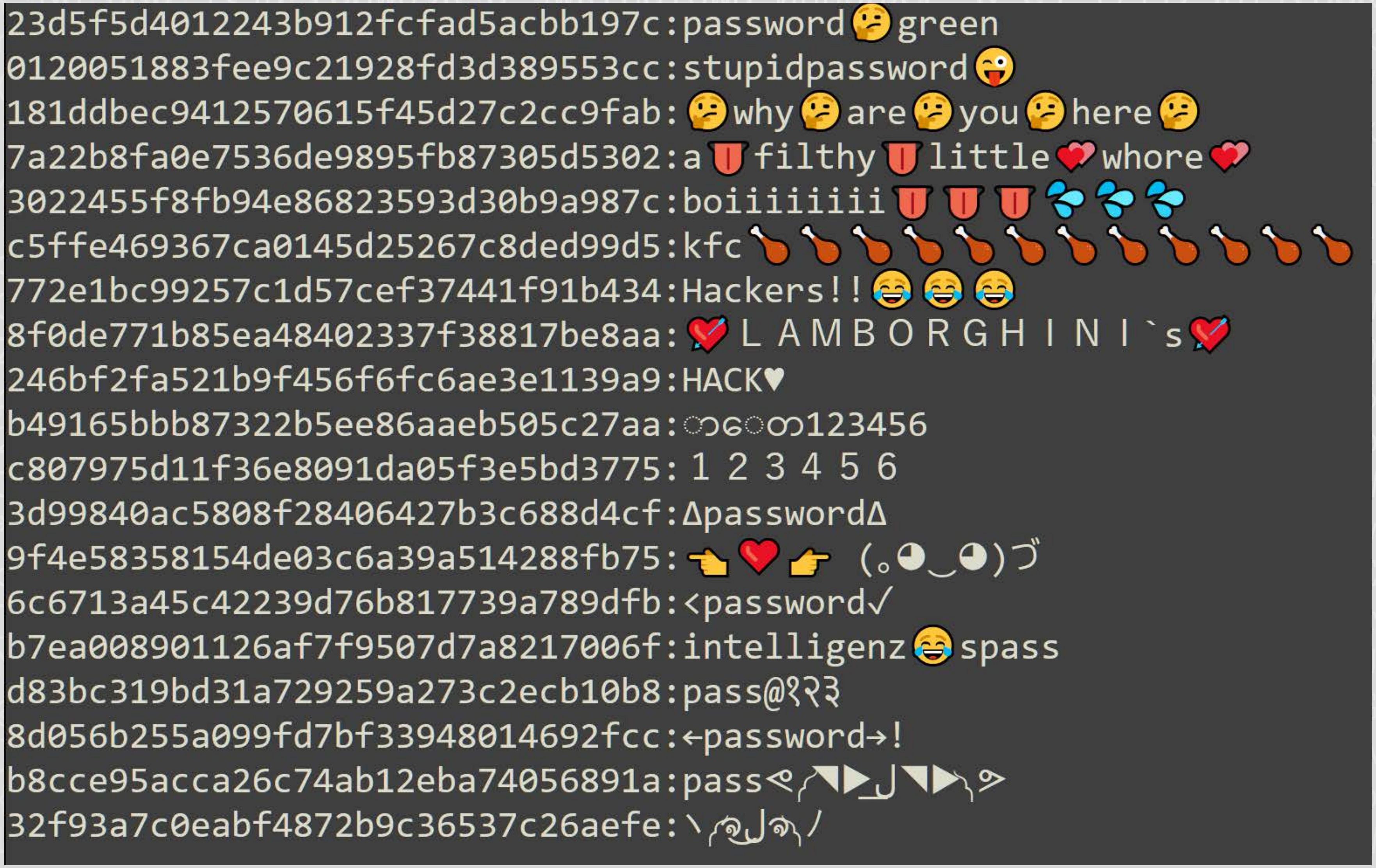
```
+---------------+
| wordlist2.txt |
+---------------+
| Password      |
| 123456        |
| pass          |
+---------------+
```



```
😇Password
😈Password
😉Password
😎Password
😡Password
😵‍💫Password
P😇assword
P😈assword
P😉assword
P😎assword
P😡assword
P😵‍💫assword
Pa😇ssword
Pa😈ssword
Pa😉ssword
Pa😎ssword
Pa😡ssword
Pa😵‍💫ssword
Pas😇sword
Pas😈sword
Pas😉sword
Pas😎sword
Pas😡sword
Pas😵‍💫sword
Pass😇word
Pass😈word
Pass😉word
Pass😎word
Pass😡word
Pass😵‍💫word
Passw😇ord
Passw😈ord
```

Does anyone ACTUALLY using characters from different Unicode blocks to generate passwords ? Or is it an empty rabbit hole ?

