crunch 3.6 Cheatsheet (ingenieriainformatica.uniovi.es)

Tool that generates wordlists from a character set

https://tools.kali.org/password-attacks/crunch

GENERAL USAGE

```
crunch <min-len> <max-len> [<charset string>] [options]
```

NOTES

This is a reference tool to generate wordlists to bruteforce passwords and other similar requirements. Crunch can create a wordlist based on criteria you specify. The output from crunch can be sent to the screen, file, or to another program

OPTIONS

Required parameters

charset string: You may specify character sets for crunch to use on the command line or if you leave it blank crunch will use the default character sets. The order MUST BE lower case characters, upper case characters, numbers, and then symbols. If you don't follow this order you will not get the results you want. You MUST specify either values for the character type or a plus sign. NOTE: If you want to include the space character in your character set you must escape it using the \ character or enclose your character set in quotes i.e. "abc ". See the examples 3, 11, 12, and 13 for examples.

max-len: The maximum length string you want crunch to end at. This option is required even for parameters that won't use the value.

min-len: The minimum length string you want crunch to start at. This option is required even for parameters that won't use the value.

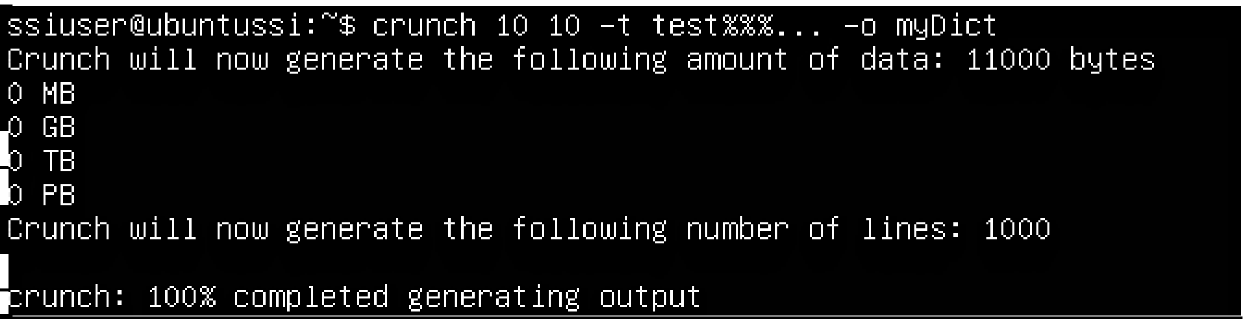
Options

-b number[type]: Specifies the size of the output file, only works if -o START is used, i.e.: 60MB The output files will be in the format of starting letter-ending letter for example: ./crunch 4 5 -b 20mib -o START will generate 4 files: aaaa-gvfed.txt, gvfee-ombqy.txt, ombqz-wcydt.txt, wcydu-zzzzz.txt valid values for type are kb, mb, gb, kib, mib, and gib. The first three types are based on 1000 while the last three types are based on 1024. NOTE There is no space between the number and type. For example 500mb is correct 500 mb is NOT correct.

-c number: Specifies the number of lines to write to output file, only works if -o START is used, i.e.: 60. The output files will be in the format of starting letter-ending letter for example: ./crunch 1 1 -f /pentest/password/crunch/charset.lst mixalpha-numeric-all-space -o START -c 60 will result in 2 files: a-7.txt and 8-\ .txt The reason for the slash in the second filename is the ending character is space and ls has to escape it to print it. Yes you will need to put in the \ when specifying the filename because the last character is a space.

-d numbersymbol: Limits the number of duplicate characters. -d 2@ limits the lower case alphabet to output like aab and aac. aaa would not be generated as that is 3 consecutive letters of a. The format is number then symbol where number is the maximum number of consecutive characters and symbol is the symbol of the the character set you want to limit i.e. @,%^ See examples 17-19.

