

SP2023 Week 14 • 2023-04-27

Java Reverse Engineering

Hassam and Suchit



Announcements

- eCTF Third Place!
- Michael Goetzmann!
 - Sunday at 1:30



sigpwny{cookie_jar}



How does Java work?

- .jar files
 - This is a java "executable"
 - Just a zip file but with the .jar file extension
 - Contains compiled Java code (.class) and metadata (META-INF)
- .class files
 - Represent a compiled .java file
 - Contains JVM bytecode
- Metadata
 - In file META-INF/MANIFEST.MF
 - Tells the JVM which file to start in, and some other data



Aside: The Java Virtual Machine

- javac (The Java Compiler), converts your Java code into a .class file
 - .class files are run by the JVM, processed one instruction at a time
- The JVM is a **stack-based** virtual machine
 - All instructions take arguments using push and pop instructions onto a global stack
- This is a lot easier to compile than modern computers, but it is also a lot easier to reverse, since there is less complex obfuscation you can do



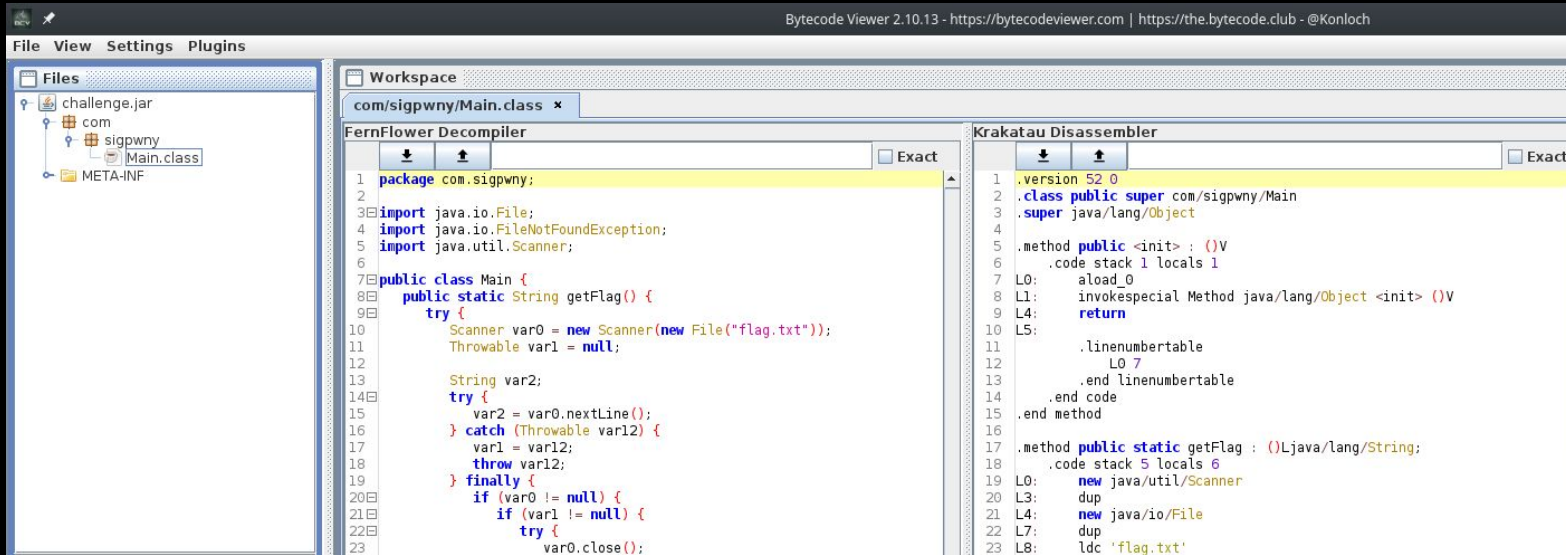
Good news about Java

- Very easy to decompile accurately
- You can keep:
 - Class names
 - Function names
 - Class variable names
 - Line numbers!!!
 - (Used for printing exception stack traces, but makes decompilation even easier)
- You might lose:
 - Local variable names
 - Comments



How to decompile

- Use Bytecode-Viewer
 - <https://github.com/Konloch/bytecode-viewer>
 - Procyon, FernFlower decompilers included



