

Forum:[Hack Forums](#)

Thread: [Anyone recognize this obfuscation?](#)

sneezing panda - June 9, 2018, 10:15 pm

Anyone here recognize this type of obfuscation?

```
Code public class 11 extends 5
{
    public 11() {
        Label_0004: {
            break Label_0004;
            try {
                throw;
            Label_0009_Outer:
                while (true) {
                    null;
                    while (true) {
                        continue;
                    }
                    continue Label_0009_Outer;
                }
            }
            // iftrue(Label_0013:, 4 > 0)
            catch (Throwable t) {
                throw t;
            }
        }
        Label_0013: {
            final 11 11 = this;
        }
        final String 1i11I1I1I1I1I = "\u71d8\u71d3\u71f1\u71d3\u71f0\u71fc\u71fb";
        final String 1i11I1I1I1I1I1 =
"\u71bd\u71bd\u7192\u71bb\u7191\u7192\u718f\u71bb\u7186\u7191\u71bb\u718b\u71bb\u7186\u718d\u718d\u718a\u71bb\u7186\u71bb\u718a\u7189\u71ba\u7186\u7192\u7190\u71bb\u718e\u71be\u7189\u718b\u718f\u7190\u7192\u718f\u7192";
        " ".length();
        final String 1i11I1I1I1I1I1 = 1i11I1I1I1I1I1((CharSequence)1i11I1I1I1I1I,
(CharSequence)1i11I1I1I1I1I1);
        final String[] array = null;
        " ".length();
        new 5(1i11I1I1I1I1I1, array);
        this = null;
    }
}
```

GenericSkid - June 10, 2018, 9:42 am

Need more for confirmation, but maybe Stringer.

Edit: See post below for analysis

sneezing panda - June 10, 2018, 10:58 am

I'm mainly interested in the parts that look like this.

```
Code final String 1i11I1I1I1I11 = 1i11I1I1I1I11((CharSequence)1i11I1I1I1I1I,  
(CharSequence)1i11I1I1I1I11);
```

There are parts like that all over the source code. I've seen a reference to a "FumeObfuscator" but I see nothing online about it..

GoodVibes - June 10, 2018, 1:58 pm

(Jun 10, 2018 10:58 AM)sneezing panda Wrote: [\[CITE\]](#) I'm mainly interested in the parts that look like this.

```
Code final String 1i11I1I1I1I11 = 1i11I1I1I1I11((CharSequence)1i11I1I1I1I1I,  
(CharSequence)1i11I1I1I1I11);
```

There are parts like that all over the source code. I've seen a reference to a "FumeObfuscator" but I see nothing online about it..

I've seen private obfuscator do this, but this is a simple method I'm sure any of the better obfuscators employ by now. Can you say what software you're looking at?

edit: was talking about the jump parts, that's just the string obfuscating part

sneezing panda - June 10, 2018, 5:18 pm

(Jun 10, 2018 01:58 PM)GoodVibes Wrote: [\[CITE\]](#) I've seen private obfuscator do this, but this is a simple method I'm sure any of the better obfuscators employ by now. Can you say what software you're looking at?

edit: was talking about the jump parts, that's just the string obfuscating part

Some minecraft client. The file I'm looking at is "LiquidBounce1.12.2.jar" I just wanna see how they do stuff in it.

GoodVibes - June 10, 2018, 6:06 pm

(Jun 10, 2018 05:18 PM)sneezing panda Wrote: [\[CITE\]](#) Some minecraft client. The file I'm looking at is "LiquidBounce1.12.2.jar" I just wanna see how they do stuff in it.

What you looking to do bro, I (and genericskid, he's prolly better connected than I am if you looking for exploits it has) can probably help you out with it

GenericSkid - June 10, 2018, 6:29 pm

(Jun 10, 2018 05:18 PM)sneezing panda Wrote: [\[CITE\]](#) Some minecraft client. The file I'm looking at is "LiquidBounce1.12.2.jar" I just wanna see how they do stuff in it.

Link for jar: <https://my.mixtape.moe/epjvbu.jar>

The javap output of any file in the jar shows this:

```
Code public class byCCBlueX.AA extends byCCBlueX.wF
```

```
SourceFile: "FumeObfuscator"
```

So GoodVibes is right on the part of it being a private obfuscator.

