

Forum:[Hack Forums](#)

Thread: [\[C#\] Flags](#)

FireNice - June 15, 2016, 3:33 pm

Hello.

Many of you use malware that uses multiple options to beautify their malware.

Some of you save the data in the form of string, like this:

Code ANTISANDBOXIE|ON

ANTISANDBOXIE|OFF

There is a system called Flags which is simple and easy to implement, and provides an economic level memory.

A flag represents a bit in a sequence of bits (typically an integer).

A Int32 (or int) is formed of 4 bytes, or 32 bits, so we can make 32 flags.

A flag is a bit, and who say bit, say 1 or 0, true or false.

Only options requiring only a true or false to be activated can be used. Anti Sandoxie

Anti VirusTotal

Anti Firewall

Anti UAC

RC4 Encryption

XOR Encryption

DES Encryption

Base64 Encryption

That's the example that will be used.

Here, only 8 flags are used, one byte is sufficient to store them.

Creating a structure that represents all this:

Code [Flags]

```
public enum BuildFlags : byte {  
    AntiSandoxie    = 0x01, // 0000 0001 = 1  
    AntiVirusTotal  = 0x02, // 0000 0010 = 2  
    AntiFirewall    = 0x04, // 0000 0100 = 4  
    AntiUAC         = 0x08, // 0000 1000 = 8  
    RC4Encryption   = 0x10, // 0001 0000 = 16  
    XOREncryption   = 0x20, // 0010 0000 = 32  
    DESEncryption   = 0x40, // 0100 0000 = 64
```

```
Base64Encryption = 0x80 // 1000 0000 = 128
}
```

Comments shows as well how are made each flag.

Now, if an option is selected, you must force the bit (or flag) corresponding to 1.

The logical binary OR allows to do this because:

Code $x | 1 = 1$

If an option is not selected, the bit is forced to 0 with AND:

Code $x \& 0 = 0$

Creating a variable that will hold our flags:

Code BuildFlags oFlags = 0;

To enable XOR and ANTI UAC:

Code oFlags |= BuildFlags.XOREncryption;

oFlags |= BuildFlags.AntiUAC;

Now we stock it in our malware as you normally would.

To see if a flag is used or not, a simple condition:

```
Code if ((oFlags | BuildFlags.XOREncryption) == BuildFlags.XOREncryption) {
    Console.WriteLine("XOR!");
}
Etc...
```

You just put up a clever and economic system that uses the bit!

By reading this tutorial from beginning to end, you have become smarter than you were before. Continue like this and you'll accomplish great things.

SL x TnT - June 15, 2016, 4:57 pm

Rare to see a tutorial that explains things, but shouldn't the last snippet be this?

```
Code if ((oFlags & BuildFlags.XOREncryption) == BuildFlags.XOREncryption) {
    Console.WriteLine("XOR!");
}
```

With .NET, you can also use [this method](#).

Not part of this tutorial, but on the UI side you can add all the options in during runtime. Set the Tag property as the enum, have them all use the same check changed event, grab the enum from the Tag and set the flag from that.

Temploit - June 16, 2016, 3:19 am

(Jun 15, 2016 04:57 PM)SL x TnT Wrote: [CITE](#) Rare to see a tutorial that explains things, but shouldn't the last snippet be this?

```
Code if ((oFlags & BuildFlags.XOREncryption) == BuildFlags.XOREncryption) {
    Console.WriteLine("XOR!");
}
```

With .NET, you can also use [this method](#).

Not part of this tutorial, but on the UI side you can add all the options in during runtime. Set the Tag property as the enum, have them all use the same check changed event, grab the enum from the

Tag and set the flag from that.

You're right, it should be & instead of |. HasFlag does the exact same thing internally, but is only available from .net 4.0 onward (kinda late).

severos - June 16, 2016, 4:14 am

Now that's something nice, I already used it before but it's good to see someone posting a good tutorial about it, but I have a question that came up while reading, can I or multiple flags in a single state? likeCode oFlags |= BuildFlags.XOREncryption | BuildFlags.AntiUAC;

FireNice - June 16, 2016, 6:40 am

(Jun 15, 2016 04:57 PM)SL x TnT Wrote: [\[CITE\]](#) Rare to see a tutorial that explains things, but shouldn't the last snippet be this?

```
Code if ((oFlags & BuildFlags.XOREncryption) == BuildFlags.XOREncryption) {  
    Console.WriteLine("XOR!");  
}
```

With .NET, you can also use [this method](#).

Not part of this tutorial, but on the UI side you can add all the options in during runtime. Set the Tag property as the enum, have them all use the same check changed event, grab the enum from the Tag and set the flag from that.

It can be both, that's why I said "etc..." at the end.

SL x TnT - June 16, 2016, 1:27 pm

(Jun 16, 2016 06:40 AM)FireNice Wrote: [\[CITE\]](#) It can be both, that's why I said "etc..." at the end.

Did you test it? Earlier you said that when you OR something, you're adding. Unless you're starting with a single bit set of the one you're testing, it shouldn't work. With no bits set, it'll always be true. With 2, it'll always be false.

BahNahNah - June 16, 2016, 7:00 pm

What am i missing

Spoiler [\(Click to View\)](#)

SL x TnT - June 16, 2016, 8:14 pm

(Jun 16, 2016 07:00 PM)BahNahNah Wrote: [\[CITE\]](#) What am i missing

Spoiler [\(Click to View\)](#)

You need to set the values.

Spoiler [\(Click to View\)](#)

TERR - June 28, 2016, 10:08 am

This is basic bitwise arithmetic... If you don't know this by now you shouldn't be programming.

@BahNahNah You're creating a variable with the enum type, and setting it to the value of "One" which is in terms, 0000 0000

Then you're using the OR operation with the variable with two, which then becomes 0000 0001

Next you're using the OR operation with Three, 0000 0011

Your final result will be

0000 0011 << == 3.

You should be using the AND bitwise operation, as AND works like this:

0010 & 0001 = 0000

or

0010 & 0011 = 0010

Your flag's shouldn't coincide with each other, for example, having one flag as 1, and another as 2, and another as 3.

If three is enabled, but two and one are not, logically you will assume they are considering THREE takes the place of 2 and 1.... as flag & 1 will == 1 and flag & 10 == 10.

OP should've done a better job explaining why the flags are separate by a factor of 2^n

FireNice - June 28, 2016, 3:36 pm

(Jun 28, 2016 10:08 AM)TERR Wrote: [\[CITE\]](#) This is basic bitwise arithmetic... If you don't know this by now you shouldn't be programming.

True.

CoLDarkness - June 29, 2016, 12:38 am

The whole time you spent on this is nice and thread looks neatly formatted, as the other user said it's very basic.

ubbelol - June 29, 2016, 1:47 am

(Jun 28, 2016 03:36 PM)FireNice Wrote: [\[CITE\]](#) True.

You say this, but in your tutorial you use OR to check if a flag is set??