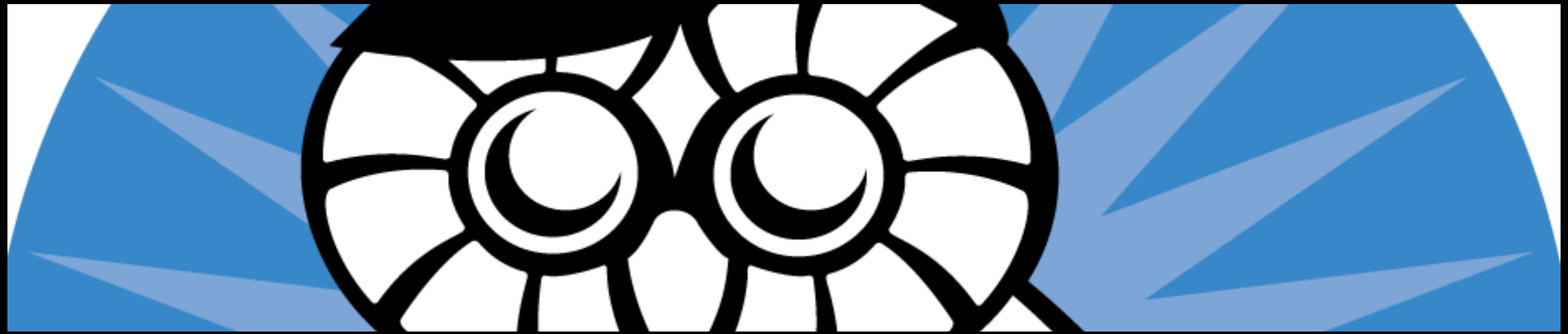
Reversing Java (Malware) with Radare

Adam Pridgen
April 2014

About me

- Rice SecLab, a PhD Student
- Independent InfoSec Consultant/Contractor



Overview

- Typical Java Reversing Talk
  - Decompile Code
  - Make Changes
  - Recompile and Win?
- Java Malware: Fail!

Overview

Has this happened to you?

```
do
{
  char[]                                  tmp357_356
  switch
  {
  case 0:
    tmpTernaryOp = 57; break;
  case 1:
    tmpTernaryOp = 83; break;
  case 2:
    tmpTernaryOp = 44; break;
  case 3:
    tmpTernaryOp = 38                     ak;
  }
  tmp357_355[tmp357_356] = ((char)(tmp357_355[tmp357_356]
} while (// INTERNAL ERROR //
```

What?!?

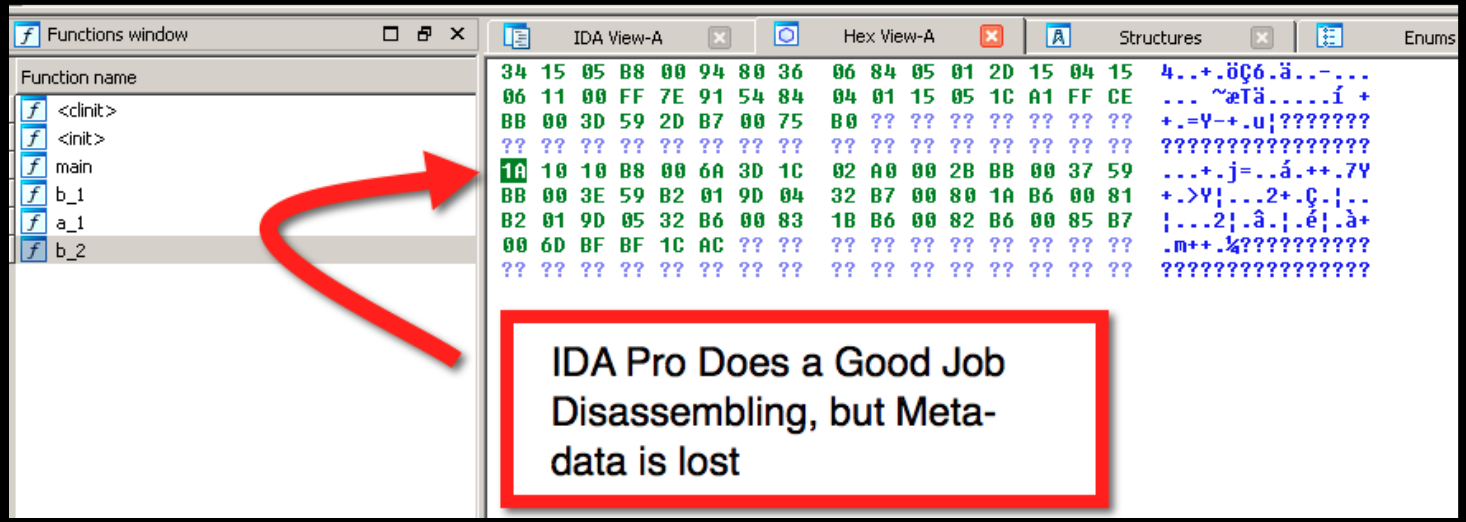
b.class | c.class

```
import sun.misc.BASE64Encoder;

public class c
{
  public static boolean a;
  private static final String z = -1;

  public static String a(String paramString)
  {
    return new BASE64Encoder().encode(paramString.getBytes(z));
  }

  /* Error */
  public static Str                       ring paramString)
  {
    // Byte code:
    //   0: getstatic 43    c:a Z
    //   3: istore_1
    //   4: new 3    java/lang/String
    //   7: dup
    //   8: new 5    sun/misc/BASE64Decoder
    //   11: dup
    //   12: invokespecial 10  sun/misc/BASE64Decoder:<init>  ()V
    //   15: aload_0
```

OOPs?

Overview

- IDA Pro 6.4 does not include meta-data



Overview

- Malicious code analysis is hard
- Relevant information is key
- Tools assume code is complete or correct

Overview

- Reversing JVM Bytecode viewed as a “simple” problem
  - Until you need to actually do it
  - Or you need to extract some type of information

- Too Long Didn’t Listen (tldl;)
  - Radare now supports basic class file manipulations
  - Hooking by rewriting class and method names
  - Manipulation of Access Flags
  - Inserting values in constant pool
  - More detailed inspection of files

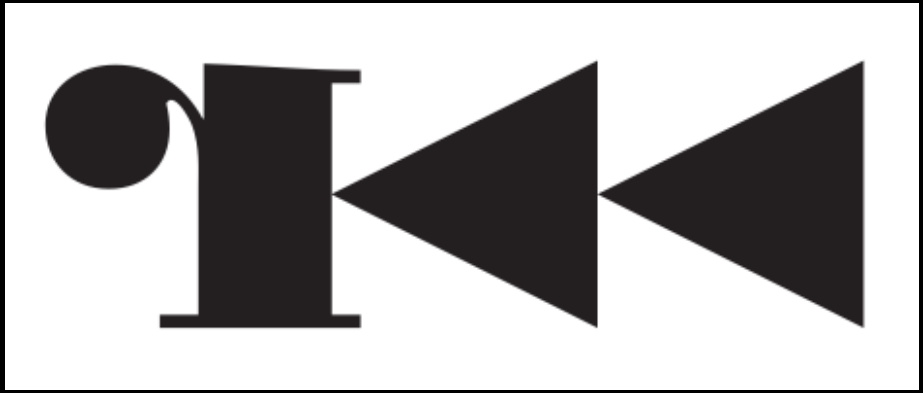
Multiple Architectures

Command Based

Extendible Components

Multi-Language w/ Ctypes

Open Source



IL in progress

2048

Cross Platform

GDB Interface

Hex Editor

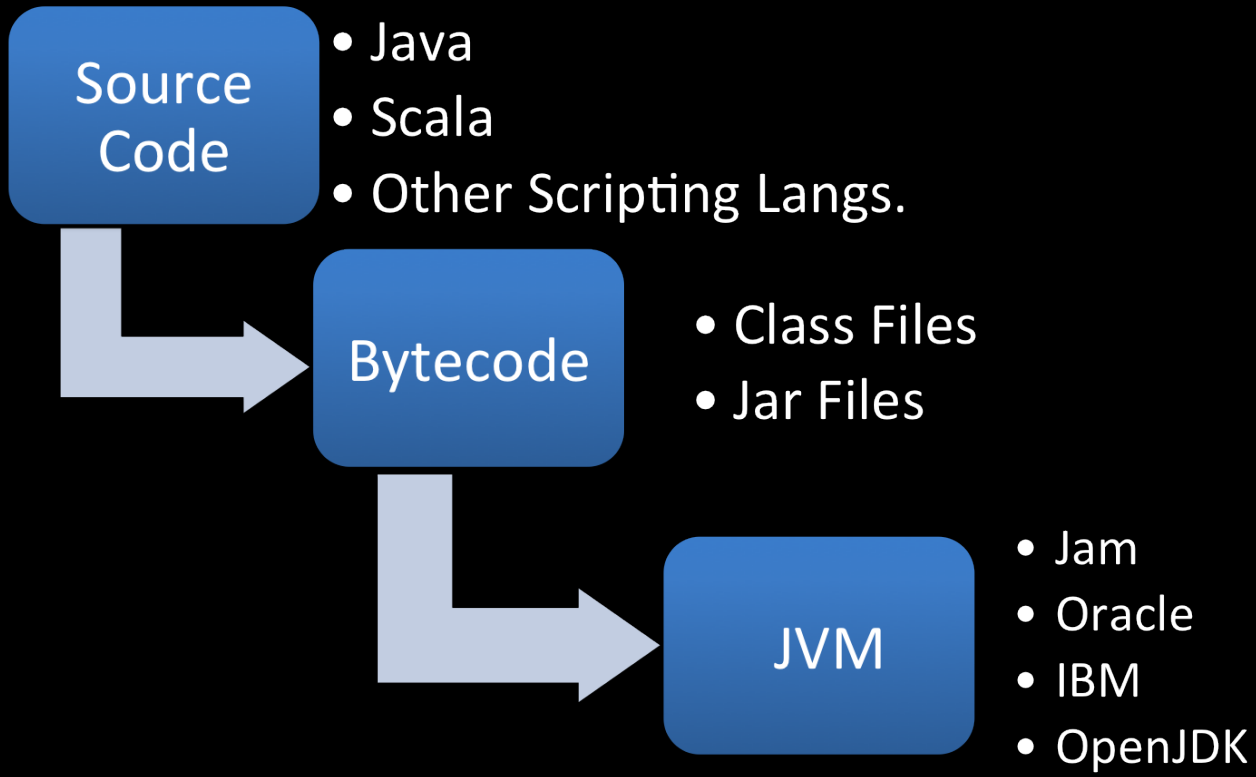
Supports IO Layers

Web UI

Agenda

- Discuss Java Class File and Format
- Discuss Java Malware and Obfuscation
- Introduce Java Reversing with Radare
- Discuss Some Techniques
- Conclude with Future Work