Not an empty rabbit hole…



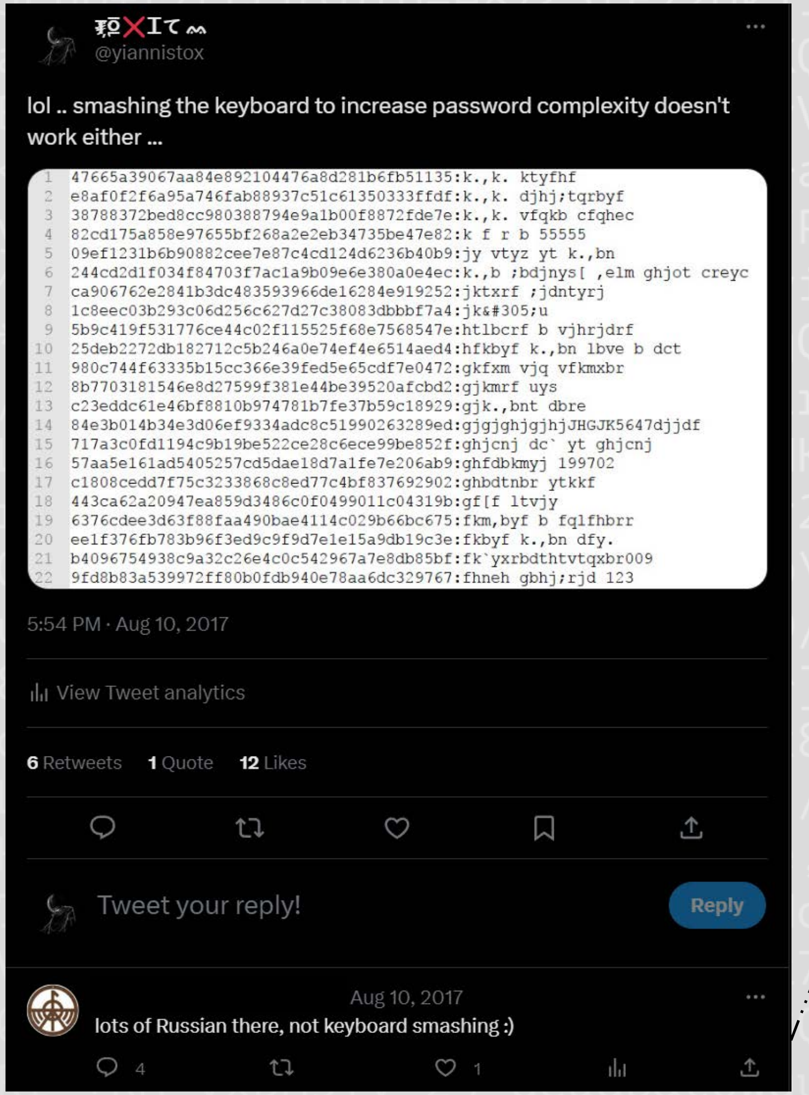
```
23d5f5d4012243b912fcfad5acbb197c:password🤔green
0120051883fee9c21928fd3d389553cc:stupidpassword😜
181ddbec9412570615f45d27c2cc9fab:🤔why🤔are🤔you🤔here🤔
7a22b8fa0e7536de9895fb87305d5302:a👅filthy👅little💔whore💔
3022455f8fb94e86823593d30b9a987c:boiiiiiiii👅👅👅💦💦💦
c5ffe469367ca0145d25267c8ded99d5:kfc🍗🍗🍗🍗🍗🍗🍗🍗🍗🍗🍗🍗
772e1bc99257c1d57cef37441f91b434:Hackers!!😂😂😂
8f0de771b85ea48402337f38817be8aa:💘L A M B O R G H I N I `s💘
246bf2fa521b9f456f6fc6ae3e1139a9:HACK♥
b49165bbb87322b5ee86aaeb505c27aa:ာ၆ိတ123456
c807975d11f36e8091da05f3e5bd3775: 1 2 3 4 5 6
3d99840ac5808f28406427b3c688d4cf:ΔpasswordΔ
9f4e58358154de03c6a39a514288fb75:👈❤👉 (｡◕‿◕)づ
6c6713a45c42239d76b817739a789dfb:<password✓
b7ea008901126af7f9507d7a8217006f:intelligenz😂spass
d83bc319bd31a729259a273c2ecb10b8:pass@१२३
8d056b255a099fd7bf33948014692fcc:←password→!
b8cce95acca26c74ab12eba74056891a:pass੭╭◥▶ل◥▶╮୨
32f93a7c0eabf4872b9c36537c26aefe:\╭ര͜لര╮/
```

$ agenda

we are here

$ charset / transliteration (translit)

Some accidental crack of some hashes a long, long time ago … (2017!)