The screenshot displays the Bytecode Viewer application interface. The title bar reads "Bytecode Viewer 2.10.13 - https://bytecodeviewer.com | https://the.bytecode.club - @Konloch". The interface is divided into several panes:

- Files:** Shows a file tree with "challenge.jar", "com", "sigpwny", "Main.class", and "META-INF".
- Workspace:** Shows the loaded class "com/sigpwny/Main.class".
- FernFlower Decompiler:** Displays the decompiled Java source code for "Main.class". The code includes package declarations, imports for File, FileNotFoundException, Scanner, and String, and a public class Main with a static method getFlag() that uses Scanner to read a file.
- Krakatau Disassembler:** Displays the corresponding bytecode instructions for the decompiled code, including version information, class and super class declarations, and method bodies for <init> and getFlag.



Java Obfuscation

- Obfuscators can
 - Change control flow
 - Rename classes/variables/functions
 - Remove + add functions
 - Encrypt strings
 - Make it impossible to use a decompiler
- Popular obfuscators:
 - ZKM, Allatori, Proguard



Java Deobfuscation

- Non-manual strategy:
 - Run the obfuscated .jar through a deobfuscator
 - Run the deobfuscated jar through your decompiler
 - Typically only works if obfuscation is not custom
 - Still lose class/variable/function names
- Use <https://github.com/java-deobfuscator/deobfuscator>
- See <https://youtu.be/oal1F7D2Z2A?t=428> for a tutorial



Java Deobfuscation

Original

Obfuscated

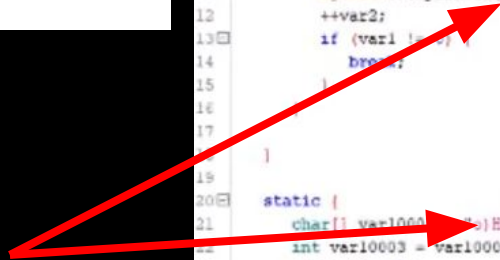
Deobfuscated

```
Fernflower Decompiler - Editable: false
1 package com.sigpwny;
2
3 public class Main {
4     public static void main(String[] args) {
5         for(int i = 0; i < 5; ++i) {
6             System.out.println("Hello world: " + i);
7         }
8     }
9 }
10
11
```

```
Fernflower Decompiler - Editable: false
1 public class a {
2     public static int a;
3     public static int b;
4     private static final String c;
5
6     public static void main(String[] var0) {
7         int var2 = 0;
8         int var1 = b;
9
10        while(var2 < 5) {
11            System.out.println(c + var2);
12            ++var2;
13            if (var1 != 0) {
14                break;
15            }
16        }
17
18        static {
19            char[] var10001 = "H2R\nEH>B;\u0007\n".toCharArray();
20            int var10003 = var10001.length;
21            int var0 = 0;
22            char[] var10002 = var10001;
23            int var2 = var10003;
24            String var1;
25            char[] var4;
26            int var10004;
27            boolean var5;
28        }
29    }
30}
```

```
Fernflower Decompiler - Editable: false
1 public class a {
2     public static int a;
3     public static int b;
4     private static final String c = "Hello world: ";
5
6     public static void main(String[] var0) {
7         int var2 = 0;
8         int var1 = b;
9
10        while(var2 < 5) {
11            System.out.println("Hello world: " + var2);
12            ++var2;
13            if (var1 != 0) {
14                break;
15            }
16        }
17
18    }
19}
```

Encrypted
"Hello world"



Tackling Custom Obfuscation

- See how far regular deobfuscators get you
- Use ObjectWeb ASM library to parse + manipulate class bytecode yourself
 - Prepare to spend a lot of time on this
- Newer constructs in later JVM standards (invokedynamic) have created opportunities for new crazy obfuscators
 - See <https://itzsomebody.xyz/2021/07/11/javaisez3-writeup.html>
- If all else fails: Take a peek at the bytecode and treat it like normal rev!



Go try for yourself!

<https://ctf.sigpwny.com>

- Start with Java Reversing 1
- Bytecode-Viewer can solve Java Reversing 1, 1.5, and 2
- Practice practice practice! Ask for help!