Java Overview



JVM Bytecode

- ~203 Operations
- Fairly easy to disassemble
  - Except for the built in “switch-tables”
- JVM is Stack Based
- Local Variables are stored in a local variable position

JVM Bytecode

- Caller copy the entire thread stack to caller
- JVM resolves Class Name, Method Name, and argument types
- Types are not important until they are important

Java Malware Obfuscation

- Static Obfuscation Techniques
- Dynamic Techniques

Java Malware via Static Obfuscation

- Flatten Classes and Package Hierarchy
- Homogenous type signatures
- Make class names uninterpretable
- Exploit compiler features
- Dead code
- Local variable Type overloading
- Hiding strings or files in strange places

Java Malware via Dynamic Obfuscation

- Reflection or Custom Class loaders
- Starting a new process
- Scripting Engine
- String Manipulation
- Encryptions

Java Malware Reversing

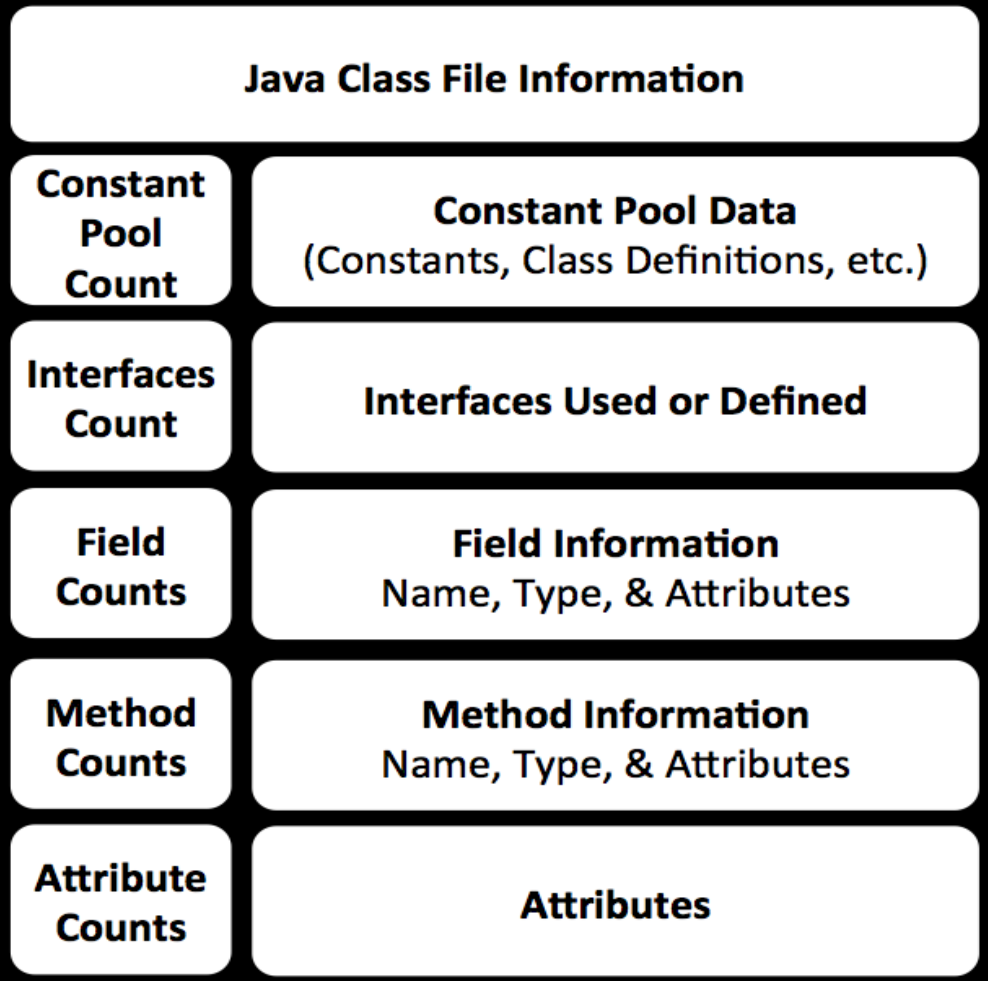
- Not easily decompilable (if at all)
- No standard tools for inspections
- Modification is tedious to do by hand

What Radare can do with Java?

- Basic hooking of class methods
- Change constant pool Values
- Modify method and field access flags
- Disassemble code
- Load classes from strings
- Open the JAR and view all the files
- Yank a file to disk or insert it in the JAR

