

Forum:[Hack Forums](#)

Thread: [Secure Password Generator \[Re-vamped\] \[Updated 12/29/2016\]](#)

Malum™ - December 20, 2016, 4:28 am

I've had this project laying around awhile I made for a friend to further understand something (I don't recall anymore). Feel free to use this for account creators, login systems, etc. Ask any questions! I'm sorry I do not really comment my code.

GUI

Password Generator Class

Spoiler ([Click to View](#))

Code class PasswordGenerator

```
{
    private SecureRandom RND = new SecureRandom();

    public bool Special { get; set; }
    public bool Numbers { get; set; }
    public bool UpperCase { get; set; }
    public bool LowerCase { get; set; }

    private const string Alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    private const string SpecialChars = "~`!@#$%^&*()-_+";

    public PasswordGenerator() { }

    public string Generate(int Length)
    {
        if (!Special && !Numbers && !UpperCase && !LowerCase)
            throw new ArgumentException("At least one option must be set.");

        StringBuilder sBuilder = new StringBuilder(Length);

        for (int i = 0; i < Length; i++)
            sBuilder.Append(RandomChoice());

        return sBuilder.ToString();
    }

    private string RandomChoice()
    {
        string response = "";

        switch (RND.Next(0, 4))
        {
            case 0:
                if (!UpperCase)
```

```

        return "";
        response = Alphabet[RND.Next(Alphabet.Length)].ToString();
        break;
    case 1:
        if (!LowerCase)
            return "";
        response = Alphabet[RND.Next(Alphabet.Length)].ToString().ToLower();
        break;
    case 2:
        if (!Numbers)
            return "";
        response = RND.Next(0, 9).ToString();
        break;
    case 3:
        if (!Special)
            return "";
        response = SpecialChars[RND.Next(SpecialChars.Length)].ToString();
        break;
    default: break;
}

return response;
}
}

```

Secure Random Class

Spoiler ([Click to View](#))

Code class SecureRandom : RandomNumberGenerator

```

{
    private RandomNumberGenerator RNG = new RNGCryptoServiceProvider();
    private byte[] data = new byte[sizeof(uint)];

    public SecureRandom() { }

    public int Next()
    {
        RNG.GetBytes(data);
        return BitConverter.ToInt32(data, 0) & (int.MaxValue - 1);
    }

    public int Next(int maxValue)
    {
        return Next(0, maxValue);
    }

    public int Next(int minValue, int maxValue)
    {
        if (minValue > maxValue)
            throw new ArgumentOutOfRangeException();

        return (int)((minValue + ((double)maxValue - minValue) * NextDouble()));
    }
}

```

```

public double NextDouble()
{
    RNG.GetBytes(data);
    var randUint = BitConverter.ToUInt32(data, 0);
    return randUint / (uint.MaxValue + 1.0);
}

public double NextDouble(double min, double max)
{
    if (min > max)
        throw new ArgumentOutOfRangeException();

    return NextDouble() * (max - min) + min;
}

public override void GetBytes(byte[] data)
{
    RNG.GetBytes(data);
}

public override void GetNonZeroBytes(byte[] data)
{
    RNG.GetNonZeroBytes(data);
}

```

Usage

Spoiler ([Click to View](#))

```

Code PasswordGenerator pGenerator = new PasswordGenerator() { UpperCase = true, LowerCase
= true, Special = True };

```

```

string password = pGenerator.Generate(12) //12 being the length

```

[Project Source Download](#)

Changed (12/29/2016) Changed While Loop to For Loop.

Used StringBuilder.

Removed Math.Floor.

Removed issue with byte buffer being created every time.

Updated download link to new project source.

Chickendipper - December 20, 2016, 9:10 am

Your code is very nice, however I do have one gripe with it.

You use a while loop to select characters randomly based off of your RNG. This is a lot more hassle and uses more resources, and is a messier solution.