[illegible]
@yiannistox

lol .. smashing the keyboard to increase password complexity doesn't work either ...

```
47665a39067aa84e892104476a8d281b6fb51135:k.,k. ktyfhf
e8af0f2f6a95a746fab88937c51c61350333ffdf:k.,k. djhj;tqrbyf
38788372bed8cc980388794e9a1b00f8872fde7e:k.,k. vfqkb cfqhec
82cd175a858e97655bf268a2e2eb34735be47e82:k f r b 55555
09ef1231b6b90882cee7e87c4cd124d6236b40b9:jy vtyz yt k.,bn
244cd2d1f034f84703f7ac1a9b09e6e380a0e4ec:k.,b ;bdjnys[ ,elm ghjot creyc
ca906762e2841b3dc483593966de16284e919252:jktxrf ;jdntyrj
1c8eec03b293c06d256c627d27c38083dbbbf7a4:jk&#305;u
5b9c419f531776ce44c02f115525f68e7568547e:htlbcrf b vjhrjdrf
25deb2272db182712c5b246a0e74ef4e6514aed4:hfkbyf k.,bn lbve b dct
980c744f63335b15cc366e39fed5e65cdf7e0472:gkfxm vjq vfkmxbr
8b7703181546e8d27599f381e44be39520afcbd2:gjkmrf uys
c23eddc61e46bf8810b974781b7fe37b59c18929:gjk.,bnt dbre
84e3b014b34e3d06ef9334adc8c51990263289ed:gjgjghjgjhjJHGJK5647djjdf
717a3c0fd1194c9b19be522ce28c6ece99be852f:ghjcnj dc` yt ghjcnj
57aa5e161ad5405257cd5dae18d7a1fe7e206ab9:ghfdbkmyj 199702
c1808cedd7f75c3233868c8ed77c4bf837692902:ghbdtnbr ytkkf
443ca62a20947ea859d3486c0f0499011c04319b:gf[f ltvjy
6376cdee3d63f88faa490bae4114c029b66bc675:fkm,byf b fqlfhbrr
ee1f376fb783b96f3ed9c9f9d7e1e15a9db19c3e:fkbyf k.,bn dfy.
b4096754938c9a32c26e4c0c542967a7e8db85bf:fk`yxrbdthtvtqxbr009
9fd8b83a539972ff80b0fdb940e78aa6dc329767:fhneh gbhj;rjd 123
```

