OLYMPUS - WriteUp

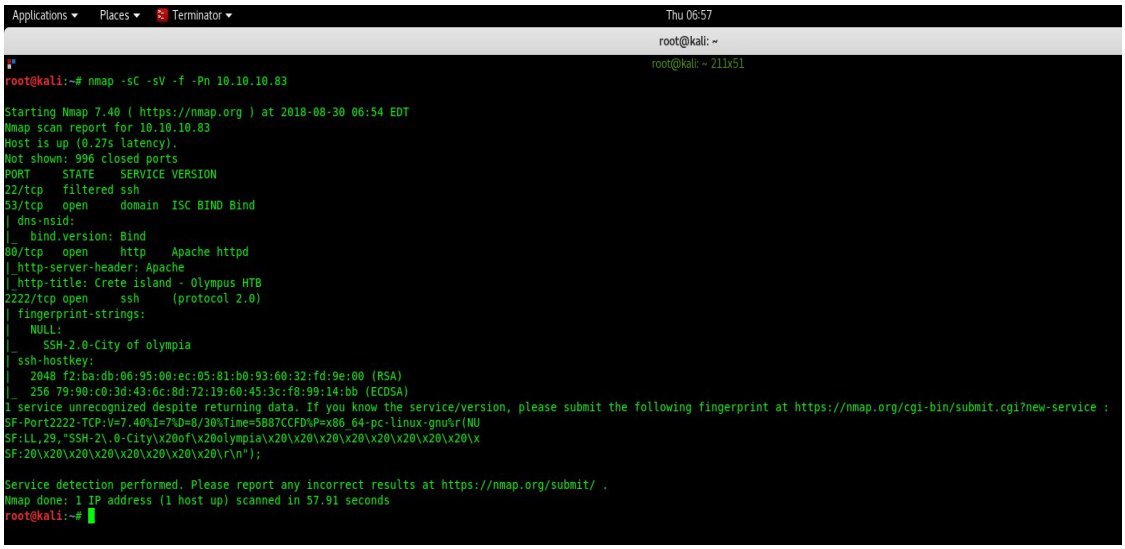
Author : - 3z culprit

Hello culprits,

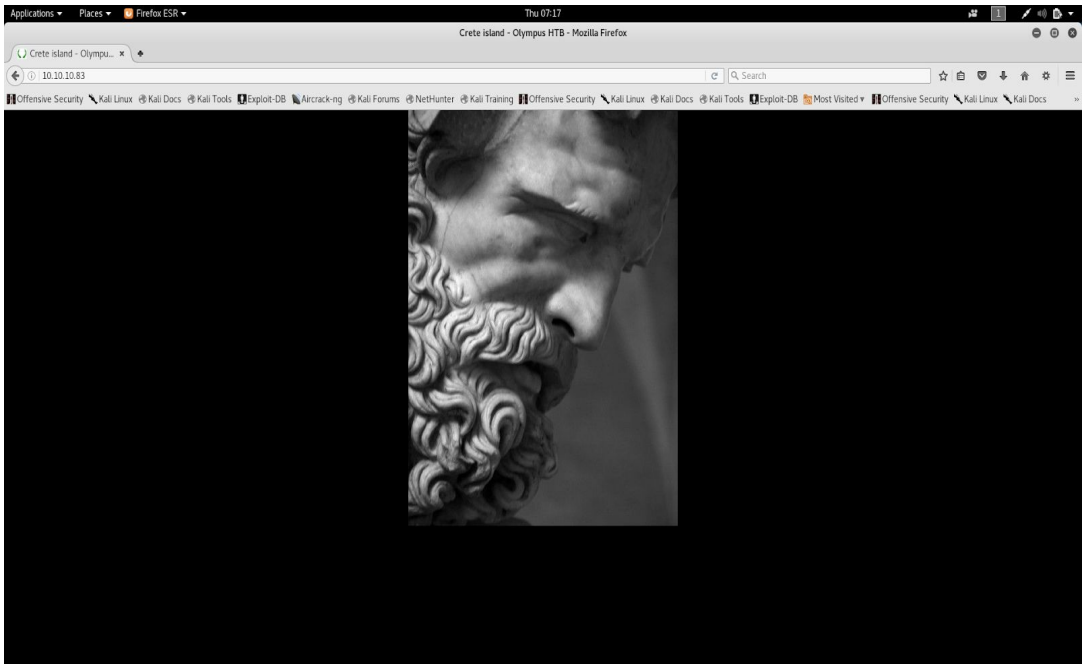
The following write-up will cover the intrinsics of rooting Olympus machine present on HTB.