Lets take a look at another sample:

PHP Code

```
<?php
```

```
// CFR decompilation
```

```
public pE() {
```

```
    if (7 > 0) {
```

```

        pE pE2;
        pE2.İ = LogFactory.getLog(pE2.getClass());
        pE2.İ- = new ConcurrentHashMap<a, d>(1024);
        pE2.İµ = new pL();
        pE2.Đ€ = new qc();
        pE2.Đ^ = false;
        return;
    }
    Object v0 = null;
    do {
        // Infinite loop
    } while (true);
}

```

PHP Code

```

<?php
// Procyon Decompilation
public pE() {
    Label_0004: {
        break Label_0004;
    }
    try {
        throw;
        while (true) {
            while (true) {
                continue;
                null;
                continue;
            }
            continue;
        }
    }
    // iftrue(Label_0014:, 7 > 0)
    catch (Throwable t) {
        throw t;
    }
}
Label_0014: {
    super();
}
this.\u00cf = LogFactory.getLog((Class)this.getClass());
this.\u00cf\u00ad = new ConcurrentHashMap<a, d>(1024);
this.\u00cfµ = new pL();
this.\u00d0\u20ac = new qc();
this.\u00d0\u02c6 = false;
}

```

This actually is pretty weak obfuscation if you break it down. Lets take a look at the bytecode of the method above.

PHP Code

```

<?php
0: goto      4
3: athrow
4: bipush     7

```

```

6: ifgt      14
9: aconst_null
10: goto     10
13: athrow
14: aload_0
15: invokespecial #69 // Method byCCBlueX/pz."<init>":()V
18: aload_0
19: aload_0
20: invokevirtual #75 // Method java/lang/Object.getClass:()Ljava/lang/Class;
23: invokestatic
#81 // Method org/apache/commons/logging/LogFactory.getLog:(Ljava/lang/Class;)Lorg/apache/commons/logging/Log;
26: putfield    #83 // Field Ĩ:Lorg/apache/commons/logging/Log;
29: aload_0
30: new         #85 // class java/util/concurrent/ConcurrentHashMap
33: dup
34: sipush     1024
37: invokespecial #88 // Method java/util/concurrent/ConcurrentHashMap."<init>":(I)V
40: putfield    #90 // Field Ĩ:Ljava/util/Map;
43: aload_0
44: new         #92 // class byCCBlueX/pL
47: dup
48: invokespecial #93 // Method byCCBlueX/pL."<init>":()V
51: putfield    #95 // Field Ĩµ:LbyCCBlueX/pL;
54: aload_0
55: new         #97 // class byCCBlueX/qc
58: dup
59: invokespecial #98 // Method byCCBlueX/qc."<init>":()V
62: putfield    #100 // Field Ð€:LbyCCBlueX/pW;
65: aload_0
66: iconst_0
67: putfield    #102 // Field Ð^:Z
70: return

```

Looks kinda complicated, but I'm going to format this a bit and add some more notes. It should make a lot more sense once this is done. The only reason why decompilers trip up on this code is the first part, range 0-14, so lets look at that first.

PHP Code

```

<?php
0: goto      4 // Instantly jump to <@4>
3: athrow           // UNUSED
4: bipush      7 // Push "7" to the stack
6: ifgt      14 // Jump to <@14> if the top-stack value (7) > 0, which it is
9: aconst_null     // UNUSED
10: goto      10 // UNUSED
13: athrow           // UNUSED
14: aload_0         // The method actually starts here