-e string: Specifies when crunch should stop early



```
ssiuser@ubuntussi:~$ crunch 10 10 -t test%%%... -o myDict
Crunch will now generate the following amount of data: 11000 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 1000

crunch: 100% completed generating output
```

EXAMPLES

Example 1: crunch 1 8 - crunch will display a wordlist that starts at a and ends at zzzzzzzz

Example 2: crunch 1 6 abcdefg - crunch will display a wordlist using the character set abcdefg that starts at a and ends at gggggg

Example 3: crunch 1 6 abcdefg - there is a space at the end of the character string. In order for crunch to use the space you will need to escape it using the \ character. In this example you could also put quotes around the letters and not need the \, i.e. "abcdefg ". Crunch will display a wordlist using the character set abcdefg that starts at a and ends at (6 spaces)

Example 4: crunch 1 8 -f charset.lst mixalpha-numeric-all-space -o wordlist.txt - crunch will use the mixalpha-numeric-all-space character set from charset.lst and will write the wordlist to a file named wordlist.txt. The file will start with a and end with "   "

Example 5: crunch 8 8 -f charset.lst mixalpha-numeric-all-space -o wordlist.txt -t @@dog@@@ -s cbdogaaa - crunch should generate a 8 character wordlist using the mixalpha-number-all-space character set from charset.lst and will write the wordlist to a file named wordlist.txt. The file will start at cbdogaaa and end at " dog   "

Example 6: crunch 2 3 -f charset.lst ualpha -s BB - crunch with start generating a wordlist at BB and end with ZZZ. This is useful if you have to stop generating a wordlist in the middle. Just do a tail wordlist.txt and set the -s parameter to the next word in the sequence. Be sure to rename the original wordlist BEFORE you begin as crunch will overwrite the existing wordlist.

Example 7: crunch 4 5 -p abc - The numbers aren't processed but are needed. crunch will generate abc, acb, bac, bca, cab, cba.

Example 8: crunch 4 5 -p dog cat bird - The numbers aren't processed but are needed. crunch will generate birdcatdog, birddogcat, catbirddog, catdogbird, dogbirdcat, dogcatbird.

Example 9: crunch 1 5 -o START -c 6000 -z bzip2 - crunch will generate bzip2 compressed files with each file containing 6000 words. The filenames of the compressed files will be first_word-last_word.txt.bz2

Example 10: crunch 4 5 -b 20mib -o START - will generate 4 files: aaaa-gvfed.txt, gvfee-ombqy.txt, ombqz-wcydt.txt, wcydu-zzzzz.txt the first three files are 20MBs (real power of 2 MegaBytes) and the last file is 11MB.

Example 11: crunch 3 3 abc + 123 !@# -t @%^ - will generate a 3 character long word with a character as the first character, and number as the second character, and a symbol for the third character. The order in which you specify the characters you want is important. You must specify the order as lower case character, upper case character, number, and symbol. If you aren't going to use a particular character set you use a plus sign as a placeholder. As you can see I am not using the upper case character set so I am using the plus sign placeholder. The above will start at a1! and end at c3#

Example 12: crunch 3 3 abc + 123 !@# -t ^%@ - will generate 3 character words starting with !1a and ending with #3c

-f /path/to/charset.lst charset-name: Specifies a character set from the charset.lst

-i: Inverts the output so instead of aaa,aab,aac,aad, etc you get aaa,baa,caa,daa,aba,bba, etc

-l: When you use the -t option this option tells crunch which symbols should be treated as literals. This will allow you to use the placeholders as letters in the pattern. The -l option should be the same length as the -t option. See example 15.

-m: Merged with -p. Please use -p instead.

