comboleetor.pl - A Password List Generator

-~+~-

Jim Brown

jpb@jimby.name

comboleetor.pl

- The problem:
  - You need to generate a comprehensive password list using known text elements, numbers, and punctuation, possibly with added “leetspeak” - i.e.

    +3><+ (“text”)

    |=/2()|\/| (“from”)

    7#3 (“the”)

    |-|@K3r$ (“hackers”)

comboleetor.pl

- comboleetor.pl combines text blocks, numbers, and punctuation in any order determined by an input specification. The resulting output can be further subjected to 'leetspeak' substitutions and serial capitalization.

comboleetor.pl

Usage:

perl comboleetor.pl [-b blocksfile] [-n numbersfile] [-c] [-s] [-h | H]

- **where blocksfile and numbersfile contain text blocks and numbers respectively,**
- **default files are blocks.txt and numbers.txt in the current directory.**

-c Serialize capitalization throughout the output word
-s Print all hashes and exit
-h Print short help (this text)
-H Print longer help text with examples

comboleetor.pl

- The “blocks.txt” file contains words (or text fragments):

```
cat+
dog-
marsh
mellow
23skidoo
07/04/1976
correct
horse
battery
staple
```

comboleetor.pl

- The “numbers.txt” file contains numbers, lists, and/or a number format specification (loosely based on the perl sprintf function) that generates a list of numbers to use:

```
%03.3d

1,2,4,10-15,200-300

%05.5x

40-50

%d

19-25
```

comboleetor.pl

- The input specification allows for up to five characters of the following set in any order:

```
B – text block
b – text blocks subjected to leet substitutions
N - number elements
P - punctuation elements from the set
    !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

- The goal is to produce all possible combinations of these elements in the order they are specified. The specification of elements comes from the user on stdin via terminal, pipe, file, etc.

comboleetor.pl

Example 1: Basic Usage

- blocks.txt file:

```
tin
can
```

- numbers.txt file:

```
31,72,2600
5
```

---

echo ‘BB’ | perl comboleetor.pl 2> /dev/null

```
can
cantin
cancan
tin
tintin
tincan
```

comboleetor.pl

Example 1 (cont.)

echo ‘BN’ | perl comboleetor.pl 2> /dev/null

```
tin
tin5
tin31
tin72
tin2600
can
can5
can31
can72
can2600
```

comboleetor.pl

Example 1 (cont.)

echo ‘BPB’ | perl comboleetor.pl 2> /dev/null

```
can
can
can tin
can can
can!
can!tin
can!can
can"
can"tin
can"can
can#
can#tin
can#can
can$
can$tin
can$can
can%
can%tin
```

. . .

```
tin_
tin_tin
tin_can
tin`
tin`tin
tin`can
tin{
tin{tin
tin{can
tin|
tin|tin
tin|can
tin}
tin}tin
tin}can
tin~
tin~tin
tin~can
```

This example generated 200 lines of output.

comboleetor.pl

Example 1 (cont.)

echo ‘BNP’ | perl comboleetor.pl 2> /dev/null

```
tin                 can2600;
tin5                can2600=
tin5                can2600<
tin5!               can2600>
tin5"               can2600?
tin5#               can2600@
tin5$     . . .     can2600[
tin5%               can2600\
tin5&               can2600]
tin5'               can2600^
tin5(               can2600_
tin5)               can2600`
tin5*               can2600{
tin5+               can2600|
tin5,               can2600}
tin5-               can2600~
```

This example generated 274 lines of output.

comboleetor.pl

Example 2: Using number formats and number ranges

- blocks.txt file:

```
tin
can
```

- numbers.txt file:

```
%05.5d
20-30,90-120
```

---

echo ‘BN’ | perl comboleetor.pl 2> /dev/null

```
tin
tin00020
tin00021
tin00022
tin00023
tin00024   . . .
tin00025
tin00026
tin00027
tin00028
tin00029
tin00030
tin00115
tin00116
tin00117
tin00118
tin00119
tin00120   . . .
can
can00020
can00021
can00022
can00023
can00024
can00111
can00112
can00113
can00114
can00115
can00116
can00117
can00118
can00119
can00120
```

comboleetor.pl

Example 3: Combining numbers with text and punctuation

- blocks.txt file:

```
tin
can
```

- numbers.txt file:

```
%05.5x
10-20,50-1000
```

Note 'x' for hexadecimal

---

echo 'BPN' | perl comboleetor.pl 2> /dev/null

```
can
can
can 0000a
can 0000b
can 0000c
can 0000d
can 0000e
can 0000f
can 00010
can 00011
can 00012
can 00013
```

Note space is a punctuation character.