```

As you can see in the comments, this just jumps from 0->4 then 4->14. So the method really begins at offset 14. The decompiler shows this code because it does not check the bytecode for redundancy like this. This is a technique called [opaque predicates](#). The rest of the code is stupid simple.

```

PHP Code
<?php
// everything before is opaque predicates
// AKA junk that gets skipped.
14: aload_0
15: invokespecial #69          // Method byCCBlueX/pz."<init>":()V
   // this.super();
18: aload_0
19:  aload_0
20:  invokevirtual #75          // Method java/lang/Object.getClass:()Ljava/lang/Class;
23: invokestatic
#81          // Method org/apache/commons/logging/LogFactory.getLog:(Ljava/lang/Class;)Lorg/a
pache/commons/logging/Log;
26: putfield    #83          // Field Ĩ:Lorg/apache/commons/logging/Log;
   // this.Ĩ = LogFactory.getLog(this.getClass())
29: aload_0
30: new         #85          // class java/util/concurrent/ConcurrentHashMap
33: dup
34: sipush     1024
37: invokespecial #88          // Method java/util/concurrent/ConcurrentHashMap."<init>":(I)V
40: putfield    #90          // Field Ĩ:Ljava/util/Map;
   // this.Ĩ = new ConcurrentHashMap(1024);
43: aload_0
44: new         #92          // class byCCBlueX/pL
47: dup
48: invokespecial #93          // Method byCCBlueX/pL."<init>":()V
51: putfield    #95          // Field Ĩµ:LbyCCBlueX/pL;
   // this.Ĩµ = new pL();
54: aload_0
55: new         #97          // class byCCBlueX/qc
58: dup
59: invokespecial #98          // Method byCCBlueX/qc."<init>":()V
62: putfield    #100         // Field Đ€:LbyCCBlueX/pW;
   // this.Đ€ = new qc();
65: aload_0
66: iconst_0          // 0 in the jvm = false
67: putfield    #102         // Field Đ^:Z
   // this.Đ^ = false;
70: return

```

This is your standard constructor where fields are instantiated. The indented sections are used to visualize the stack size. I've sectioned the code into where line breaks in the source code would appear.

For recommendations on handling this, I would personally write a small [ASM](#) program to snip off the bullshit at the beginning of methods (It seems to be applied to all of them, and the junk only shows up at the beginning, which is hilariously weak and easy to remove via automation) and then run the cleaned output through proguard for simpler (Non-unicode) names.

If you wanted to remap some cleartext names, I would suggest looking at the mixins package. It contains proxy/wrapper classes for core Minecraft classes. The annotations and mixin framework

will redirect calls to these cleartext names to the obfuscated methods in the classes protected by the obfuscation. If you know what the mixins are doing, you can get a sense of what the obfuscated code calls they make in the methods are doing.

Also I don't see any string obfuscation in here. Most of the shit is just junk code and basic class and member renaming.

sneezing panda - June 10, 2018, 9:24 pm

Well that's pretty damn neat.. Unfortunately I'm an idiot when it comes to coding.. I can read a lot of it pretty well, but I can't make it do shit.. Thanks for the fantastic info though.

GenericSkid - June 10, 2018, 9:36 pm

(Jun 10, 2018 09:24 PM)sneezing panda Wrote: [\[CITE\]](#) Well that's pretty damn neat.. Unfortunately I'm an idiot when it comes to coding.. I can read a lot of it pretty well, but I can't make it do shit.. Thanks for the fantastic info though.

Check:

<https://github.com/java-deobfuscator/deobfuscator>

<https://github.com/java-deobfuscator/deobfuscator-gui>

The code removal may be supported by [DeadCodeRemover](#).

sneezing panda - June 10, 2018, 10:18 pm

Yea, I followed through with what deobfuscator-gui said last night and I get something like this. I just clicked on a random .class file btw.