-o wordlist.txt: Specifies the file to write the output to, eg: wordlist.txt

-p charset OR -p word1 word2 …: Tells crunch to generate words that don't have repeating characters. By default crunch will generate a wordlist size of #of_chars_in_charset ^ max_length. This option will instead generate #of_chars_in_charset!. The ! stands for factorial. For example say the charset is abc and max length is 4.. Crunch will by default generate 3^4 = 81 words. This option will instead generate 3! = 3x2x1 = 6 words (abc, acb, bac, bca, cab, cba). THIS MUST BE THE LAST OPTION! This option CANNOT be used with -s and it ignores min and max length however you must still specify two numbers.

-q filename.txt: Tells crunch to read filename.txt and permute what is read. This is like the -p option except it gets the input from filename.txt.

-r: Tells crunch to resume generate words from where it left off. -r only works if you use -o. You must use the same command as the original command used to generate the words. The only exception to this is the -s option. If your original command used the -s option you MUST remove it before you resume the session. Just add -r to the end of the original command.

-s startblock: Specifies a starting string, eg: 03god22fs

-t @,%^: Specifies a pattern, eg: @@god@@@@ where the only the @'s, ,'s, %'s, and ^'s will change; @ will insert lower case characters; , will insert upper case characters; % will insert numbers; ^ will insert symbols

-u: The -u option disables the printpercentage thread. This should be the last option.

-z gzip, bzip2, lzma, and 7z: Compresses the output from the -o option. Valid parameters are gzip, bzip2, lzma, and 7z. gzip is the fastest but the compression is minimal. bzip2 is a little slower than gzip but has better compression. 7z is slowest but has the best compression.

by José Manuel Redondo López

Example 13: crunch 4 4 + + 123 + -t %%@^ - the plus sign (+) is a place holder so you can specify a character set for the character type. crunch will use the default character set for the character type when crunch encounters a + (plus sign) on the command line. You must either specify values for each character type or use the plus sign. I.E. if you have two characters types you MUST either specify values for each type or use a plus sign. So in this example the character sets will be:

```
       abcdefghijklmnopqrstuvwxyz
       ABCDEFGHIJKLMNOPQRSTUVWXYZ123
       !@#$%^&*()-_+=~`[]{}|\:;"'<>,.?/
```

there is a space at the end of the above string the output will start at 11a! and end at "33z ". The quotes show the space at the end of the string.

Example 14: crunch 5 5 -t ddd@@ -o j -p dog cat bird - any character other than one of the following: @,%^ is the placeholder for the words to permute. The @,%^ symbols have the same function as -t. If you want to use @,%^ in your output you can use the -l option to specify which character you want crunch to treat as a literal. So the results are

```
       birdcatdogaa
       birdcatdogab
       birdcatdogac
       <skipped>
       dogcatbirdzy
       dogcatbirdzz
```

Example 15: crunch 7 7 -t p@ss,%^ -l a@aaaaa - crunch will now treat the @ symbol as a literal character and not replace the character with a uppercase letter. this will generate

```
       p@ssA0!
       p@ssA0@
       p@ssA0#
       p@ssA0$
       <skipped>
       p@ssZ9
```

Example 16: crunch 5 5 -s @4#S2 -t @%^,2 -e @8 Q2 -l @dddd -b 10KB -o START - crunch will generate 5 character strings starting with @4#S2 and ending at @8 Q2. The output will be broken into 10KB sized files named for the files starting and ending strings.

Example 17: crunch 5 5 -d 2@ -t @@@%% - crunch will generate 5 character strings staring with aab00 and ending at zzy99. Notice that aaa and zzz are not present.

Example 18: crunch 10 10 -t @@@^%%%%^^ -d 2@ -d 3% -b 20mb -o START - crunch will generate 10 character strings starting with aab!0001!! and ending at zzy 9998. The output will be written to 20mb files.

Example 19: crunch 8 8 -d 2@ - crunch will generate 8 characters that limit the same number of lower case characters to 2. Crunch will start at aabaabaa and end at zzyzzyzz.

Example 20: crunch 4 4 -f unicode_test.lst japanese -t @@%% -l @xdd - crunch will load some Japanese characters from the unicode_test character set file. The output will start at @日00 and end at @語99.