Apache ActiveMQ Artemis 反序列化—【CVE-2016-4978】

orich1 / 2018-02-05 09:45:00 / 浏览数 5346

漏洞描述

The getObject method of the javax.jms.ObjectMessage class in the (1) JMS Core client, (2) Artemis broker, and (3) Artemis REST component in Apache ActiveMQ Artemis before 1.4.0 might allow remote authenticated users with permission to send messages to the Artemis broker to deserialize arbitrary objects and execute arbitrary code by leveraging gadget classes being present on the Artemis classpath.

Apache ActiveMQ Artemis

ActiveMQ Artemis 是 ActiveMQ 的子项目，而ActiveMQ 是Apache出品，最流行的，能力强劲的开源消息总线。ActiveMQ 是一个完全支持JMS1.1和J2EE 1.4规范的 JMS Provider实现，尽管JMS规范出台已经是很久的事情了，但是JMS在当今的J2EE应用中间仍然扮演着特殊的地位。

触发流程

分析过程

根据描述中讲的，三个漏洞点，我们就去看下第一个 JMS Core client

但是里面的源码文件实在是太多了....又不熟悉 ActiveMQ 的原理，再仔细看下漏洞描述
因为是和 getObject 这个函数有关，而函数在 javax.jms.ObjectMessage 中，那么就从这里入手

一个只有 set 和 get 操作的接口，此时就可以想想了
如果跟进 getObject
的实现类，那么很快就能找出漏洞的触发点，但是可能会找不出触发的处理流程，比如到底是如何接受数据之类的，如果说能够站在了解这一架构的情况下，这点是可以忽略
那么跟进 setObject 呢，仅仅是 setObject 跟下去的话，可能找不出触发点，但是对于整个流程的起始应该能跟出来，比如如何生成 payload
到向目标发包的整个过程

我这里肯定是先跟的 get 后跟的 set，但是一般源码里会有 test 示例，所以如果直接跟 set 的话会很快跟到 test
中，运气好的话，对于整个流程很快有一个大致的概念

在文章里就先从 set 操作跟起

跟进 artemis-jms-client 模块里

一个很简单的序列化操作，并将其结果赋值给类成员 data
继续跟，找当前类的 setObject 的使用者，应该可以找到 test 示例

ObjectMessageTest 名字都这么粗暴，跟进去看看

找到类中的一个测试函数： testClassLoaderIsolation
里面的逻辑看着很简洁

后面的 assert 可以不用管了，一看就是比较接收到的 testObject2 和源数据 testObject 的差别

照着它前面的这个思路来
首先是一些准备工作，然后注意到使用 om 引用 message 做了 setObject 的操作，放入的是一个 SomeObject 的对象，然后调用了 queueProd 成员的 send 函数

接着就是用 queueCons 成员的 receive 操作接受相应的 Message ，最后就是 getObject ，getObject 我们先不看，先跟进 send 函数

同样还是选择 client 的模块

关注点在 mesage 上，继续跟进

这个 doSendx 的函数体过长，只截取关键部分

如上图，如果 jmsMessage（就是之前的 message ）是外来之物的话，那么就对其包装一下
否则直接赋值给 activeMQJmsMessage

现在关注 activeMQJmsMessage

这里是比较关键的一步，跟进去

之前第一步就是跟入的 ActiveMQObjectMessage ，所以这里也同样的选中它

如上图，将 data 中的数据写入到 message 成员中

现在回到 doSendx 函数中，最后的处理逻辑如下

上图里将 activeMQJmsMessage 做了一次 getCoreMessage 处理，并将其处理结果送入到 clientProducer 的 send 函数中（无论是否 if 成立，都会传入 send 函数中）

那么看一下 getCoreMessage 做了啥

很简洁，将之前已经带有 data 数据的 message 返回

那么也就是说，之前已经序列化的 SomeObject 已经带入了 message 中，并且返回给了 coreMessage ，最后还将其 send 出去
(更具体的操作就不跟进了，因为太底层也搞不动了....比如具体 jms 指定发送信息的目的地、底层包装信息和发包等过程)

现在就当带有 SomeObject 的 message 已经发出，然后回到测试类 ObjectMessageTest 中
查看 receive 操作

选红框里的类

继续跟

getMessage 函数体也略长，只贴出关键代码

```
private ActiveMQMessage getMessage(final long timeout, final boolean noWait) throws JMSException {
  try {
     ClientMessage coreMessage;

     if (noWait) {
        coreMessage = consumer.receiveImmediate();
     }
     else {
        coreMessage = consumer.receive(timeout);
     }

     ActiveMQMessage jmsMsg = null;

     if (coreMessage != null) {
        boolean needSession = ackMode == Session.CLIENT_ACKNOWLEDGE || ackMode == ActiveMQJMSConstants.INDIVIDUAL_ACKNOWLEDGE;
        jmsMsg = ActiveMQMessage.createMessage(coreMessage, needSession ? session.getCoreSession() : null);

        try {
           jmsMsg.doBeforeReceive();
        }
        catch (IndexOutOfBoundsException ioob) {
   [....]
        }

[....]

     }

     return jmsMsg;
  }
  catch (ActiveMQException e) {
  [....]
  }
```

上面的代码中，又使用了 ClientMessage 这个类，并将 receive 的结果返回给它，对比一下之前的 doSendx 里的操作，正是将 ClientMessage 作为信息载体发出到目的地的

jmsMsg 也是用 ClientMessage 作为参数生成的 ActiveMQObjectMessage 类

那么在doBeforeReceive 操作时，jmsMsg 已经是 ActiveMQObjectMessage 的对象，跟进去看看

这里是将 message 中所带的数据写入 data 中

如果没有错误，最后将 jmsMsg 返回

回到 testClassLoaderIsolation 中， r 就是 jmsMsg，里面带有 message 和 data 成员变量

继续跟进 r.getObject

将 data 中的数据反序列化，这里就触发了

因为是 ObjectMessage 的 getObject 函数的问题，其他两处的流程也都差不多

漏洞相关信息：https://nvd.nist.gov/vuln/detail/CVE-2016-4978
ActiveMQ 相关信息：https://www.cnblogs.com/Peter2014/p/8080192.html

点击收藏 | 0 关注 | 1

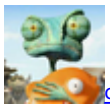
上一篇：Discuz!因Memcached... 下一篇：【代码审计】某开源商城前台Gets...

1. 2 条回复


assassinator 2018-02-09 10:23:06

这个代码再哪里能下载

0 回复Ta


orich1 2018-02-14 03:23:14

@assassinator github上有镜像

0 回复Ta