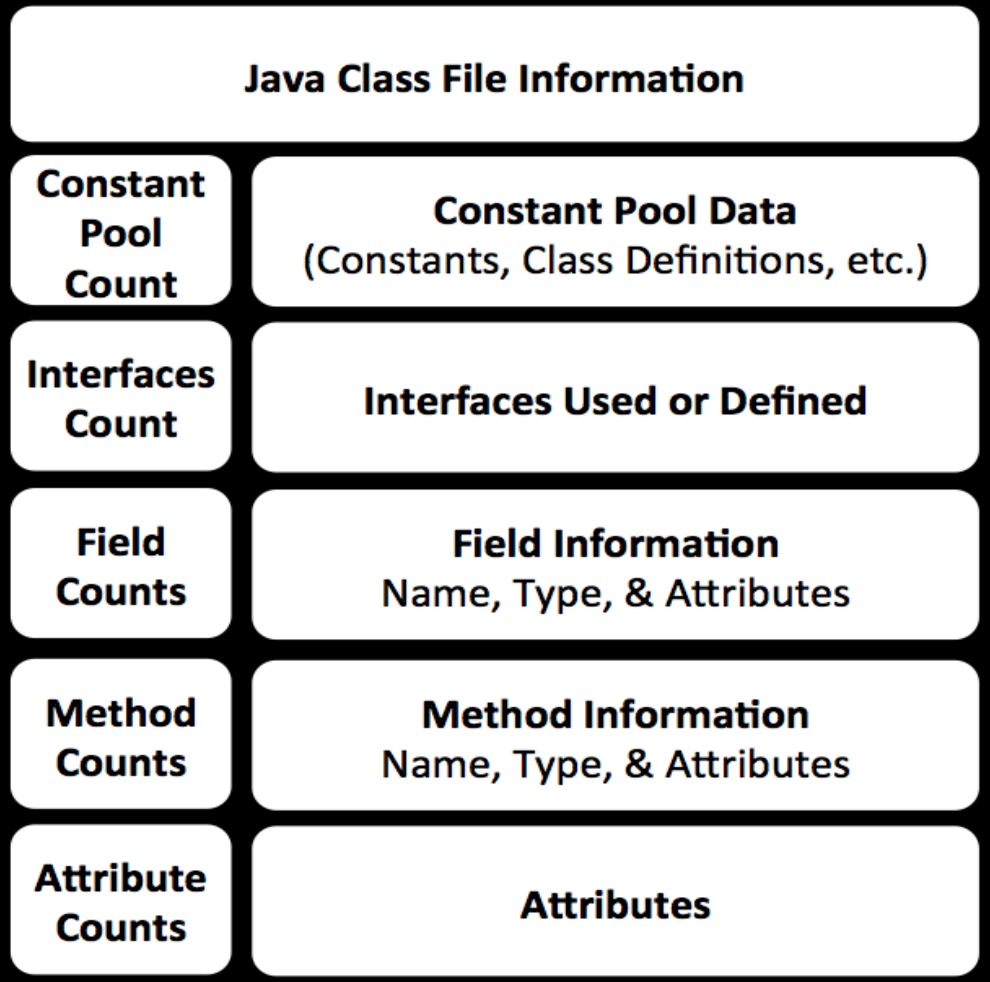
Class File Organization

Class File Organization



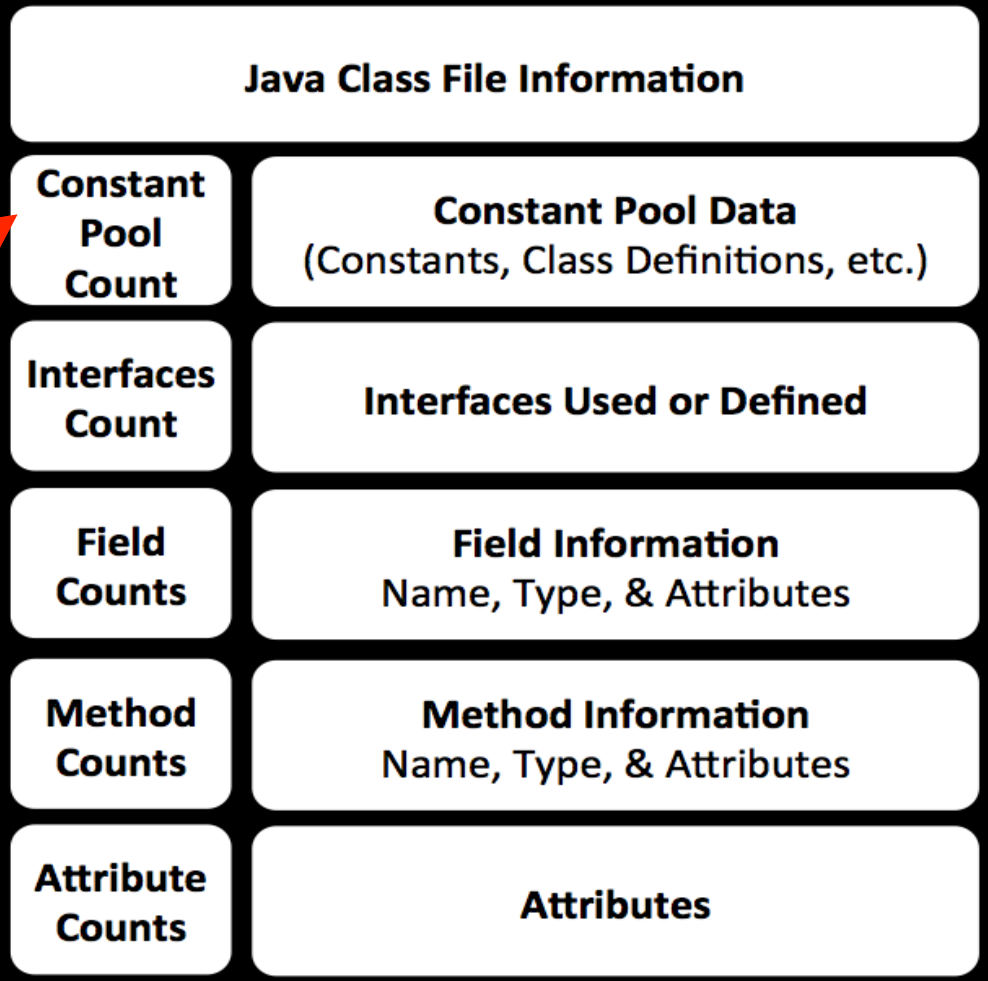
Class File Organization

- Magic Bytes
- Version Information



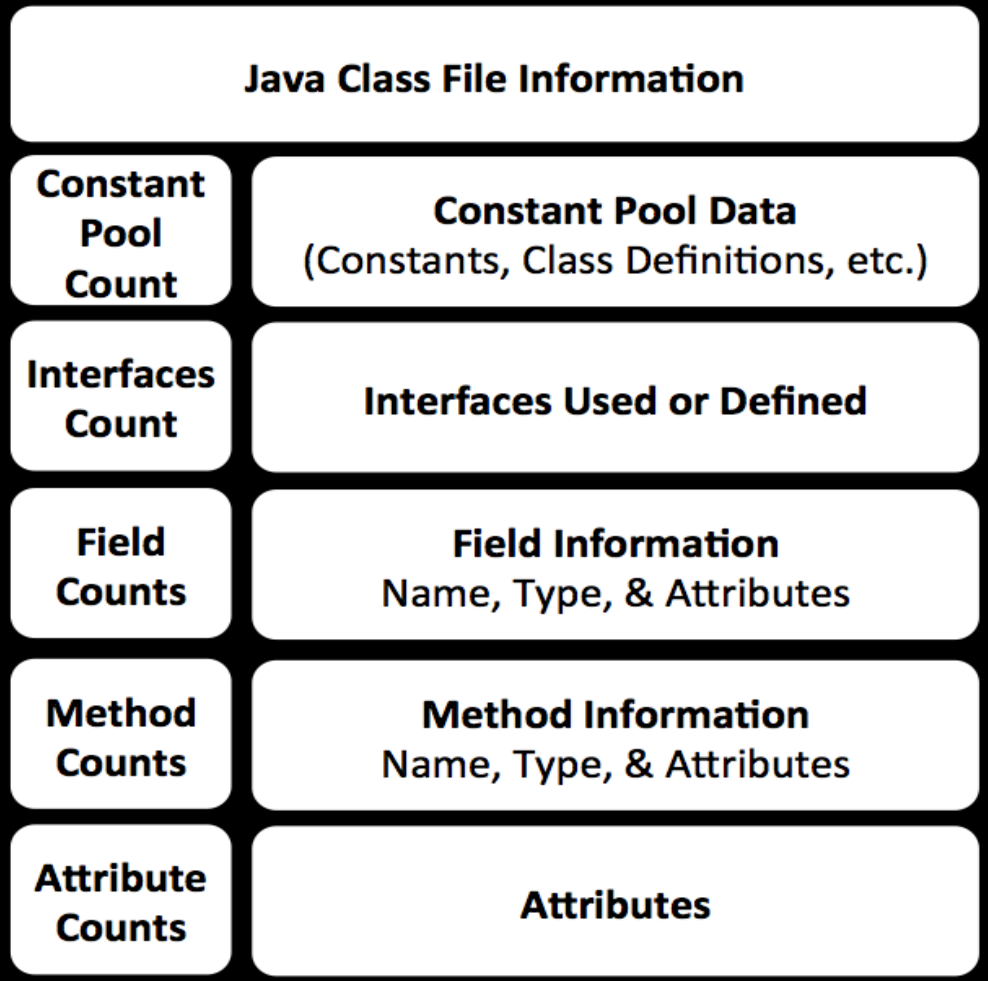
Class File Organization

- Constant Values
  - Long, Integers
  - Float, Doubles
  - Strings
- Class Definitions
- Field Definitions
- Method Definitions