Instead, what you could do is:

```

Code StringBuilder charPoolBuilder = new StringBuilder();

```

```
if (UpperCase) charPoolBuilder.Append("ABCDEFGHIJKLMNOPQRSTUVWXYZ");
// ... etc
StringBuilder myPassword = new StringBuilder();
for (int i = 0; i < passwordLength; i++)
myPassword.Append(charPoolBuilder[RND.Next(charPoolBuilder.Length)]);
return myPassword.ToString();
```

This way, instead of doing twice the selection & having a messy case selection, you have a "one size fits all" solution. Now, if you wanted to add European characters (eg éíúóá), you could do so by doing

Code if (European) charPoolBuilder.Append("éíúóá"); rather than having an extra case statement. Otherwise, your code is great and a great learning point to others about CSPRNGs.

Zilog80 - December 20, 2016, 12:06 pm

Pull Request for PasswordGenerator class

PHP Code
<?php
while (password.Length < 12)

Should be:

PHP Code
<?php
while (password.Length < Length)

If you change this you can do this:

PHP Code
<?php
string password = pGenerator.Generate(12) //12 being the length
string password2 = pGenerator.Generate(15) //15 being the length
string password3 = pGenerator.Generate(25) //25 being the length

Mr Whiskey - December 20, 2016, 12:11 pm

I'm confused, you have a box to choose the length but it's hard-coded to 12?

Kaspah - December 20, 2016, 12:12 pm

I mean, nice contribution but what's the point when I can use a web-based generator?

Chickendipper - December 20, 2016, 12:21 pm

(Dec 20, 2016 01:12 PM)Kaspah Wrote: [\[CITE\]](#) I mean, nice contribution but what's the point when I can use a web-based generator?

Are you retarded? He's making a contribution so that others can learn from it.

Mr Whiskey - December 20, 2016, 12:25 pm

(Dec 20, 2016 01:12 PM)Kaspah Wrote: [\[CITE\]](#) I mean, nice contribution but what's the point when I can use a web-based generator?

I think he gave a pretty good example in the OP. "Feel free to use this for account creators"

Malum™ - December 21, 2016, 1:48 am

(Dec 20, 2016 01:06 PM)Zilog80 Wrote: [\[CITE\]](#) Pull Request for PasswordGenerator class

PHP Code
<?php
while (password.Length < 12)

Should be:

PHP Code
<?php
while (password.Length < Length)

If you change this you can do this:

PHP Code
<?php
string password = pGenerator.Generate(12) //12 being the length
string password2 = pGenerator.Generate(15) //15 being the length
string password3 = pGenerator.Generate(25) //25 being the length

Oh shit! My bad. I forgot to remove that. I was testing and forgot to remove that part. I'll fix it now.

passwords - December 22, 2016, 11:40 am

Very nice, this "Friend" of yours must have been something really special.
Anyways, Good contribution to the community!

Malum™ - December 23, 2016, 3:31 am

(Dec 22, 2016 12:40 PM)DOT1 Wrote: [\[CITE\]](#) Very nice, this "Friend" of yours must have been something really special.

Anyways, Good contribution to the community!

Why the hostility? It was someone on Skype wanting to learn about ListView/ListBox and Random String generation. So I threw this project together and added a little more to it to post on here.

EDIT: God damn it. DOT1 was the friend I made it for. Lmfao. You dick. Haha.

BahNahNah - December 23, 2016, 6:02 am

Just some things to think about:

Empty constructors can be omitted.

StringBuilder is more efficient when... building a string. [\(Click to View\)](#)

Decimal/float/double to int cast will drop the decimals, so no need for the extra Math.Floor.

Creating a new byte buffer for every random number is quite costly. [\(Click to View\)](#)

Malum™ - December 23, 2016, 6:16 am

Empty constructors can be omitted. - This seems extraordinarily picky..more like one of your pet-peeves.

StringBuilder is more efficient when... building a string. [\(Click to View\)](#) - Yea, that was my bad. I was having issues (brain fart) when using StringBuilder and resorted to concatenations. Never bothered to revert back.

Decimal/float/double to int cast will drop the decimals, so no need for the extra Math.Floor. - Didn't know that. Thank you!

Creating a new byte buffer for every random number is quite costly. [\(Click to View\)](#) - I appreciate that information. I didn't take that into consideration.

I would love and greatly appreciate whatever you use to benchmark. It would come in handy for future projects.

sinfulshotx - January 14, 2017, 5:38 pm

I just made one of these myself using very similar methods, great work. All improvements are done with practice and time.