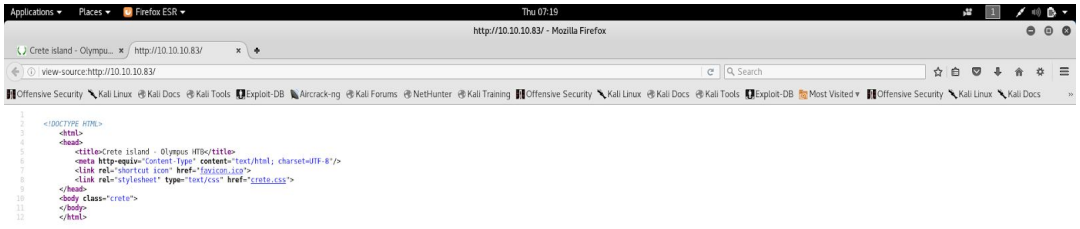
Machine IP : 10.10.10.83
Machine type : Linux

1) **Initial Foothold**: We start with our basic nmap command to find out the open ports , services. Here it goes:-

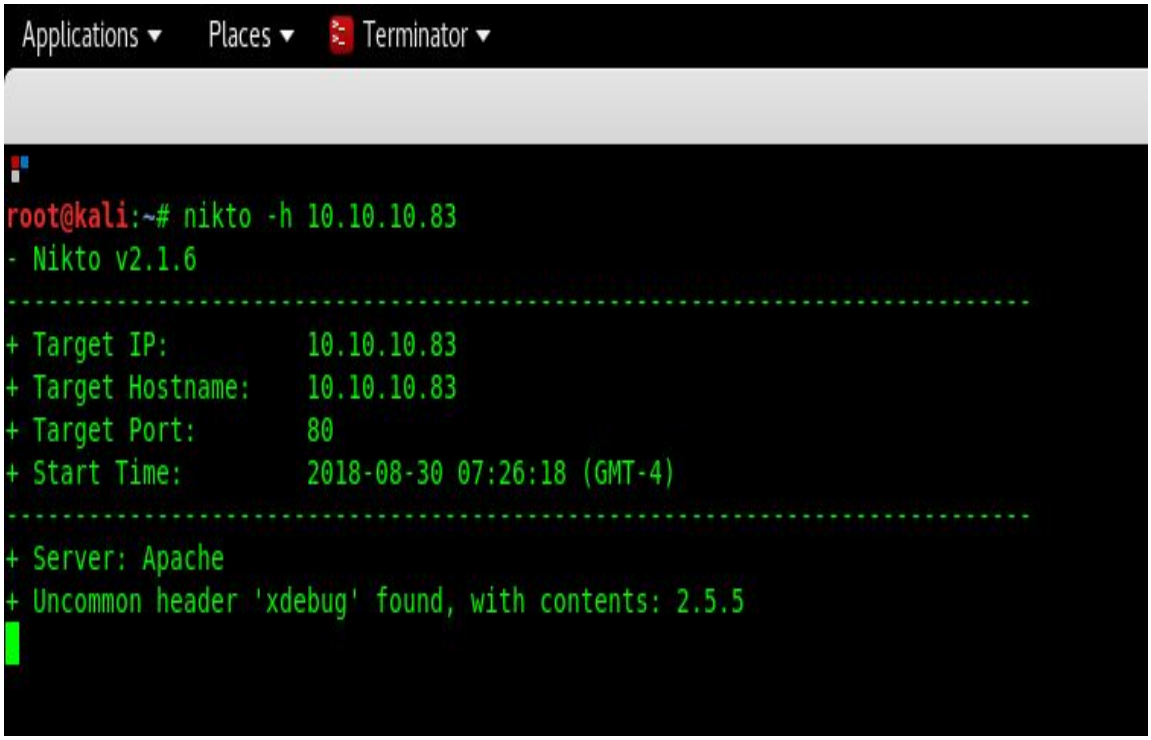


We see from the nmap scan that there is a **port 22** which is used for SSH but it is filtered, so we can assume that after performing some actions it will get opened. Next we have **port 53** opened which is domain bind . Other ports are , **port 80** which has http server running and webpage is hosted with title **Crete Island - Olympus HTB**. Last we have **port 2222** which is another Port for SSH with title as City of Olympia which is a stage of challenge. Now we Navigate to port 80 which shows us a picture of zeus. Here it is:-