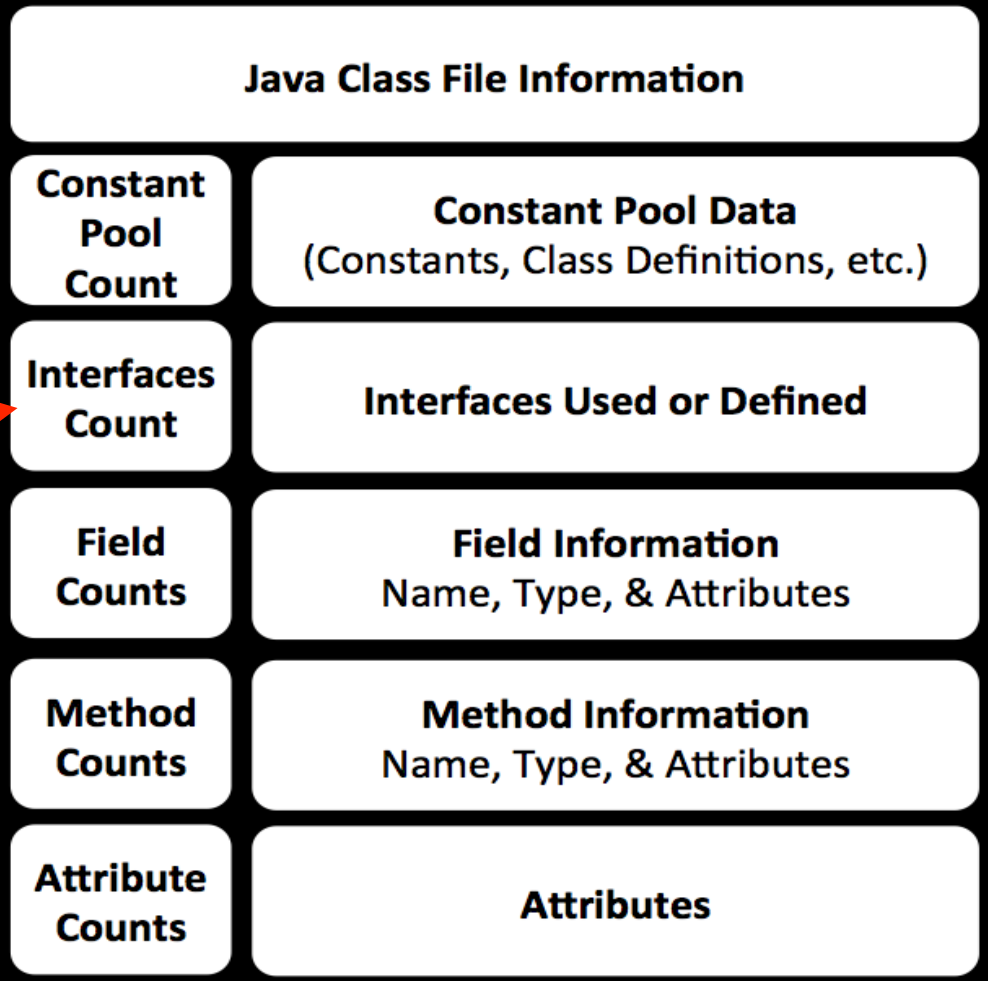
Class File Organization

- Omitted, but worth Mentioning
- Class Definition
- Super Class Info



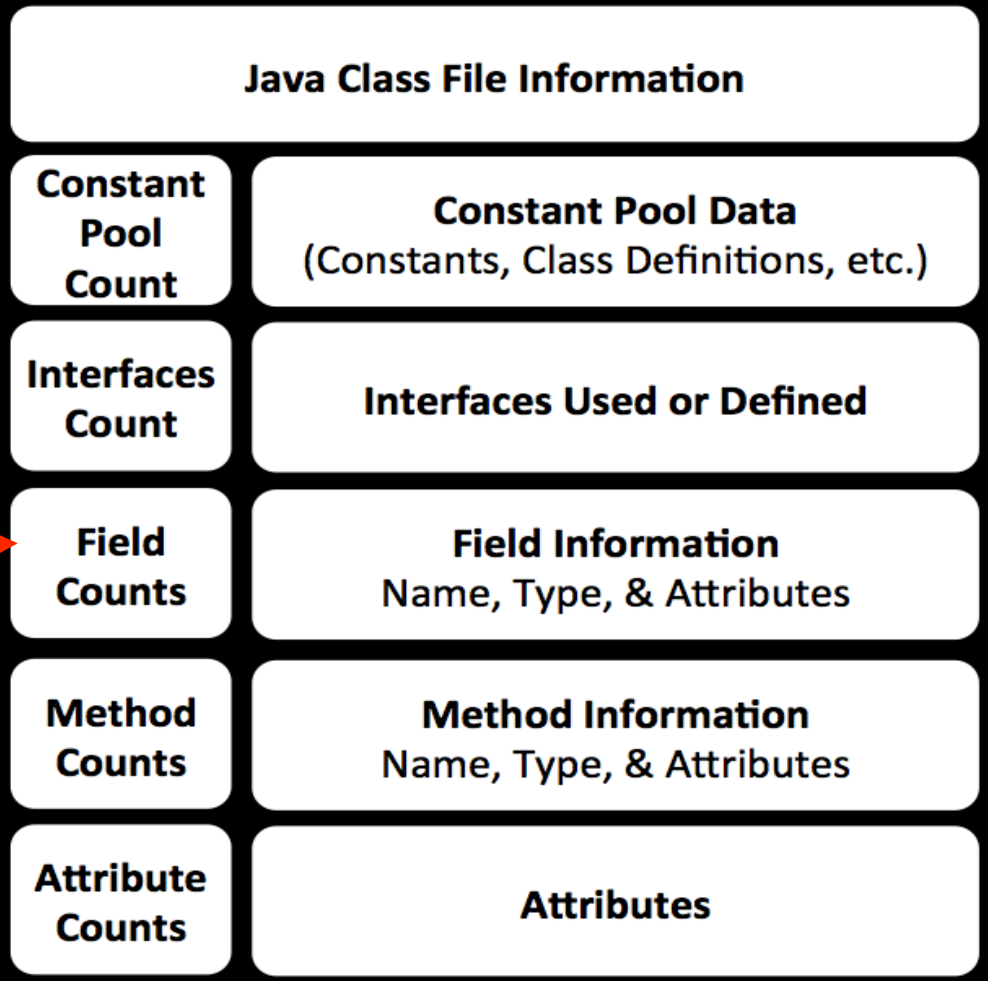
Class File Organization

- Interface Information



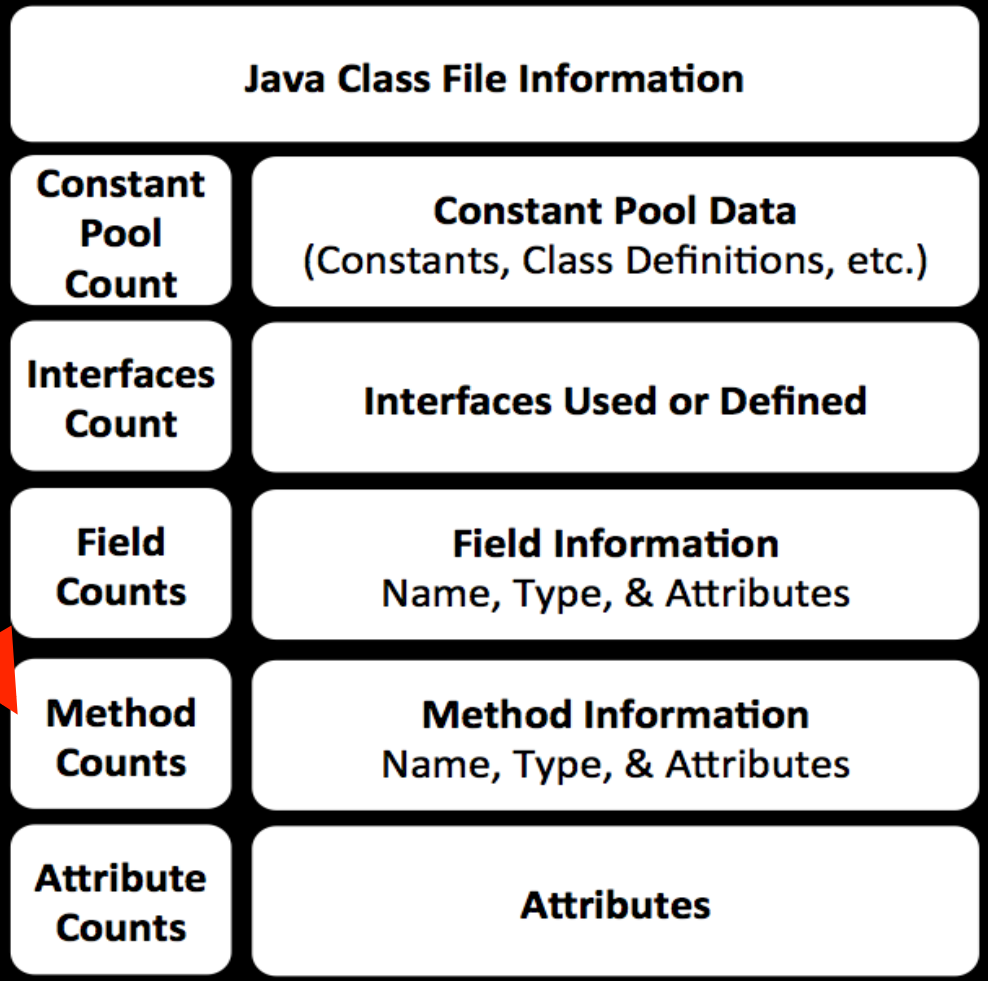
Class File Organization

- Access Flags
- Name and Description
- Attributes
  - Runtime Annotations
  - Constant Value



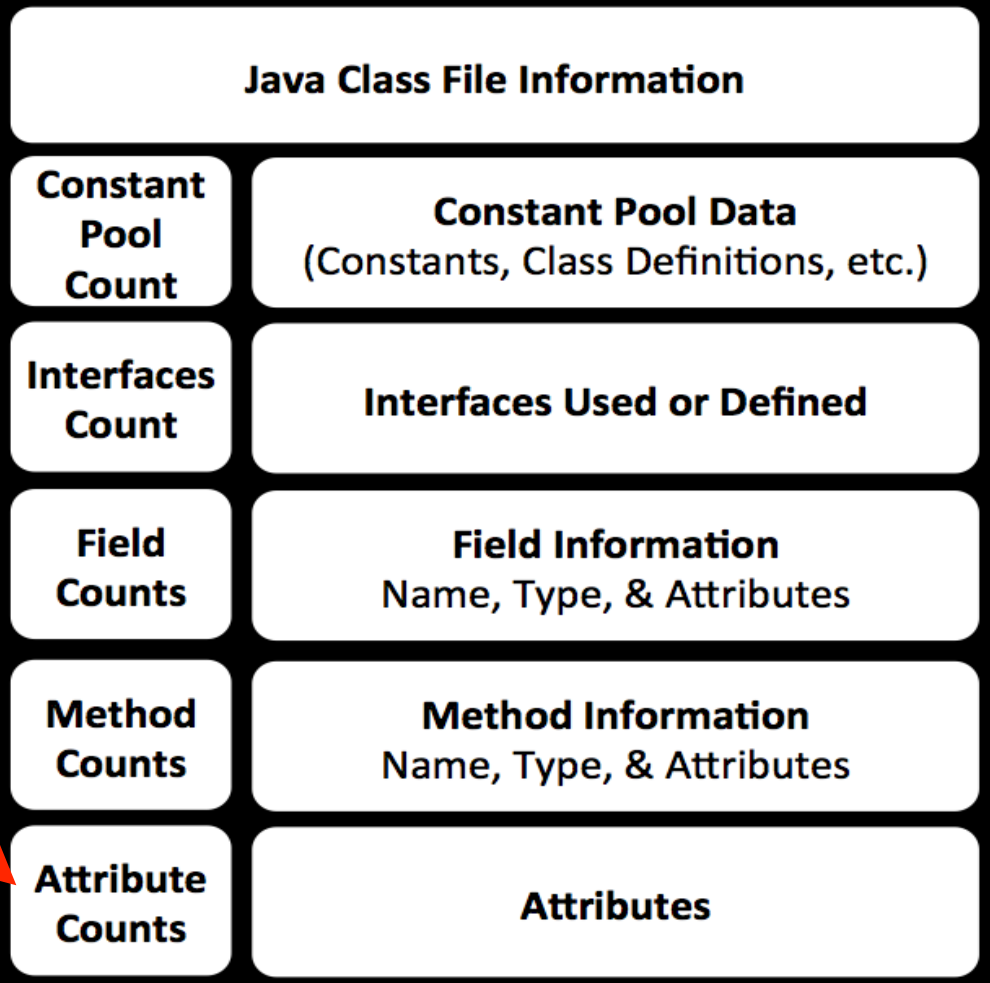
Class File Organization

- Access Flags
- Name and Description
- Attributes
  - Runtime Annotations
  - Code & Exceptions
  - Stack Map Table
  - Local Variable Tables
  - Inner Classes
  - ...



Class File Organization

- Class File Attributes
  - Runtime Annotations
  - Source File
  - User defined
  - ...



