登录

Java沙箱逃逸走过的二十个春秋（一）

mss**** / 2018-10-02 12:09:29 / 浏览数 3708 技术文章 技术文章 顶(0) 踩(0)

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

--[1 – 引言

如今，Java平台已经广泛部署到了数十亿台的设备之上，其中，这些设备的类型包括服务器、桌面工作站和各种消费电子产品。不过话说回来，Java平台的设计初衷只是想实

在这篇文章中，我们回顾了Java安全情况的过去和现在。我们的目标是概述Java平台安全性是如何失败的，以便我们可以从过去的错误中吸取教训。这里介绍的所有安全漏洞

--[2 - 背景知识

----[2.1 - Java沙箱的漏洞简史

1995年，Sun
Microsystems公司发布了Java的第一个版本[2]。一年之后，普林斯顿大学的研究人员就发现了多个可以用来绕过沙箱的安全漏洞[3]。这些漏洞存在于语言、字节码和对象模

几年后，也就是2002年的时候，The Last Stage of
Delirium（LSD）研究小组提出了Java虚拟机安全性的研究报告[29]。他们详细介绍了影响字节码验证程序和类加载程序方面的漏洞，这些漏洞能够导致类型混淆或类欺骗攻

2012年，Guillardoy[5]描述了CVE-2012-4681问题，这两个漏洞允许攻击者绕过沙箱。第一个漏洞允许访问受限制的类，而第二个漏洞则允许修改私有字段。在同一年，C

2013年，Gorenc和Spelman对120个Java漏洞进行了大规模的研究，并得出结论：不安全的反射机制是Java中最常见的漏洞，而类型混淆则是最常被利用的漏洞[8]。同年，
Forshaw发现的可靠方法链漏洞，而CVE-2012-5088则是Security Explorations公司发现的Java反射漏洞。在2012年至2013年期间，Security
Explorations公司的安全研究人员发现了20多个Java漏洞[7]。

从2014年开始，主流的网络浏览器（如Chrome或Firefox）的开发人员决定默认禁用NAPI（因此，默认情况下，浏览器是无法执行任何Java代码的）[11]
[12]。此后，针对Java的攻击面开始减少，同时，针对Java沙箱绕过的研究也越来越难得。但是，沙箱绕过方面的漏洞仍然会时不时地蹦出一个来。例如，在2018年，Lee指

----[2.2 - Java平台

Java平台可以分为两个抽象组件：Java虚拟机（JVM）和Java类库（JCL）。

JVM是Java平台的核心，它是以本机代码来实现的，并为程序执行提供了所需的所有基本功能，例如字节码解析器、JIT编译器、垃圾收集器等。由于它是本机实现的，因此

JCL是JVM自带的一个标准库，含有数百个系统类，其中大部分都是用Java语言实现的，只有很少一部分是本机实现的。由于所有系统类都是可信任的，因此，默认情况下，

因此，本文的主要内容分为两大部分：其中一部分介绍内存破坏漏洞，另一部分则侧重于Java级别的漏洞。

----[2.3 - Java安全管理器

在JCL的代码中，沙箱是通过授权检查的方式实现的，其中大多数都是权限检查。例如，在访问文件系统之前，JCL中的代码会检查调用者是否具有访问文件系统的权限。下面

```
1: public FileInputStream(File file) throws FileNotFoundException {
 2:     String name = (file != null ? file.getPath() : null);
 3:     SecurityManager security = System.getSecurityManager();
 4:     if (security != null) {
 5:         security.checkRead(name);
 6:     }
 7:     if (name == null) {
 8:         throw new NullPointerException();
 9:     }
10:     if (file.isInvalid()) {
11:         throw new FileNotFoundException("Invalid file path");
12:     }
13:     fd = new FileDescriptor();
14:     fd.incrementAndGetUseCount();
15:     this.path = name;
16:     open(name);
17: }
```

请注意，出于性能原因的考虑，仅在设置了安全管理器时才会检查授权情况（第3-4行）。因此，Java沙箱逃逸方面的典型攻击，通常目标就是将安全管理器设置为null。这

但是，检查授权仅适用于Java级别。因为本机代码在运行时会被授予全部权限。虽然安全分析人员在利用内存损坏漏洞时，有时可以直接运行受自己控制的本机代码，但在

----[2.4 - doPrivileged方法

当检查权限"P"时，JVM将会检测调用堆栈的每个元素是否都具有权限"P"。只要有一个元素没有"P"权限，它就会抛出安全异常。对于这种方法来说，在大部分情况下都是有

对于上面的问题，解决办法是把m1()内对于m2()的调用封装到doPrivileged()调用中。因此，当检查"P2"时，堆栈遍历停止在调用doPrivileged()的方法处，这里是m1()。由

这方面的一个实际例子是_java.nio.Bits_类中的unaligned()方法。它是用来处理网络流的，并且必须知道处理器的体系结构。但是，获取该信息所需要的"get_property"权限

```
1: static boolean unaligned() {
 2:     if (unalignedKnown)
 3:         return unaligned;
 4:     String arch = AccessController.doPrivileged(
 5:         new sun.security.action.GetPropertyAction("os.arch"));
 6:     unaligned = arch.equals("i386") || arch.equals("x86")
 7:         || arch.equals("amd64") || arch.equals("x86_64");
 8:     unalignedKnown = true;
 9:     return unaligned;
10: }
```

当检查"get_property"权限时，堆栈遍历将开始检查各个方法，直至bit.unaligned()方法，然后停止检查。

小结

由于原文篇幅较长，为了让译文及时与读者见面，我们将分段发表。在本文中，回顾了Java沙箱的漏洞简史，介绍了Java平台的两个基本组成部分，同时，还讲解了Java安全