We see that there is not much information present on the source code and the picture did not give us any information to move forward. Since it is a webpage let's try running nikto to see any vulnerability present itself in response. Here it goes:-



It gives out more information but for now the “uncommon header ‘xdebug’ ” gives us a little direction to move forward. A quick google search lists out the vulnerability related to xdebug version 2.5.5 . For more depth of how the RCE works , refer to :- XDEBUG-RCE .

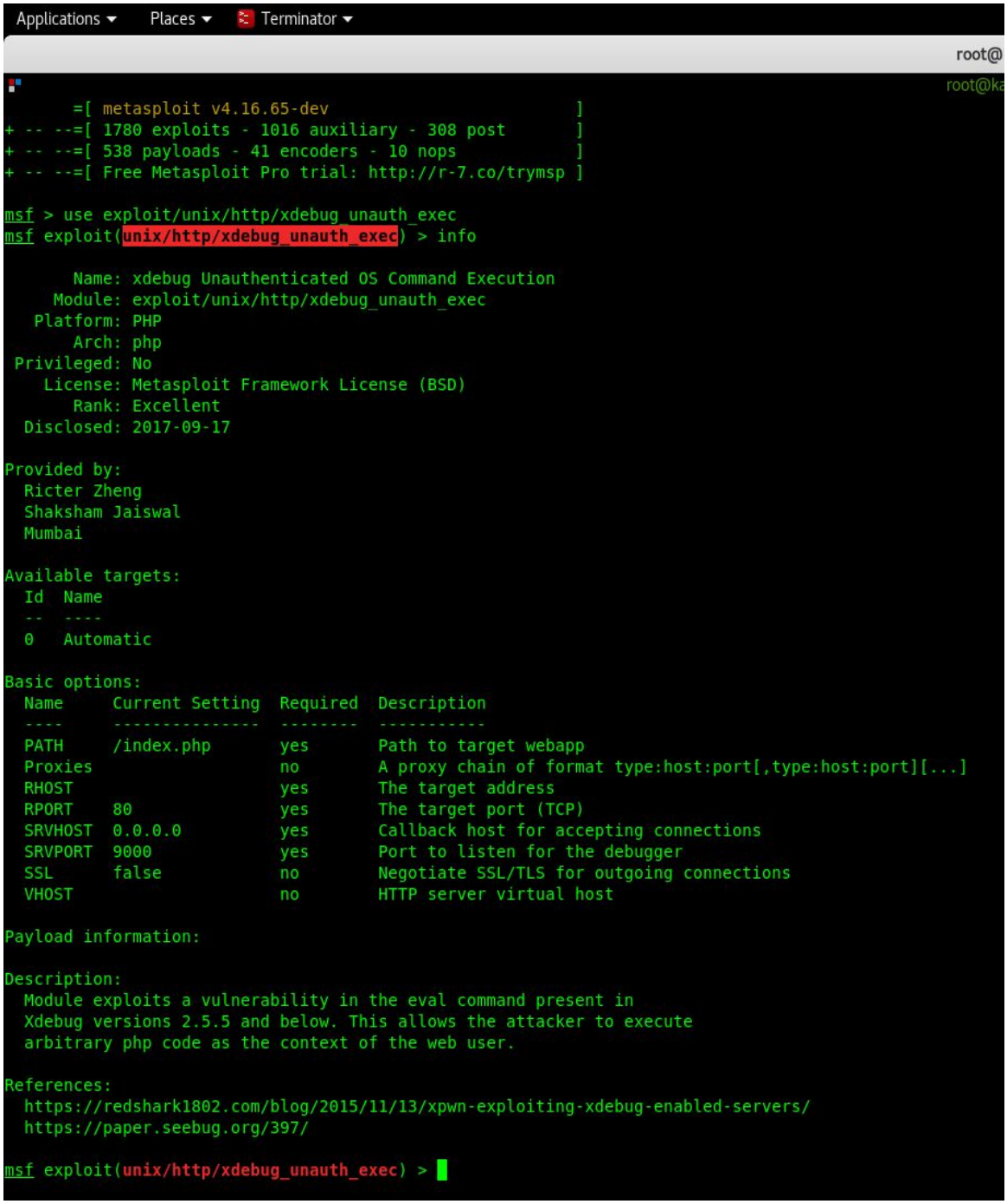
Form this moment we have two paths , either we use Metasploit module :-