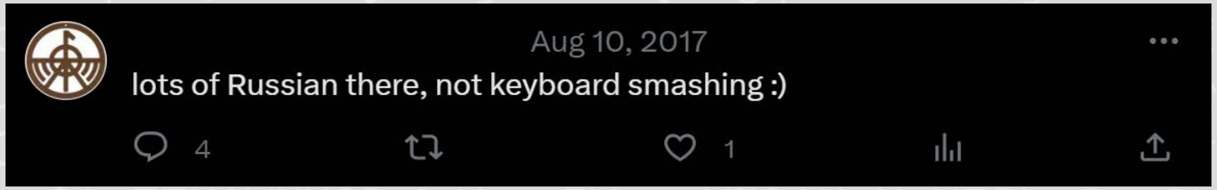
5:54 PM · Aug 10, 2017

View Tweet analytics

6 Retweets **1** Quote **12** Likes

Tweet your reply! Reply

Aug 10, 2017
lots of Russian there, not keyboard smashing :)
4 1

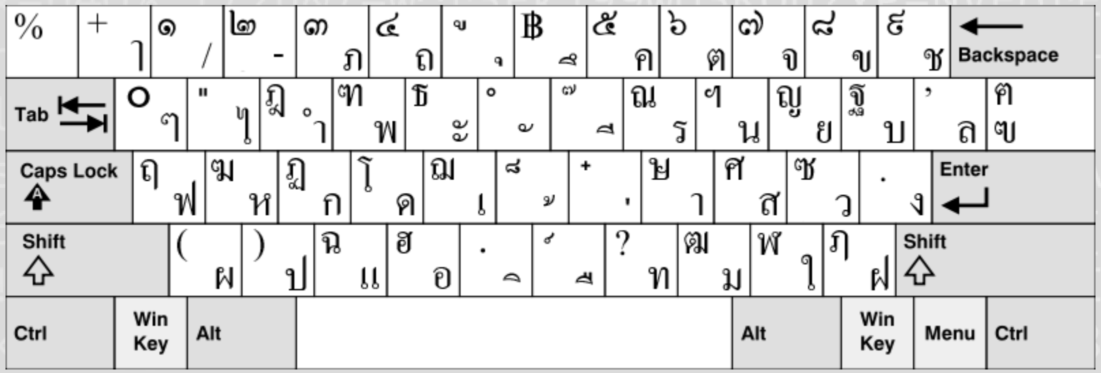


Aug 10, 2017
lots of Russian there, not keyboard smashing :)
4 1

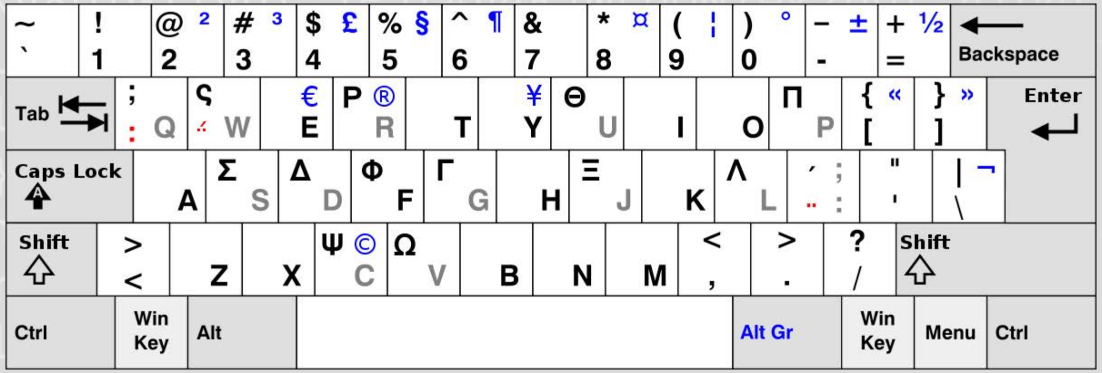
$ charset / transliteration / keyboards

Transliteration is the process of transferring a word from the alphabet of one language to another.
Transliteration helps people pronounce words and names in foreign languages.