Hooking Java Methods

- Easiest all references to a class
  - Write an implementation that wraps the target class
  - Rewrite all of the strings
  - Modify access flags
  - Put the class in the class path
  - Run the JAR File

Hooking the Easy Way

Swap StringBuilder with sb class

```
bopeepsevilsheep:test_rewrite_mod apridgen$ r2 simple_hooking.class
 -- duck my sick!
[0x000003c1]> af
[0x000003c1]> ii
[Imports]
ordinal=008 plt=0x00000000 bind=NONE type=METHOD classname=java/lang/Object name=<i
ordinal=018 plt=0x00000000 bind=NONE type=METHOD classname=java/lang/StringBuilder
ordinal=021 plt=0x00000000 bind=NONE type=METHOD classname=java/lang/StringBuilder
ordinal=027 plt=0x00000000 bind=NONE type=METHOD classname=java/lang/StringBuilder
ordinal=034 plt=0x00000000 bind=NONE type=METHOD classname=java/lang/String name=va
ordinal=040 plt=0x00000000 bind=NONE type=METHOD classname=java/lang/StringBuilder
ordinal=043 plt=0x00000000 bind=NONE type=METHOD classname=java/lang/StringBuilder
ordinal=047 plt=0x00000000 bind=NONE type=INTERFACE_METHOD classname=java/lang/Syst
ordinal=053 plt=0x00000000 bind=NONE type=METHOD classname=java/io/PrintStream name
ordinal=073 plt=0x00000000 bind=NONE type=METHOD classname=simple_hooking name=dosc

10 imports

[0x000003c1]> il
[Linked libraries]
simple_hooking
java/lang/Object
java/lang/StringBuilder
java/lang/String
java/lang/System
java/io/PrintStream
java/lang/SecurityException
java/lang/NoSuchMethodException
javax/script/ScriptException
```

Note the use of StringBuilder

Hooking the Easy Way

Swap StringBuilder with sb class

```
bopeepsevilsheep:test_rewrite_mod apridgen$ r2 -w simple_hooking.class
 -- Emulate the base address of a file with e file.baddr
[0x000003c1]> java replace_classname_value java/lang/StringBuilder sb
result: java/lang/StringBuilder
result: java/lang/StringBuilder
result: java/lang/StringBuilder;
result: Ljava/lang/StringBuilder;
result: java/lang/StringBuilder;
result: Ljava/lang/StringBuilder;
result: java/lang/StringBuilder;
result: Ljava/lang/StringBuilder;
[0x0000024c]> il
[Linked libraries]
simple_hooking
java/lang/Object
sb
java/lang/String
java/lang/System
java/io/PrintStream
java/lang/SecurityException
java/lang/NoSuchMethodException
javax/script/ScriptException
```

Searching and rewriting StringBuilder

Hooking the Easy Way

```
[0x0000024c]> ii
[Imports]
ordinal=008 plt=0x00000000 bind=NONE type=METHOD classname=java/lang/Object name=<init> descriptor=
ordinal=010 plt=0x00000000 bind=NONE type=METHOD classname=sb name=<init> descriptor=()V
ordinal=011 plt=0x00000000 bind=NONE type=METHOD classname=sb name=append descriptor=(Ljava/lang/St
ordinal=012 plt=0x00000000 bind=NONE type=METHOD classname=sb name=append descriptor=(I)Lsb;
ordinal=013 plt=0x00000000 bind=NONE type=METHOD classname=java/lang/String name=valueOf descriptor
ordinal=014 plt=0x00000000 bind=NONE type=METHOD classname=sb name=<init> descriptor=(Ljava/lang/St
ordinal=015 plt=0x00000000 bind=NONE type=METHOD classname=sb name=toString descriptor=()Ljava/lang
ordinal=016 plt=0x00000000 bind=NONE type=INTERFACE_METHOD classname=java/lang/System name=out desc
ordinal=017 plt=0x00000000 bind=NONE type=METHOD classname=java/io/PrintStream name=println descrip
ordinal=019 plt=0x00000000 bind=NONE type=METHOD classname=simple_hooking name=dosomething descript
```

Swap StringBuilder with sb class

Hooking the Easy Way

ClassNotFound exception: 1

```
[0x0000024c]> q
bopeepsevilsheep:test_rewrite_mod apridgen$ java simple_hooking
Exception in thread "main" java.lang.NoClassDefFoundError: sb
        at simple_hooking.dosomething(simple_hooking.java:7)
        at simple_hooking.main(simple_hooking.java:19)
Caused by: java.lang.ClassNotFoundException: sb
        at java.net.URLClassLoader$1.run(URLClassLoader.java:202)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(URLClassLoader.java:190)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:306)
        at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:301)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:247)
        ... 2 more
```

Hooking the Easy Way

ClassNotFound exception: 2.

```
        ... 2 more
bopeepsevilsheep:test_rewrite_mod apridgen$ cp ../sb.class .
bopeepsevilsheep:test_rewrite_mod apridgen$ java simple_hooking
Exception in thread "main" java.lang.NoClassDefFoundError: s_
        at sb.log(sb.java:20)
        at sb.append(sb.java:79)
        at simple_hooking.dosomething(simple_hooking.java:8)
        at simple_hooking.main(simple_hooking.java:19)
Caused by: java.lang.ClassNotFoundException: s_
        at java.net.URLClassLoader$1.run(URLClassLoader.java:202)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(URLClassLoader.java:190)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:306)
        at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:301)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:247)
        ... 4 more
bopeepsevilsheep:test_rewrite_mod apridgen$ cp ../s_.class .
```

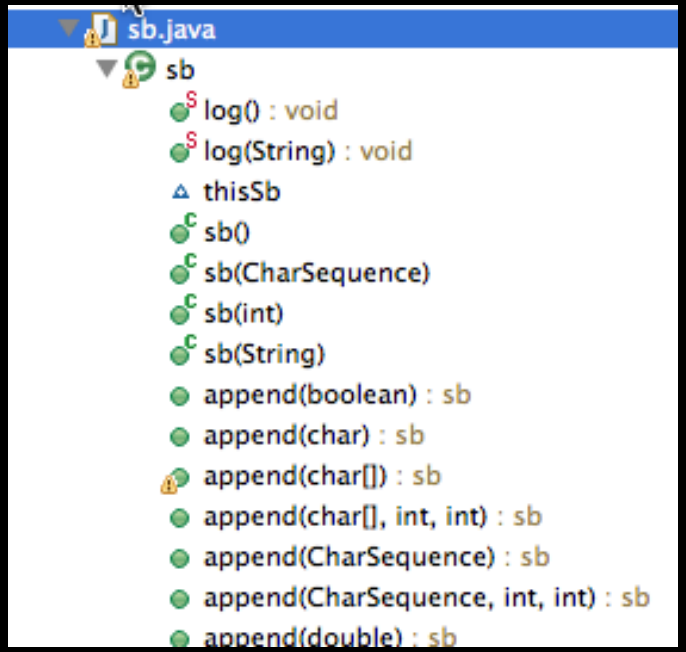
Hooking the Easy Way

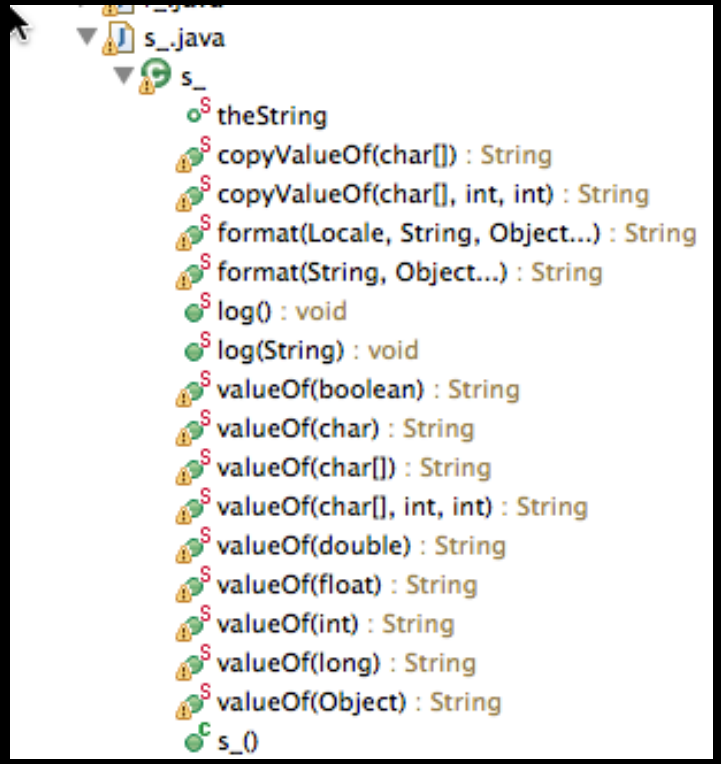
Copy classes to path and it works.

```
bopeepsevilsheep:test_rewrite_mod apridgen$ java simple_hooking
s_ calling appendHello
s_ calling appendI am a
s_ calling append100
s_ calling appendString.
s_ calling toString
s_ calling appendHello I am a 100String.
s_ calling toString
Darn tootin. Hello I am a 100String.
```

Hooking the Easy Way

Wrapper classes





Hooking Java Methods +1 Complexity

- Insert CP Objects
  - Append the CP Objects to define the new class
  - Class Info, Method Info, and Descriptor Info
  - Update the CP Object Counts
  - Modify code section and update the reference
  - Put the class in the class path
  - Run the JAR File

Primer Constant Pool Definition

```
class FooClass {
    String getItMethod ();
}
```

2	Tag = Method_Ref_Info	3	4
3	Tag = Class_Info	5	
4	Tag = NameandType	6	7
5	Tag = UTF8	7	FooClass
6	Tag = UTF8	11	getItMethod
7	Tag = UTF8	20	()Ljava/lang/String;