-> **exploit/unix/http/xdebug_unauth_exec**

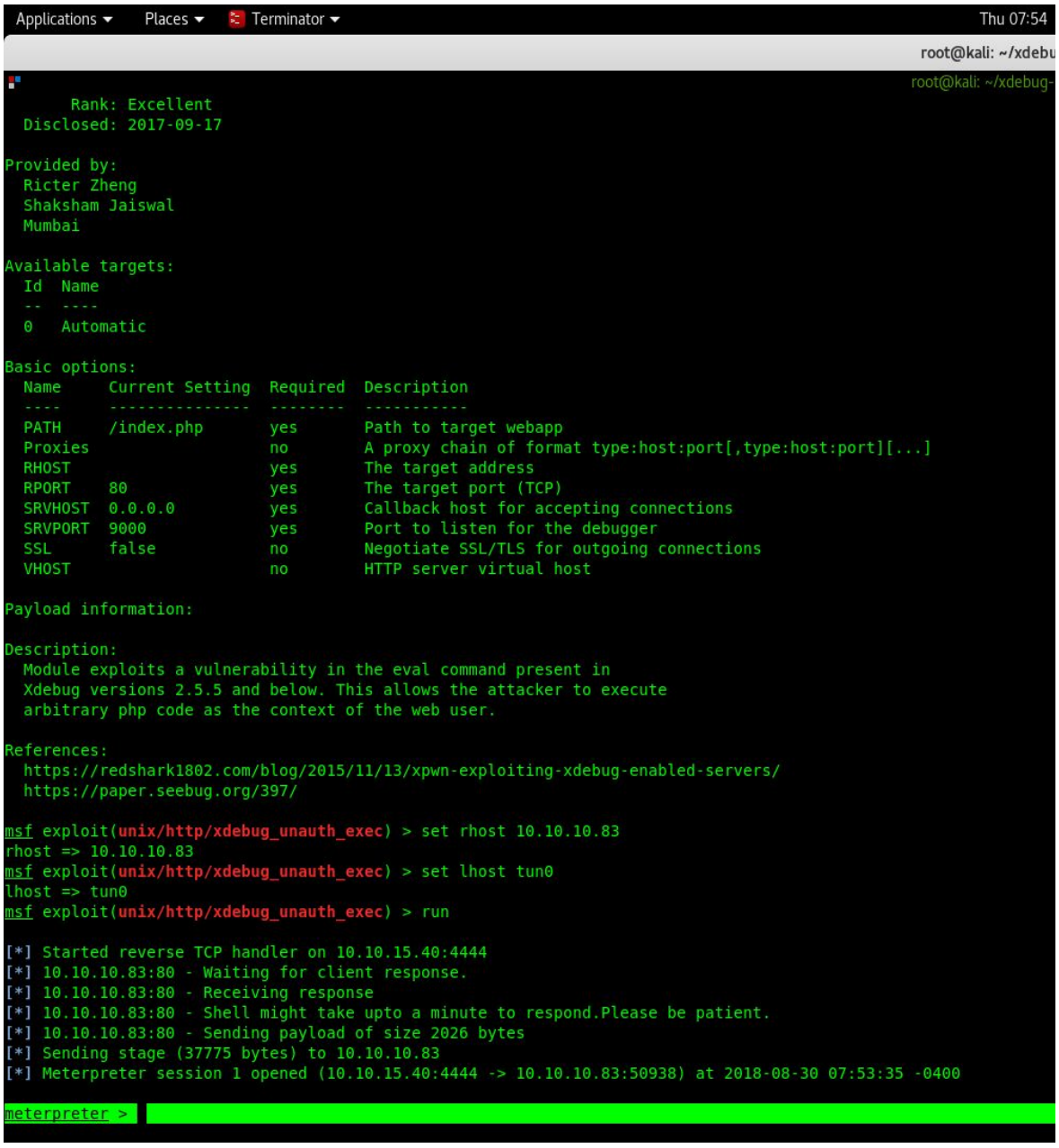
Or we can use the script hosted at :-

-> **https://github.com/gteissier/xdebug-shell**