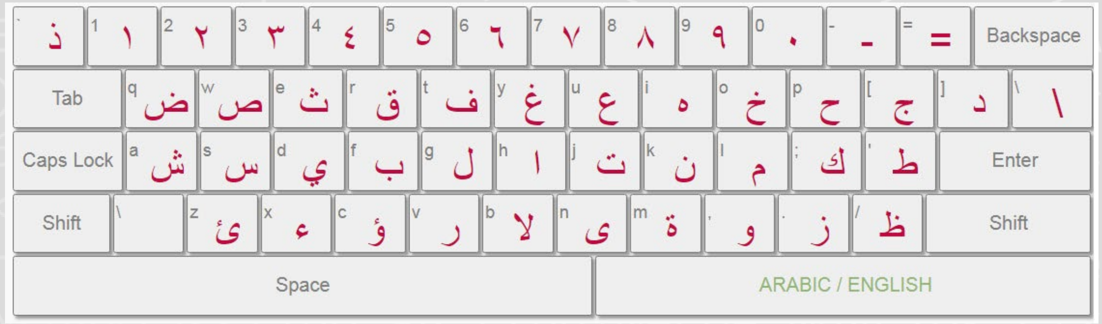
Thai



Greek

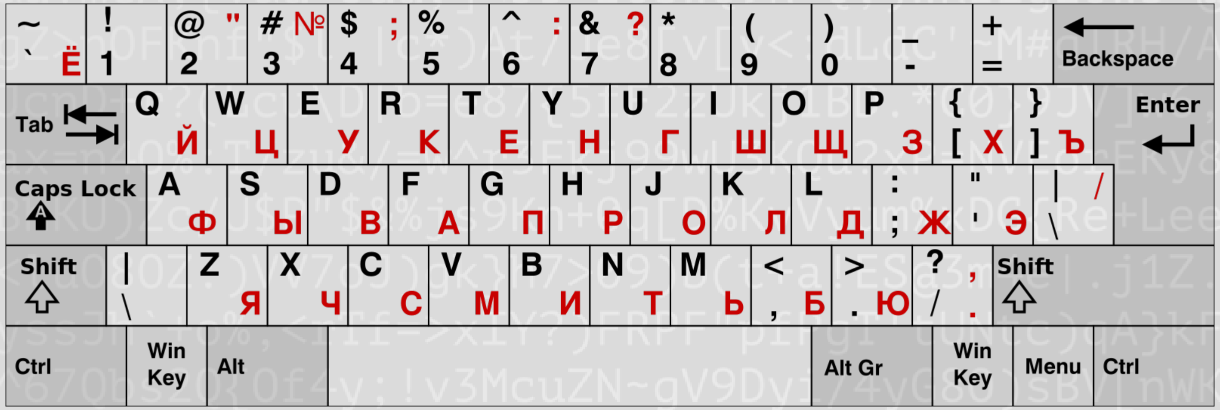


Arabic



Cyrillic

$ charset / translit / Cyrillic


~ ` Ё ! 1 @ " 2 # № 3 $; 4 % 5 ^ : 6 & ? 7 * 8 (9) 0 _ - + = Backspace
Tab Q Й W Ц E У R К T Е Y Н U Г I Ш O Щ P З { [Х }] Ъ Enter
Caps Lock A Ф S Ы D В F А G П H Р J О K Л L Д : ; Ж " ' Э | / \
Shift | \ Z Я X Ч C С V М B И N Т M Ь < , Б > . Ю ? / , . Shift
Ctrl Win Key Alt Alt Gr Win Key Menu Ctrl

cyrillic	translit	translate
пароль	gfhjkm	password
привет	ghbdtn	Hello
кактус	rfrnec	cactus
малинка	vfkbyrf	raspberry
любовь	k.,jdm	Love
вампир	Dfvgbh	a vampire

The password that is hashed is the one in the translit format!

mapping	
й	Q
ц	w
у	e
к	r
е	t
н	y
г	u
ш	i
щ	o
з	p
х	[
ъ	]
ф	a
ы	s
в	d
а	f
п	g
р	h
о	j
л	k
д	l
ж	;
э	'
ё	\
я	z
ч	x
с	c
м	v
и	b
т	n
ь	m
б	,
ю	.

$ charset / translit

How ?

Preparation

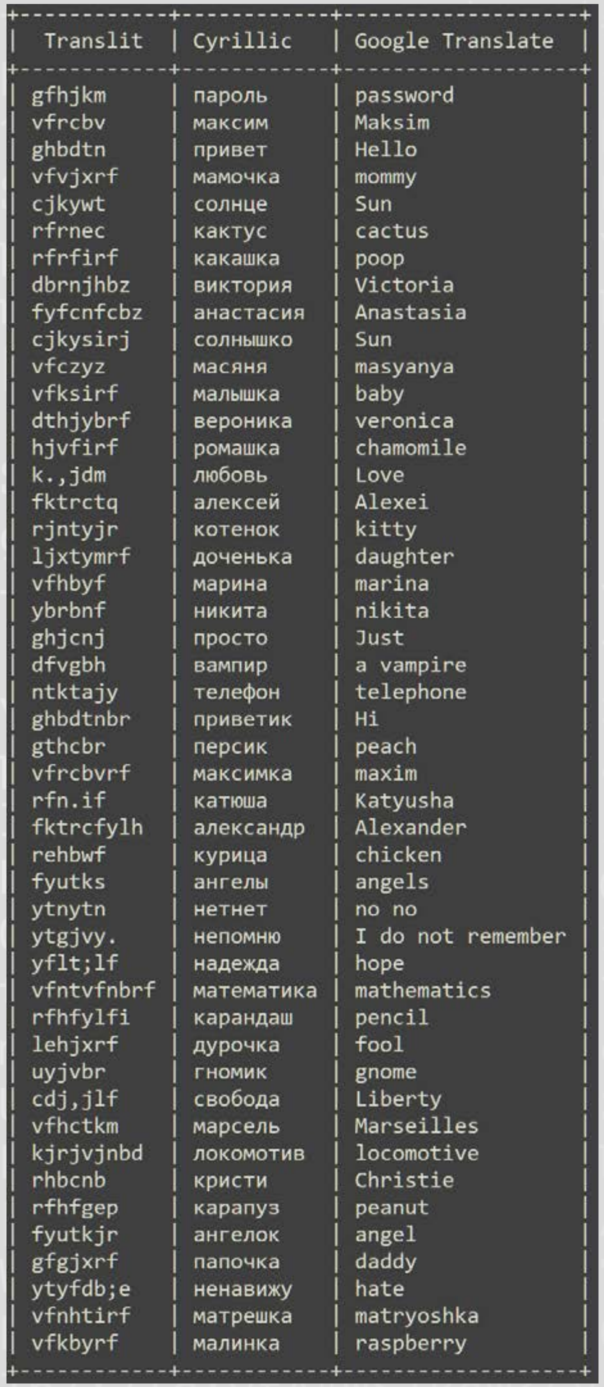
1) Create a wordlist with common Cyrillic words or extract all the words matching characters from the Cyrillic character range from your lists

```
grep -P -I "[\x{0400}-\x{04FF}]" wordlist* > Cyrillic.txt
```