Code package byCCBlueX;

```
import byCCBlueX.FumeObfuscator;
import byCCBlueX.FumeObfuscator1043;
import byCCBlueX.FumeObfuscator1046;
import byCCBlueX.FumeObfuscator1047;
import byCCBlueX.FumeObfuscator1048;
import byCCBlueX.FumeObfuscator71;
import byCCBlueX.FumeObfuscator95;
import byCCBlueX.FumeObfuscator973;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Collection;
import java.util.LinkedHashSet;
import java.util.Set;

public class FumeObfuscator975
implements FumeObfuscator1048 {
    private /* synthetic */ ClassLoader \u00d0\u0192;
    private final /* synthetic */ Set<FumeObfuscator1043> \u00d0\u02c6;

    /*
     * Enabled aggressive block sorting
     * Enabled unnecessary exception pruning
     * Enabled aggressive exception aggregation
     */
    public FumeObfuscator975() {
        if (3 > 0) {
```

```

        FumeObfuscator975 fumeObfuscator975;
        fumeObfuscator975.\u00d0\u02c6 = new LinkedHashSet(4);
        fumeObfuscator975.\u00d0\u0192 = FumeObfuscator95.\u00cf();
        return;
    }
    Object v0 = null;
    do {
        // Infinite loop
    } while (true);
}

/*
 * Enabled aggressive block sorting
 * Enabled unnecessary exception pruning
 * Enabled aggressive exception aggregation
 */
public FumeObfuscator975(ClassLoader classLoader) {
    if (9 > 0) {
        void 1i1i1i1i1i1i1i1i;
        FumeObfuscator975 fumeObfuscator975;
        fumeObfuscator975.\u00d0\u02c6 = new LinkedHashSet(4);
        fumeObfuscator975.\u00d0\u0192 = 1i1i1i1i1i1i1i1i;
        return;
    }
    Object v0 = null;
    do {
        // Infinite loop
    } while (true);
}
}

```

Also get this weird stuff. Quite a few .class files are like that. Name wise anyway.

GenericSkid - June 10, 2018, 10:31 pm

(Jun 10, 2018 10:18 PM)sneezing panda Wrote: [\[CITE\]](#) Yea, I followed through with what deobfuscator-gui said last night and I get something like this. I just clicked on a random .class file btw.;

Also get this weird stuff. Quite a few .class files are like that. Name wise anyway.
 Hmm... Can you post the modified jar? Also maybe try another tool like:

<https://github.com/Col-E/Recaf>

Edit: Dead code remover from deobfuscator does not work...

sneezing panda - June 10, 2018, 11:31 pm

(Jun 10, 2018 10:31 PM)GenericSkid Wrote: [\[CITE\]](#) Hmm... Can you post the modified jar? Also maybe try another tool like:

<https://github.com/Col-E/Recaf>

Edit: Dead code remover from deobfuscator does not work...

Yea, here's the file I ran with jar deobfuscator.

<https://mega.nz/#!8MsmAJLL!J3m7rJ06-IK0D8dTgHnxxowtfqjvxJUWy2UQmIX24DY>

And Recaf fails to load the modified file..

GenericSkid - June 11, 2018, 12:11 am

(Jun 10, 2018 11:31 PM)sneezing panda Wrote: [\[CITE\]](#) Yea, here's the file I ran with jar deobfuscator. <https://mega.nz/#!8MsmAJLL!J3m7rJ06-IK0D8dTgHnxxowtfqjvxJUWy2UQmIX24DY>

And Recaf fails to load the modified file..

Make an issue for recaf, the dude seems to respond to bug reports stupidly fast. Looks like its a name parsing issue. Probably cuz the unicode names. Also, that build looks old.

sneezing panda - June 11, 2018, 12:22 am

(Jun 11, 2018 12:11 AM)GenericSkid Wrote: [\[CITE\]](#) Make an issue for recaf, the dude seems to respond to bug reports stupidly fast. Looks like its a name parsing issue. Probably cuz the unicode names. Also, that build looks old.

Possibly, and I was on 1.0, looks like it's now 1.2.0.. In any case, it still happens so I'll make a report. :)

Just a quick update, he updated it for that bug report. Seems like an issue both with a blank name .class file and crazy long file names.

<https://github.com/Col-E/Recaf/releases/tag/1.2.1>

EDIT: And it NPE's on startup, but I made another bug report lol.

GenericSkid - June 12, 2018, 12:42 am

(Jun 11, 2018 12:22 AM)sneezing panda Wrote: [\[CITE\]](#) Possibly, and I was on 1.0, looks like it's now 1.2.0.. In any case, it still happens so I'll make a report. :)

Just a quick update, he updated it for that bug report. Seems like an issue both with a blank name .class file and crazy long file names.

<https://github.com/Col-E/Recaf/releases/tag/1.2.1>

EDIT: And it NPE's on startup, but I made another bug report lol.

Looks like that's already resolved. Anyways he's right, looks like the auto-deobfuscator is fucking things up with both naming and other attributes. At this point you may as well open an issue requesting he do the whole thing for you ?