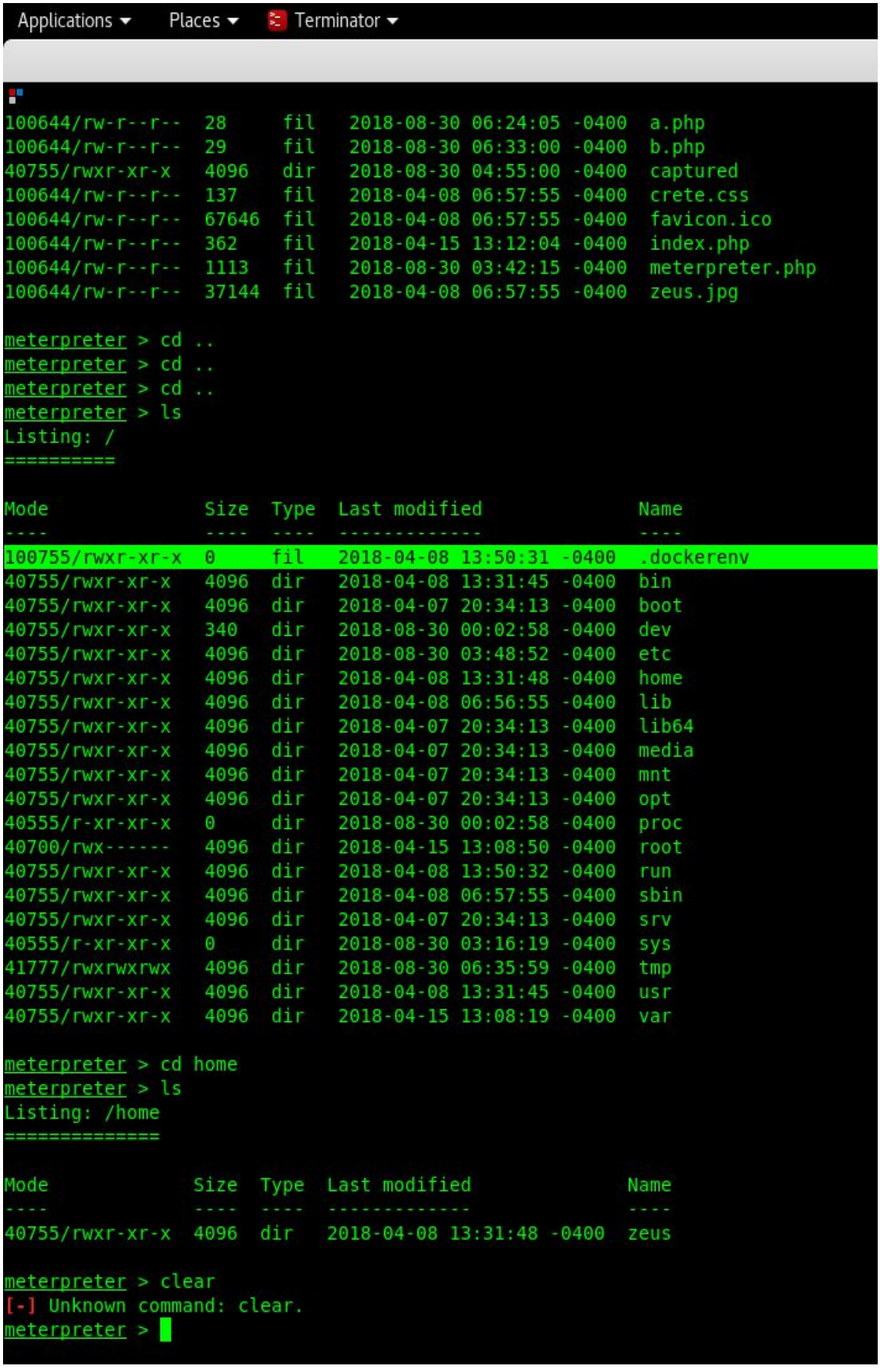
We can use either of the methods , in this write-up let's move ahead with Metasploit . Fire up msfconsole and use the module. Here it goes:-