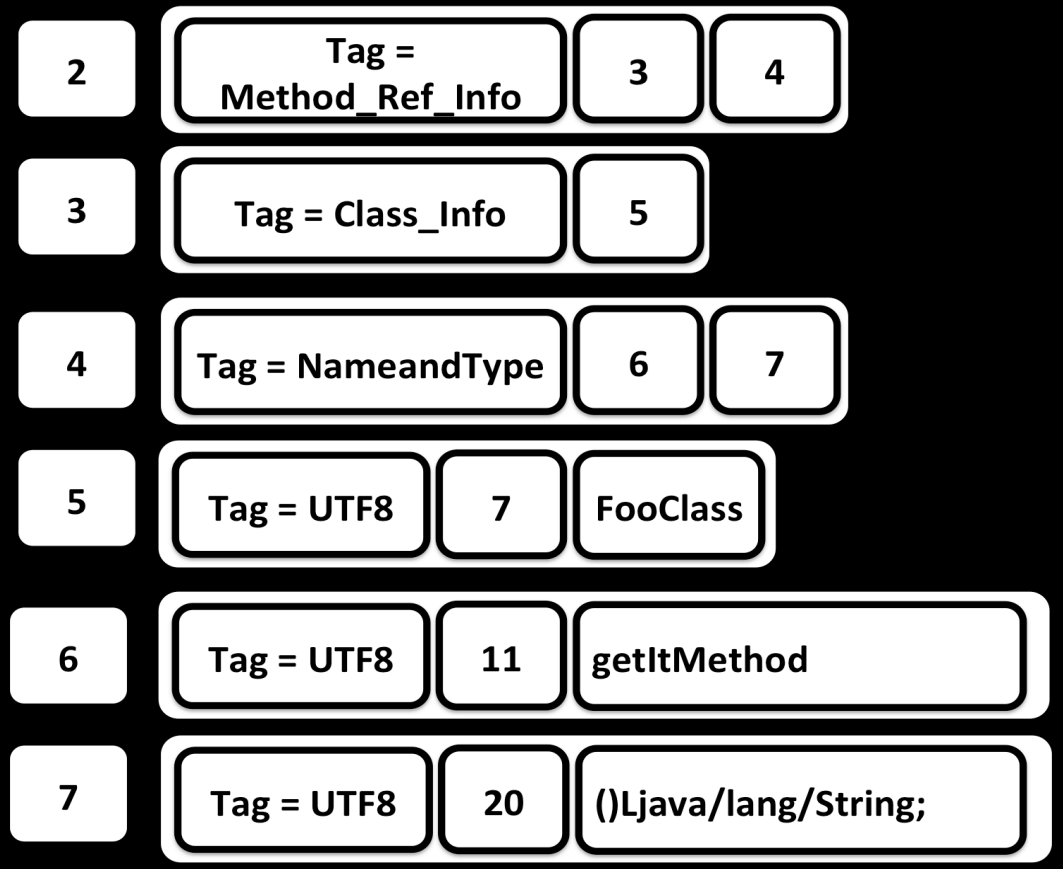
Primer Constant Pool Definition

Assume tag idx = 2

2	Tag = Method_Ref_Info	3	4
3	Tag = Class_Info	5	
4	Tag = NameandType	6	7
5	Tag = UTF8	7	FooClass
6	Tag = UTF8	11	getItMethod
7	Tag = UTF8	20	()Ljava/lang/String;

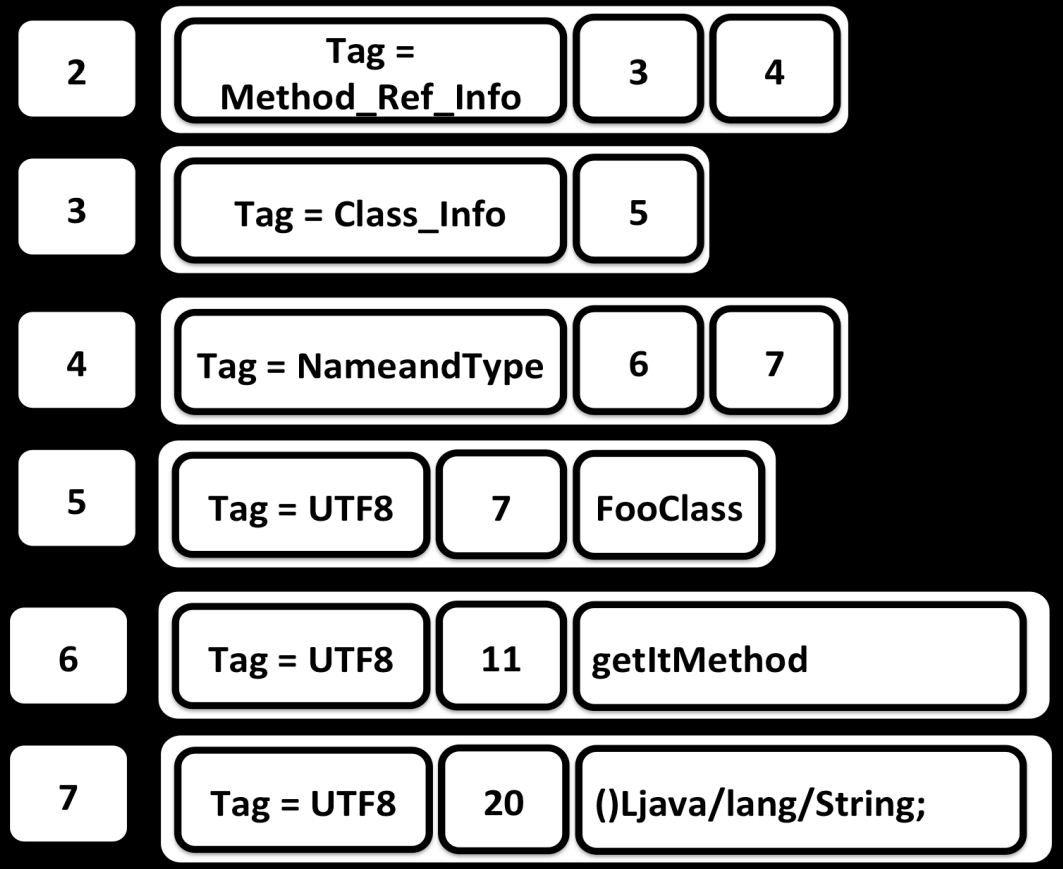
Constant Pool Definition

Resolving the
Class Name: FooClass



Constant Pool Definition

Resolving the Method Name: getItMethod



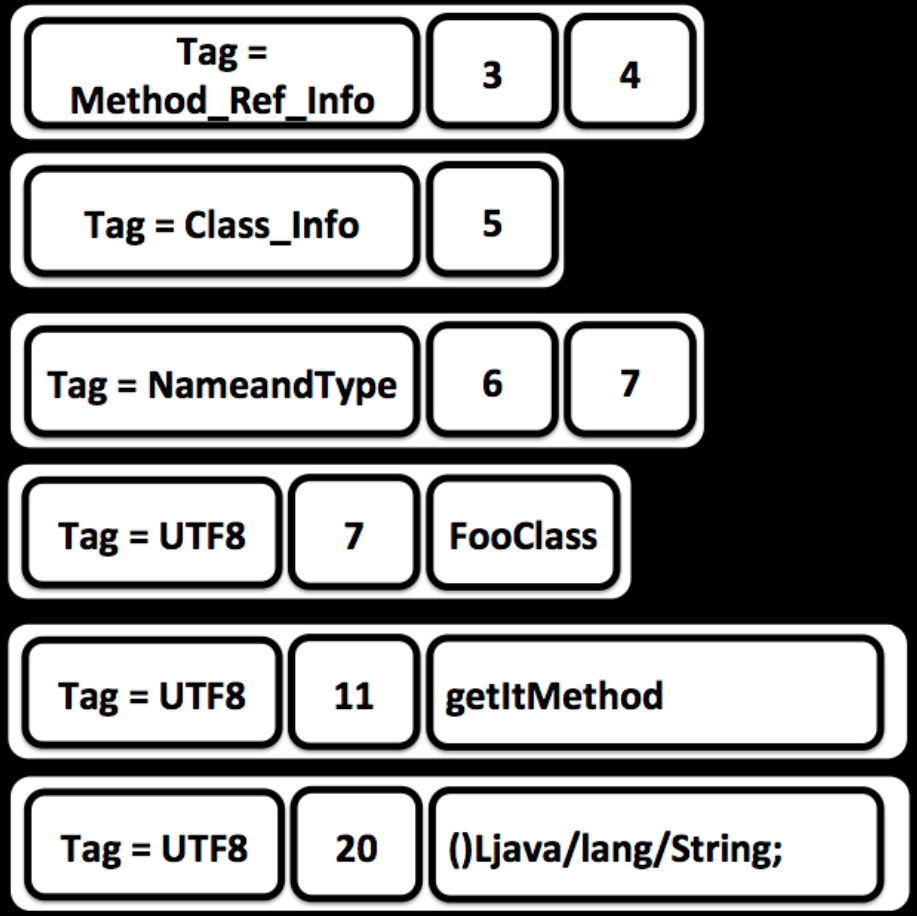
Constant Pool Definition

Resolving the Method Type: ()Ljava/lang/String;

Tag = Method_Ref_Info	3	4
Tag = Class_Info	5	
Tag = NameandType	6	7
Tag = UTF8	7	FooClass
Tag = UTF8	11	getItMethod
Tag = UTF8	20	()Ljava/lang/String;

Constant Pool Definition

```
class FooClass {
    String getItMethod ();
}
```



Hooking Java Methods ++1 Complexity

- Direct code insertion
  - Extend the code section attribute
  - Update attribute size
  - Modify code section and insert the code
  - Update the exception handling table

Changing Access Flags

Target Java Function: exploitAnnotations

```
 * deprecated eval( print(\ evaling in the eval!\\n\ ) );
 */
@MyAnnoOne (key="test", value="eval(\"print(\\\"evaling in the eval!\\n\\\")\");")
@JavScriptAnnot(javascript = "print(\"Hello world!\\n\");")
static public void exploitAnnotations () throws SecurityException, NoSuchMethodException, ScriptException {
    ScriptEngine se = new ScriptEngineManager()
    .getEngineByName("javascript");
    Annotation [] annotations = (rewrite.class).getMethod("exploitAnnotations").getAnnotations();
    for (Method method : rewrite.class.getMethods()) {
        //System.out.println(method.toString());
        if (method.isAnnotationPresent(Deprecated.class)) {
            for (Annotation anno : method.getDeclaredAnnotations()) {
                System.out.println("Annotation in Method '"
                        + method + "' : " + anno);
            }
```