2) Replace characters from the mapping table and generate a translit_wordlist.txt

3) Remove [:digit:] & [:punct:] and sort uniquely to get the root words

4)* get some stats just because



```
+-------------+-------------+--------------------+
|  Translit   | Cyrillic    | Google Translate   |
+-------------+-------------+--------------------+
| gfhjkm      | пароль      | password           |
| vfrcbv      | максим      | Maksim             |
| ghbdtn      | привет      | Hello              |
| vfvjxrf     | мамочка     | mommy              |
| cjkywt      | солнце      | Sun                |
| rfrnec      | кактус      | cactus             |
| rfrfirf     | какашка     | poop               |
| dbrnjhbz    | виктория    | Victoria           |
| fyfcnfcbz   | анастасия   | Anastasia          |
| cjkysirj    | солнышко    | Sun                |
| vfczyz      | масяня      | masyanya           |
| vfksirf     | малышка     | baby               |
| dthjybrf    | вероника    | veronica           |
| hjvfirf     | ромашка     | chamomile          |
| k.,jdm      | любовь      | Love               |
| fktrctq     | алексей     | Alexei             |
| rjntyjr     | котенок     | kitty              |
| ljxtymrf    | доченька    | daughter           |
| vfhbyf      | марина      | marina             |
| ybrbnf      | никита      | nikita             |
| ghjcnj      | просто      | Just               |
| dfvgbh      | вампир      | a vampire          |
| ntktajy     | телефон     | telephone          |
| ghbdtnbr    | приветик    | Hi                 |
| gthcbr      | персик      | peach              |
| vfrcbvrf    | максимка    | maxim              |
| rfn.if      | катюша      | Katyusha           |
| fktrcfylh   | александр   | Alexander          |
| rehbwf      | курица      | chicken            |
| fyutks      | ангелы      | angels             |
| ytnytn      | нетнет      | no no              |
| ytgjvy.     | непомню     | I do not remember  |
| yflt;lf     | надежда     | hope               |
| vfntvfnbrf  | математика  | mathematics        |
| rfhfylfi    | карандаш    | pencil             |
| lehjxrf     | дурочка     | fool               |
| uyjvbr      | гномик      | gnome              |
| cdj,jlf     | свобода     | Liberty            |
| vfhctkm     | марсель     | Marseilles         |
| kjrjvjnbd   | локомотив   | locomotive         |
| rhbcnb      | кристи      | Christie           |
| rfhfgep     | карапуз     | peanut             |
| fyutkjr     | ангелок     | angel              |
| gfgjxrf     | папочка     | daddy              |
| ytyfdb;e    | ненавижу    | hate               |
| vfnhtirf    | матрешка    | matryoshka         |
| vfkbyrf     | малинка     | raspberry          |
+-------------+-------------+--------------------+
```

$ charset / translit

Attack Ideas

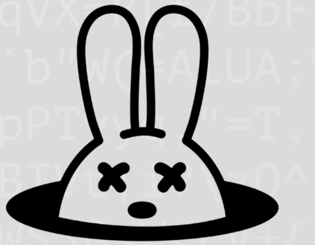
- Straight translit_wordlist.txt –r rules (don’t forget to debug!)
  - translit_wordlist.txt –r best64.rules
- Combinator attack (-a 1)
  - -a 1 translit_wordlist.txt translit_wordlist.txt
- “**tmesis.pl**” the **root translit words** to generate **insert rules** and use them with good wordlists.
  - Tmesis.pl < translit_root > translit_insert_rules
  - translit_wordlist.txt –r translit_insert_rules

tmesis.pl

1. Input wordlist contains one word: “password”
2. Destination wordlist contains one word: “123456”
3. Tmesis will make Hashcat rules that insert “password” at each possible position within “123456” and this will result in the following password candidate words:

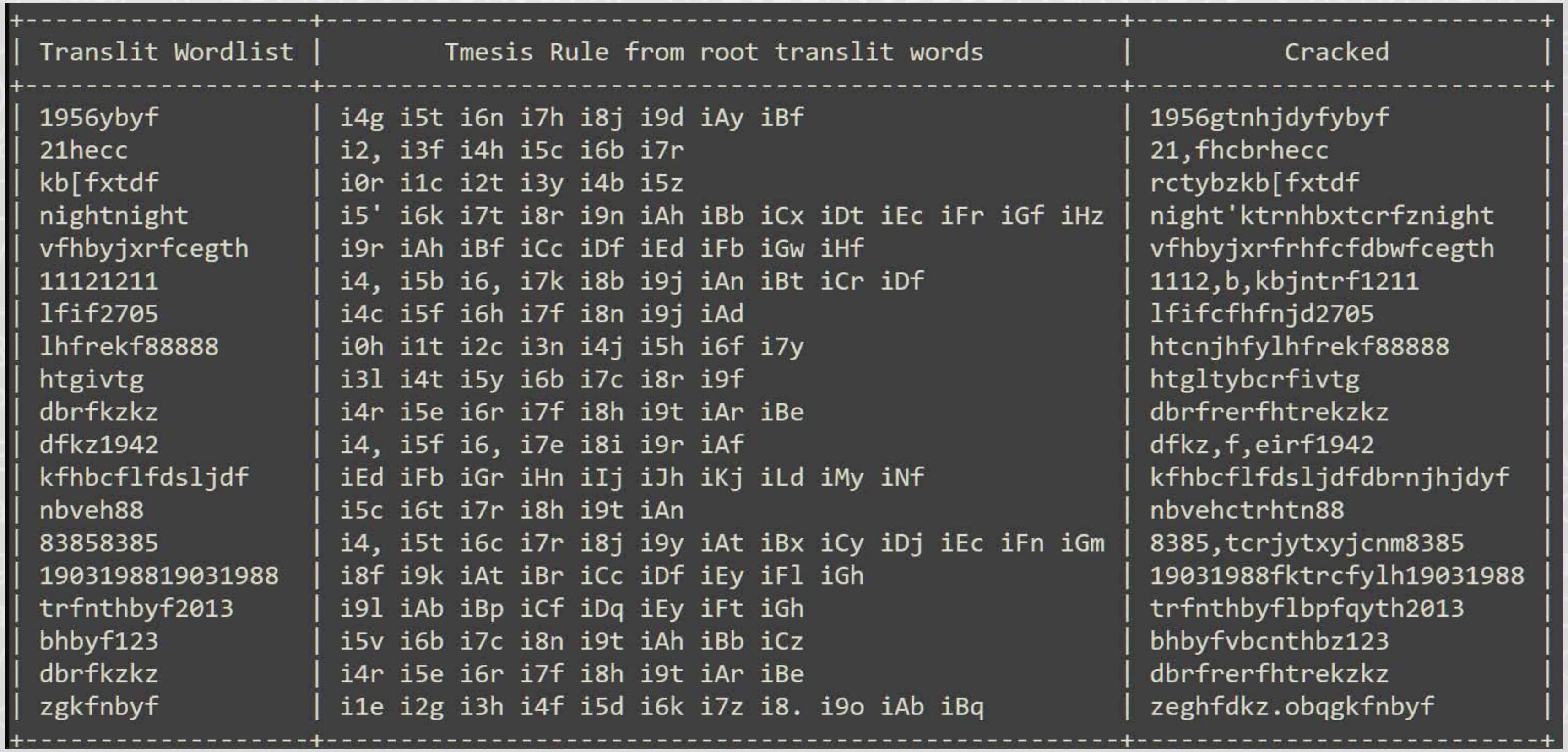
password123456
1**password**23456
12**password**3456
123**password**456
1234**password**56
12345**password**6
123456**password**

Empty rabbit hole or rabbit hole with a white rabbit … ?

