Article
Dmitry Maslennikov · Dec 2, 2021 3m read

Open Exchange

How secure is password?

How to check if the password is strong enough, so it will not be cracked very fast? And how to make a strong password?

I've developed a tool that may help with this. You can find it on OpenExchange. Install it with zpm

```
zpm "install passwords-tool"
```

This module will install just one class caretdev.Passwords, which contains a few helpful methods in it

Secure Password

To get a secure password it's usually enough to use letters in upper and lower case, digits, and special symbols, and it should be at least 8 symbols long.

Method Generate with parameters

- Length - just a length of generating password, default value 12
- IncludeUpperLetter - Include Upper case ASCII letters, 2 if required, default 1
- IncludeLowerLetter - Include Lower case ASCII letters, 2 if required, default 2
- IncludeNumber - Include numbers, 2 if required, default 1
- IncludeSymbol - Include special symbols, 2 if required, default 1

```
USER>w ##class(caretdev.Passwords).Generate(12,1,0,0,0)
FMXRQEQPOVBC
USER>w ##class(caretdev.Passwords).Generate(12,1,1,0,0)
rgbPyWApcUjp
USER>w ##class(caretdev.Passwords).Generate(12,1,1,1,0)
cDuLf8FqEDx7
USER>w ##class(caretdev.Passwords).Generate(12,1,1,1,1)
0J/ lLbW|T$
USER>w ##class(caretdev.Passwords).Generate()
w3}{OQA|T{h^
```

Instead of ordinary $random, which may be not so secure for passwords, this method uses $System.Encryption.GenCryptRand(). In addition to get best passwords, it generates a few passwords in a loop, checks its entropy, and return one with a highest score.

Entropy

Password entropy predicts how difficult a given password would be to crack through guessing, brute force cracking, dictionary attacks or other common methods. Entropy essentially measures how many guesses an attacker will need to make to guess your password. And there are a few ways on how to calculate it.

```
USER>write ##class(caretdev.Passwords).Entropy("Pas$W0rD")
```

```
52.56
```

Entropy Formula
L = Password Length; Number of symbols in the password
S = Size of the pool of unique possible symbols (character set).

For example:
Numbers (0-9): 10
Lower Case Latin Alphabet (a-z): 26
Lower Case & Upper Case Latin Alphabet (a-z, A-Z): 52
ASCII Printable Character Set (a-z, A-Z, symbols, space): 95

Number of Possible Combinations = S**L

Entropy = log2(Number of Possible Combinations)

Shannon Entropy

```
USER>write ##class(caretdev.Passwords).ShannonScore("Pas$W0rD")
24
```

This way is based on the frequency of used characters, and the whole length of the password. Details in Wiki.

NIST Score

```
USER>write ##class(caretdev.Passwords).NISTScore("Pas$W0rD")
24
```

Calculation

- The entropy of the first character is four bits;
- The entropy of the next seven characters are two bits per character;
- The ninth through the twentieth character has 1.5 bits of entropy per character;
- Characters 21 and above have one bit of entropy per character.
- A "bonus" of six bits is added if both upper case letters and non-alphabetic characters are used.
- A "bonus" of six bits is added for passwords of length 1 through 19 characters following an extensive dictionary check to ensure the password is not contained within a large dictionary. Passwords of 20 characters or more do not receive this bonus because it is assumed they are pass-phrases consisting of multiple dictionary words.

Strength

```
write ##class(caretdev.Passwords).DetermineStrength("Pas$W0rD")
REASONABLE
```

And generated password

```
USER>write ##class(caretdev.Passwords).DetermineStrength(##class(caretdev.Passwords).
Generate())
STRONG
```

- VERYWEAK - Entropy <= 32
- WEAK - Entropy <= 48
- REASONABLE - Entropy <= 64
- STRONG - Entropy <= 80
- VERYSTRONG - Entropy > 80

If you like this project please cast your vote

#Security #InterSystems IRIS
Check the related application on InterSystems Open Exchange