Changing Access Flags

Insight is good, note the flag values.

```
[0x0000093d]> java m_info c
0 <init>
1 exploitAnnotations
2 dosomething
3 main
[0x0000093d]> java m_info s 1
Method Summary Information:
        File offset: 0x000008fa Access Flags: 9
        Name Index: 14 (exploitAnnotations)
        Descriptor Index: 6 (()V)
        Access Flags: 0x09 (public static)
        Method Attributes Count: 3
        Method Attributes:
Exceptions Attribute information:
   Attribute Offset: 0x00000902
   Attribute Name Index: 15 (Exceptions)
   Attribute length: 8
   Exceptions Attribute Index[0]: 16
   Exceptions Attribute Index[1]: 18
   Exceptions Attribute Index[2]: 20
```

Changing Access Flags

Apply some Radare Magic Sauce

```
[0x0000093d]> java flags_str_at m 0x8fa
Method Access Flags String: public static
[0x0000093d]> java calc_flags m static synthetic public
Access Value for static synthetic public = 0x1009
[0x0000093d]> s 0x8fa
[0x000008fa]> p8 2
0009
[0x000008fa]> java set_flags 0x8fa m static public synthetic
[0x000008fa]> p8 2
1009
[0x000008fa]>
```

Changing Access Flags

Here is what JD-Gui shows.

rewrite.class

```
import java.io.PrintStream;

public class rewrite
{
  static void dosomething(int a)
  {
    StringBuilder sv = new StringBuilder();
    sv.append("Hello ");
    sv.append("I am a ");
    sv.append(a);
    sv.append("String. ");
    String X = "Darn tootin. ";

    X = X + sv.toString();
    System.out.println(X);
  }

  public static void main(String[] args)
    throws SecurityException, NoSuchMethodException, ScriptException
  {
    dosomething(100);
    exploitAnnotations();
  }
}
```

exploitAnnotations went missing.

Changing Access Flags

```
0:experiment apridgen$ java -jar decompiler/cfr_0_78.jar rewrite/rewrite.class
/*
 * Decompiled with CFR 0_78.
 */
import JavScriptAnnot;
import MyAnno0ne;
import java.io.PrintStream;
import javax.script.ScriptEngine;
import javax.script.ScriptEngineManager
import javax.script.ScriptException

public class rewrite {
    @JavScriptAnnot(javascript="print(\"Hello world!\\n\");")
    public static /* synthetic */ void exploitAnnotations() throws SecurityExcepti
        ScriptEngine se = new ScriptEngineManager().getEngineByName("javascript");
        Annotation[] annotations = rewrite.class.getMethod("exploitAnnotations", n
        for (Method method : rewrite.class.getMethods()) {
            if (!method.isAnnotationPresent((Class)Deprecated.class)) continue;
            for (Annotation anno : method.getDeclaredAnnotations()) {
                System.out.println("Annotation in Method '" + method + "' : " + an
```

CFR Notes the Flag

Extracting jCrypt Classloader Key

List Files: zip://zip_file.whatevs
Access Files with: ::[index] or //path/

```
0:talk_stuff apridgen$ r2 zip://20fd2eec34cd1856a0c9e735d1914bf97d13162a869649afbcf5df3450a23b4c.jar

Files in archive

0 c.dat
1 META-INF/MANIFEST.MF
2 jcrypt/
3 jcrypt/Decrypter.class
4 jcrypt/EncryptedClassLoader.class
5 jar.dat
6 enc.dat
7 key.dat
```

Listing the files

Extracting jCrypt Classloader Key

List Files: zip://zip_file.whatevs

Access Files with: ::[index] or //path/

```
Segmentation fault: 11
0:talk_stuff apridgen$ r2 zip://20fd2eec34cd1856a0c9e735d1914bf97d13162a869649afbcf5df3450a23b4c.jar::3
 -- Calculate checksums for the current block with the commands starting with '#' (#md5, #crc32, #all, ..)
[0x00000df6]> q
Segmentation fault: 11
0:talk_stuff apridgen$ r2 zip://20fd2eec34cd1856a0c9e735d1914bf97d13162a869649afbcf5df3450a23b4c.jar//jcrypt/
 -- Switch between print modes using the 'p' and 'P' keys in visual mode
[0x00000df6]> q
Segmentation fault: 11
0:talk_stuff apridgen$
```

Extracting jCrypt Classloader Key

Loading /c.dat from the archive, whats that?

```
[0x00000df6]> s sym.jcrypt_Decrypter.
sym.jcrypt_Decrypter.main                    sym.jcrypt_Decrypter.mmeta_main
sym.jcrypt_Decrypter.getClassName            sym.jcrypt_Decrypter.mmeta_getClassName      sym.j
sym.jcrypt_Decrypter.mmeta_readString        sym.jcrypt_Decrypter.decode                  sym.j
sym.jcrypt_Decrypter.toDigit                 sym.jcrypt_Decrypter.mmeta_toDigit           sym.j
sym.jcrypt_Decrypter.mmeta_fromInputStream   sym.jcrypt_Decrypter.ENCRYPTED_ARCHIVE        sym.j
sym.jcrypt_Decrypter.EXCLUDE                 sym.jcrypt_Decrypter.DEFAULT_KEY
[0x00000df6]> s sym.jcrypt_Decrypter.main
[0x00000e3d]> pdf
/ (fcn) sym.jcrypt_Decrypter.main 227
|          0x00000e3d    1201         ldc jcrypt/Decrypter
|          0x00000e3f    120b         ldc "/c.dat"
|          0x00000e41    b60021       invokevirtual java/lang/Class/getResourceAsStream(Ljava/la
|             0x00000000(0x0, 0x0) ; section.constant_pool
|          0x00000e44    b80027       invokestatic jcrypt/Decrypter/readString(Ljava/io/InputSt
|             0x00000000() ; section.constant_pool
|          0x00000e47    b8002b       invokestatic jcrypt/Decrypter/decode(Ljava/lang/String;)L
|             0x00000000() ; section.constant_pool
|          0x00000e4a    b6002f       invokevirtual java/lang/String/trim()Ljava/lang/String;
|             0x00000000() ; section.constant_pool
|          0x00000e4d    1235         ldc "\x
|          0x00000e4f    b60037       invokevirtual java/lang/String/split(Ljava/lang/String;)[
|             0x00000000(0x0) ; section.constant_pool
```

Extracting jCrypt Classloader Key

Loading /c.dat from the archive, whats that?

```
[0x00000e3d]> o zip://20fd2eec34cd1856a0c9e735d1914bf97d13162a869649afbcf5df3450a23b4c.jar//c.dat 0x8000
[0x00000e3d]> o
- 11473 zip://20fd2eec34cd1856a0c9e735d1914bf97d13162a869649afbcf5df3450a23b4c.jar//jcrypt/Decrypter.clas
- 37766 zip://20fd2eec34cd1856a0c9e735d1914bf97d13162a869649afbcf5df3450a23b4c.jar//c.dat @ 0x8000 ; r
[0x00000e3d]> o 37766
[0x00008000]>
```

Extracting jCrypt Classloader Key

```
0x00008010  3634 3332 3437 3637 3637 3538 3330 3638  6432476767583068
0x00008020  3761 3435 3462 3537 3661 3666 3665 3733  7a454b576a6f6e73
0x00008030  ffff ffff ffff ffff ffff ffff ffff ffff  ................
0x00008040  ffff ffff ffff ffff ffff ffff ffff ffff  ................
0x00008050  ffff ffff ffff ffff ffff ffff ffff ffff  ................
0x00008060  ffff ffff ffff ffff ffff ffff ffff ffff  ................
0x00008070  ffff ffff ffff ffff ffff ffff ffff ffff  ................
0x00008080  ffff ffff ffff ffff ffff ffff ffff ffff  ................
0x00008090  ffff ffff ffff ffff ffff ffff ffff ffff  ................
0x000080a0  ffff ffff ffff ffff ffff ffff ffff ffff  ................
0x000080b0  ffff ffff ffff ffff ffff ffff ffff ffff  ................
0x000080c0  ffff ffff ffff ffff ffff ffff ffff ffff  ................
0x000080d0  ffff ffff ffff ffff ffff ffff ffff ffff  ................
0x000080e0  ffff ffff ffff ffff ffff ffff ffff ffff  ................
0x000080f0  ffff ffff ffff ffff ffff ffff ffff ffff  ................
:> ?x- `ps 50`
false
b
d2GggX0hzEKWjons
:>
```

Unhexlify Text

Extracting the Encrypted JAR File

```
0:talk_stuff apridgen$ r2 -w enc_jar.jar
 -- Set colors to your screen with 'e scr.color=true'
[0x00000000]> yfa zip://20fd2eec34cd1856a0c9e735d1914bf97d13162a869649afbcf5df3450a23b4c.jar::5
[0x00000000]> yy
[0x00000000]> p8 10
a2633402168867052125
[0x00000000]>
```

Open file to extract to, yank data (yfa) from file and paste it to the file.

```
In [1]: key = 'd2GggX0hzEKWjons'

In [2]: from Crypto.Cipher import AES

In [3]: cipher = AES.new (key)

In [4]: open('dec_jar.jar', 'wb').write (
   ...:         cipher.decrypt(open('en
%env         enc_jar.jar enumerate
   ...:         cipher.decrypt(open('enc_jar.jar', 'rb').read()))

In [5]:
```

Decrypt the JAR file with the key with Python.