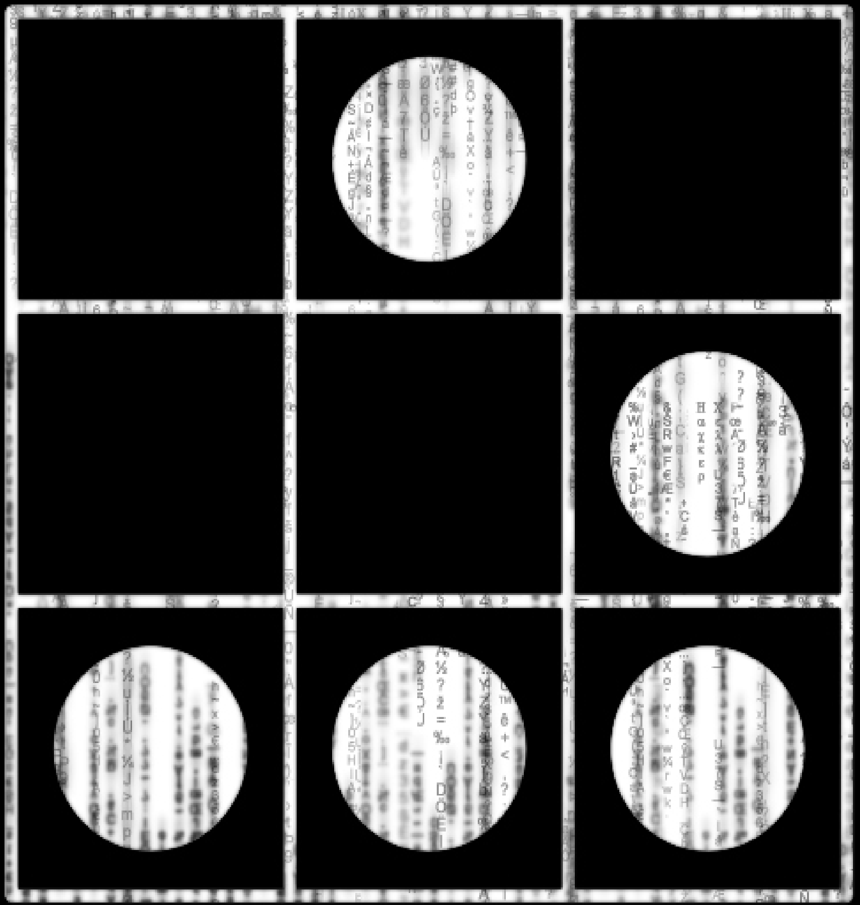


$ charset -> translit -> tmesis -> results!

```
+----------------------------------------------------------------------------------------------------------+
|  ./hashcat –m 99999 wordlist_to_match wordlist.txt –r rules.rule --debug-mode=4 --debug-file=debug.log  |
+----------------------------------------------------------------------------------------------------------+
```



```
+-------------------+-----------------------------------------------------+------------------------------+
| Translit Wordlist |         Tmesis Rule from root translit words        |            Cracked           |
+-------------------+-----------------------------------------------------+------------------------------+
| 1956ybyf          | i4g i5t i6n i7h i8j i9d iAy iBf                     | 1956gtnhjdyfybyf             |
| 21hecc            | i2, i3f i4h i5c i6b i7r                             | 21,fhcbrhecc                 |
| kb[fxtdf          | i0r i1c i2t i3y i4b i5z                             | rctybzkb[fxtdf               |
| nightnight        | i5' i6k i7t i8r i9n iAh iBb iCx iDt iEc iFr iGf iHz | night'ktrnhbxtcrfznight      |
| vfhbyjxrfcegth    | i9r iAh iBf iCc iDf iEd iFb iGw iHf                 | vfhbyjxrfrhfcfdbwfcegth      |
| 11121211          | i4, i5b i6, i7k i8b i9j iAn iBt iCr iDf             | 1112,b,kbjntrf1211           |
| lfif2705          | i4c i5f i6h i7f i8n i9j iAd                         | lfifcfhfnjd2705              |
| lhfrekf88888      | i0h i1t i2c i3n i4j i5h i6f i7y                     | htcnjhfylhfrekf88888         |
| htgivtg           | i3l i4t i5y i6b i7c i8r i9f                         | htgltybcrfivtg               |
| dbrfkzkz          | i4r i5e i6r i7f i8h i9t iAr iBe                     | dbrfrerfhtrekzkz             |
| dfkz1942          | i4, i5f i6, i7e i8i i9r iAf                         | dfkz,f,eirf1942              |
| kfhbcflfdsljdf    | iEd iFb iGr iHn iIj iJh iKj iLd iMy iNf             | kfhbcflfdsljdfdbrnjhjdyf     |
| nbveh88           | i5c i6t i7r i8h i9t iAn                             | nbvehctrhtn88                |
| 83858385          | i4, i5t i6c i7r i8j i9y iAt iBx iCy iDj iEc iFn iGm | 8385,tcrjytxyjcnm8385        |
| 1903198819031988  | i8f i9k iAt iBr iCc iDf iEy iFl iGh                 | 19031988fktrcfylh19031988    |
| trfnthbyf2013     | i9l iAb iBp iCf iDq iEy iFt iGh                     | trfnthbyflbpfqyth2013        |
| bhbyf123          | i5v i6b i7c i8n i9t iAh iBb iCz                     | bhbyfvbcnthbz123             |
| dbrfkzkz          | i4r i5e i6r i7f i8h i9t iAr iBe                     | dbrfrerfhtrekzkz             |
| zgkfnbyf          | i1e i2g i3h i4f i5d i6k i7z i8. i9o iAb iBq         | zeghfdkz.obqgkfnbyf          |
+-------------------+-----------------------------------------------------+------------------------------+
```



Questions?

- @yiannistox

atom
blandyuk
Chick3nman
coolbry95
dropdead
epixoip
EvilMog
Hydraze
K9
kontrast23
Kryczek
legion
m3g9tr0n
matrix
Minga
N|IGHT5
_NSAKEY
philsmd
purehate
radix
rurapenthe
The_Mechanic
T0XlC
TychoTithonus
unix-ninja
Xanadrel
xmisery