```
. . .
```

```
can~003e2
can~003e3
can~003e4
can~003e5
can~003e6
can~003e7
can~003e8
tin
tin
tin 0000a
tin 0000b
tin 0000c
```

```
. . .
```

```
can~003dd
can~003de
can~003df
can~003e0
can~003e1
can~003e2
can~003e3
can~003e4
can~003e5
can~003e6
can~003e7
can~003e8
```

63560 lines of output.

comboleetor.pl

Example 4: Using leetspeak with ‘b’

- blocks.txt file:

```
tin

can
```

- numbers.txt file:

```
%05.5x

10-20,50-1000
```

---

echo ‘b’ | perl comboleetor.pl 2> /dev/null

```
ti|\|       can         7in
+in         (4|\|       7!n
c@|\|       t!n         7!|\|
+!n         c/-\|\|     7i|\|
(@n         c4n
c/-\n       (a|\|
(4n         c@n
(/-\|\|     (/-\n
ca|\|       +!|\|
t!|\|       (@|\|
(an         +i|\|
c4|\|       tin
```

Note that some letters have multiple leetspeak representations. For example the letter ‘a’ is represented by:

```
a, 4, @, /-\
```

Each is used in turn.

comboleetor.pl

Example 5: Serial capitalization of text with '-c'

- blocks.txt file:

```
tin
can
```

- numbers.txt file:

```
%05.5x
10-20,50-1000
```

---

echo 'B' | perl comboleetor.pl -c 2> /dev/null

```
can         tIn
can         TIn
Can         tiN
cAn         TiN
CAn         tIN
caN         TIN
CaN
cAN
CAN
tin
tin
Tin
```

comboleetor.pl

Example 6: Combining leetspeak with serial capitalization

- blocks.txt file:

```
tin
can
```

- numbers.txt file:

```
%05.5x
10-20,50-1000
```

---

echo ‘b’ | perl comboleetor.pl -c 2> /dev/null

```
ca|\|       C/-\n       can
ca|\|       c/-\N       Can
Ca|\|       C/-\N       cAn
cA|\|       (an         CAn
CA|\|       (an         caN
ti|\|       (An
ti|\|       (aN
Ti|\|       (AN
tI|\|       +!n
TI|\|       +!n
c/-\n       +!N
c/-\n       can
```

Note that capitalization only applies to actual ASCII letters, not leetspeak representations.

`c → C, a → A, n → N`

104 lines of output.

comboleetor.pl

Example 7: Viewing the hashes with '-s'

- blocks.txt file:

```
tin
can
```

- numbers.txt file:

```
%05.5x
10-20
```

---

echo 'B' | perl comboleetor.pl -s 2> /dev/null

Prints out the contents of each hash table and exits.

- Blocks Hash
- Numbers Hash
- Main Leetz Hash (primary leetspeak hash)
- Alternate 1 Leetz Hash (secondary leetspeak hash)
- Alternate 2 Leetz Hash (tertiary leetspeak hash)
- Punctuation Hash
- Leethash Hash (result of hashing Blocks Hash)

comboleetor.pl

Example 8: Interactive use

- blocks.txt file:

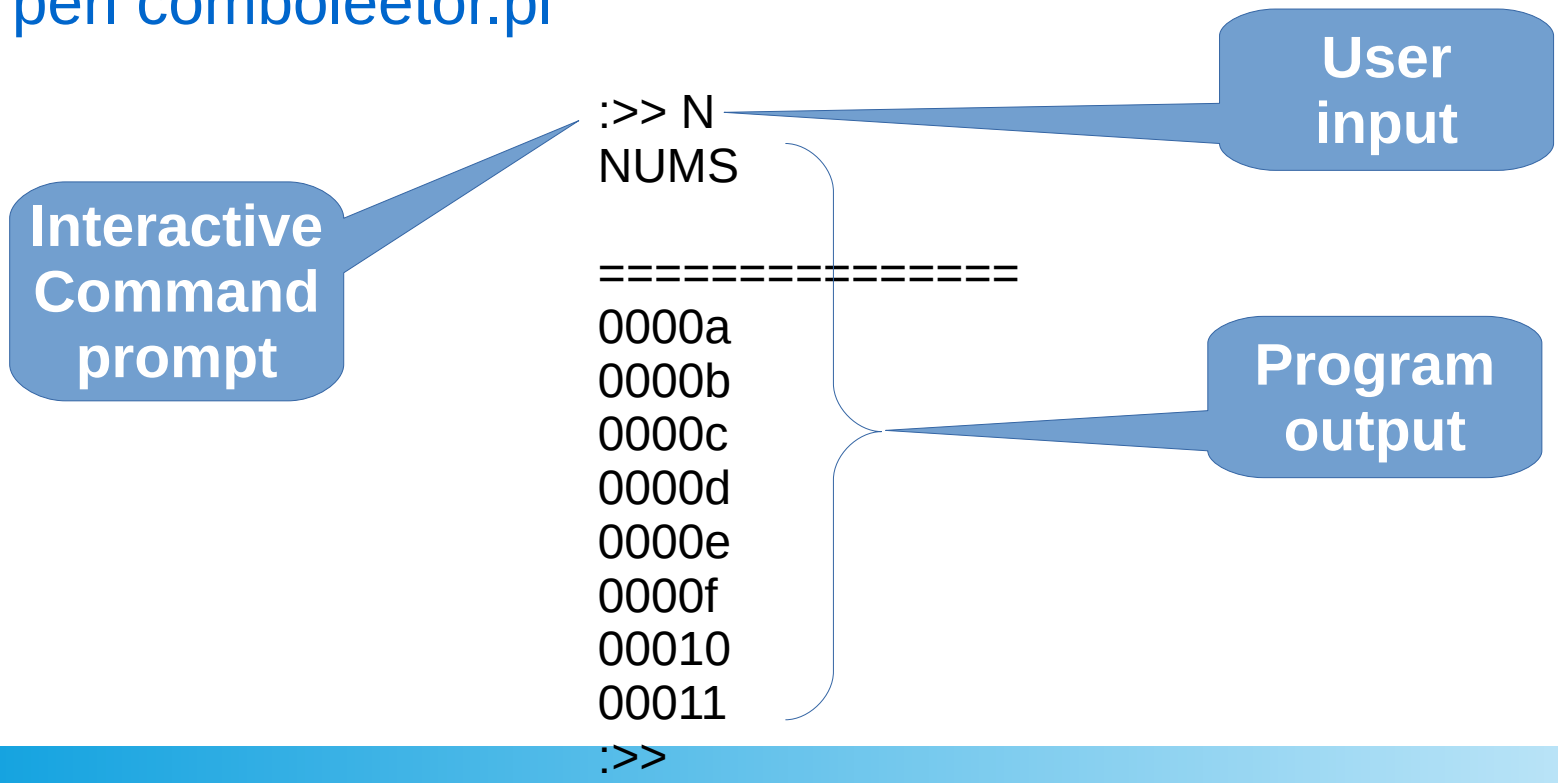
```
tin
can
```

- numbers.txt file:

```
%05.5x
10-17
```

---

perl comboleetor.pl



comboleetor.pl

Example 9: Tricks

- Suppose you only want a single character appended to a word as the basis for a text block:

- blocks.txt file:

```
tin+

can!
```

- You can use it in combination with the other examples:

echo ‘BB’ | perl comboleetor.pl -c 2> /dev/null

```
can!
can!tin+
can!can!
tin+
tin+tin+
tin+can!
```

comboleetor.pl

Example 9: Tricks (cont.)

- The more you know about how your target might use familiar objects (children's names, birthdays, planets, presidents, etc.) the closer you can narrow down a list to passwords you seriously want to check.

- blocks.txt file:

```
bob
08/16/1930
1930/08/16
carol
07/20/1938
1938/07/20
ted
08/29/1938
1938/08/29
alice
01/04/1937
1937/01/04
```

echo 'BB' | perl comboleetor.pl -c 2> /dev/null | wc

```
   Lines   Words   Bytes
    8102    8100   89366
```

echo 'BPB' | perl comboleetor.pl -c 2> /dev/null | wc

```
  Lines   Words  Bytes
  267459  275280 3217001
```

echo 'BPBPP' | perl comboleetor.pl -c 2> /dev/null | wc

```
   Lines      Words      Bytes
   289952744 307001240 4110760132
```

comboleetor.pl

Example 9: Tricks (cont.)

- blocks.txt file:

```
tin

can
```

- numbers.txt file:

```
%05.5x

10-20
```

---

comboleetor.pl can generate an ***enormous*** amount of passwords even from a simple set of blocks and numbers.

echo ‘BNBPP’ | perl comboleetor.pl -c 2> /dev/null | wc

```
Lines      Words      Bytes
4937020    5077692    68962770
```

comboleetor.pl

Download the code and try it out!

Get the code as a gzipped tarball from:

https://jimby.name/techbits/recent/comboleetor_2.0/comboleetor_2.0.tgz

Send feedback to jpb@jimby.name

Thanks!