Using Prototypes

```
0:talk_stuff apridgen$ r2 zip://dec_jar.jar

Files in archive

0 META-INF/MANIFEST.MF
1 a.class
2 b.class
3 c.class

Cannot open 'zip://dec_jar.jar'
```

Check out the classes with radare

Using Prototypes

```
0:talk_stuff apridgen$ r2 zip://dec_jar.jar

Files in archive

0 META-INF/MANIFEST.MF
1 a.class
2 b.class
3 c.class

Cannot open 'zip://dec_jar.jar'
```

Check out the classes with radare

```
[0x000004da]> java prototypes a
import java.lang.Enum;
import java.lang.String;
import java.lang.System;
import a;
import c;
import java.lang.NumberFormatException;
import [Ljava.lang.String;;
import [C;

class a { // @0x0000

        // Fields defined in the class
        public static final a a; // @0x0472
        public static final a b; // @0x047a
        public static final a c; // @0x0482
        public static final a d; // @0x048a
        public static final a e; // @0x0492
        public static final a f; // @0x049a
        public static final a g; // @0x04a2
        private static final synthetic a; h; // @0x04aa
        public static int i; // @0x04b2
        private static final java.lang.String; z; // @0x04ba

        // Methods defined in the class
        static void <clinit>(); // @0x04c4
        private void <init>(java.lang.String, int); // @0x08f4
        public static a a(java.lang.String); // @0x0921
        public static a a(); // @0x0b23
        public static a; values(); // @0x0b56
        public static a valueOf(java.lang.String); // @0x0b92
}
```

Dumping the Java Prototypes

Using Prototypes

```
[0x000004da]> java prototypes a
import java.lang.Enum;
import java.lang.String;
```

```
}
[0x000004da]> af
WARNING Analysis of sym.000004da Incorrect: Code Length: 0x204, Function size reported 0x1fb
Deadcode detected, setting code length to: 0x204
WARNING Analysis of sym.00000937 Incorrect: Code Length: 0xe3, Function size reported 0xdd
Deadcode detected, setting code length to: 0xe3
[0x000004da]> s sym.a.
sym.a._clinit_   sym.a.a          sym.a.values     sym.a.valueOf    sym.a.b
sym.a.c          sym.a.d          sym.a.e          sym.a.f          sym.a.g          sym.a.h
sym.a.i          sym.a.z
[0x000004da]> s sym.a.value
sym.a.values     sym.a.valueOf
[0x000004da]> s sym.a.valueOf
[0x00000ba8]> pdf
/ (fcn) sym.a.valueOf 10
|          0x00000ba8    1213          ldc a
|          0x00000baa    2a            aload_0
|          0x00000bab    b8001d        invokestatic java/lang/Enum/valueOf(Ljava/lang/Class;Lj
|             0x00000000(0x0, 0x0) ; section.constant_pool
|          0x00000bae    c00013        checkcast a
\          0x00000bb1    b0            arereturn
[0x00000ba8]>
```

Analyze and disassemble

Using Prototypes

a type is an Enum, created from the string this.a.z

```
0x00000632    bb0013      new a
0x00000635    59          dup
0x00000636    b20082      getstatic a/z [Ljava/lang/String;
0x00000639    1008        bipush 8
0x0000063b    32          aaload
0x0000063c    03          iconst_0
0x0000063d    b70023      invokespecial a/<init>(Ljava/lang/String;I)V
   0x00000000(0x0, 0x0, 0x0, 0x0) ; section.constant_pool
0x00000640    b30014      putstatic a/a La;
0x00000643    bb0013      new a
```

Using CFR Decompiler

CFR Decompiler to extract Java code

Problems with the Exception table?

[=] Lets dump it

```
private a(String string22, int string22) {
    super(string, n);
}

/*
 * Unable to fully structure code
 * Enabled aggressive block sorting
 * Enabled unnecessary exception pruning
 */
public static a a(String var0) {
    var2_1 = c.a;
    var0 = var0.toLowerCase();
    v0 = var0.equalsIgnoreCase(a.z[4]);
    if (!var2_1) {
        if (v0) {
            var1_2 = a.a;
            if (var2_1 == false) return var1_2;
            var3_3 = a.i;
            a.i = ++var3_3;
        }
        v0 = var0.equalsIgnoreCase(a.z[6]);
    }
    if (!var2_1) {
        if (v0) {
            var1_2 = a.b;
            if (var2_1 == false) return var1_2;
        }
        v0 = var0.contains((CharSequence)a.z[3]);
    }
```

CFR Decompiler Augmentation

Use prototypes:
‘java prototypes a’

Use exc:
‘java exc 0x937’

```
        // Methods defined in the class
        static void <clinit>(); // @0x04c4
        private void <init>(java.lang.String, int); // @0x08f4
        public static a a(java.lang.String); // @0x0921
        public static a a(); // @0x0b23
        public static a; values(); // @0x0b56
        public static a valueOf(java.lang.String); // @0x0b92
}
[0x000004da]> s 0x937
[0x00000937]> java exc 0x937
 Exception table for a (11 entries) @ 0x937:
        Catch Type: 115, java/lang/NumberFormatException @ 0xa22
                Start PC: (0xc1) 0x9f8 @ 0xa1c
                End PC: (0xce) 0xa05 0xa1e
                Handler PC: (0xd1) 0xa08 0xa20
        Catch Type: 115, java/lang/NumberFormatException @ 0xa2a
                Start PC: (0xb2) 0x9e9 @ 0xa24
                End PC: (0xb9) 0x9f0 0xa26
                Handler PC: (0xbc) 0x9f3 0xa28
        Catch Type: 115, java/lang/NumberFormatException @ 0xa32
```

Future Work

- Enable some more static conveniences
- Tie into a JVM for run-time information
- Enable code instrumentation via Code Attribute
- Look at reversing *native* code with JVM code
- Move on to other managed code implementations

Conclusion

- Discussed some basic constructs in Java classfile
- Introduced improvements to Radare
- Talked about how an analyst could use them

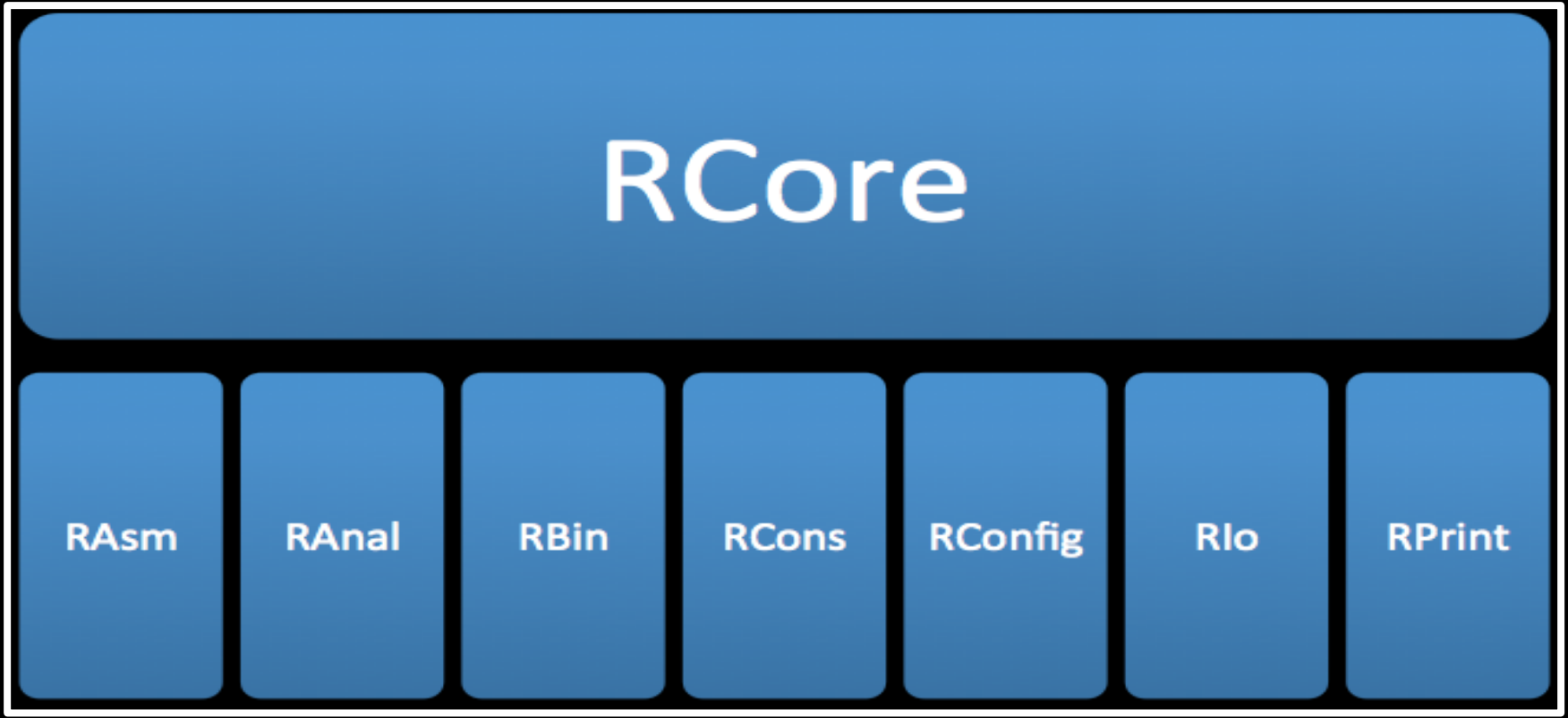
Questions and Contact Info

Thanks For Your Time.

email: adam.pridgen@thecoverofnight.com
twitter: @apridgen
github/bitbucket: deeso

Java Reversing Tools

Radare Architecture



Recent Additions to Radare

- Testing Framework
- Gameboy Reversing and Emulation
- Java Support
- Loading/reloading binaries from buffer
- Extending (inserting bytes in the middle)
- Opening multiple files
- Zip URI support