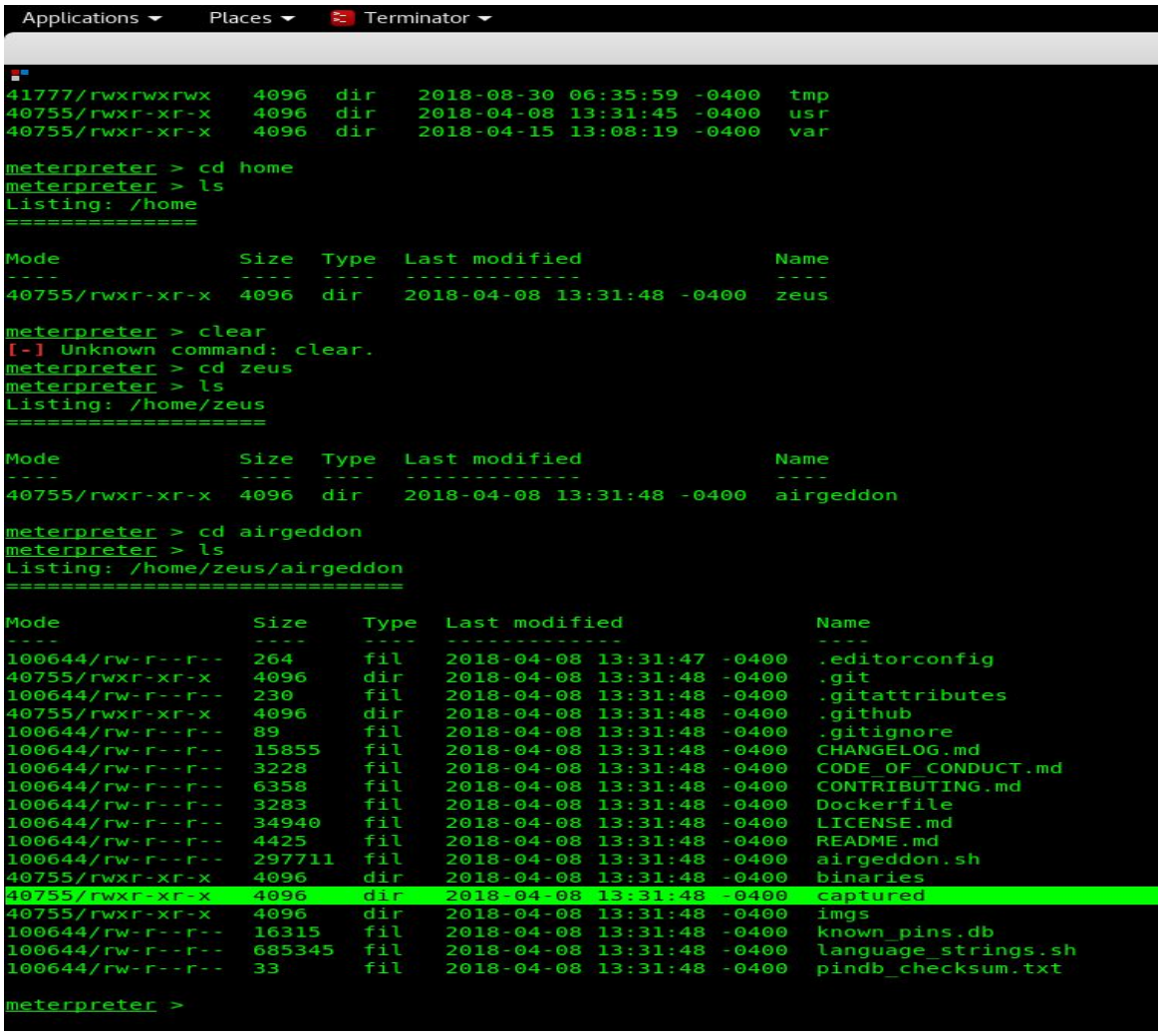
Fill in the required fields for RHOST, LHOST and we are good to go. Let’s exploit this :-



And we have our meterpreter session. After some enumeration we can figure out that we are inside a docker container **(Docker is a program that performs operating-system-level virtualization , also known as “containerization”)** by looking at the following file:- **(.dockerenv)**


Applications
Places
Terminator
100644/rw-r--r-- 28 fil 2018-08-30 06:24:05 -0400 a.php
100644/rw-r--r-- 29 fil 2018-08-30 06:33:00 -0400 b.php
40755/rwxr-xr-x 4096 dir 2018-08-30 04:55:00 -0400 captured
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-- 1113 fil 2018-08-30 03:42:15 -0400 meterpreter.php
100644/rw-r--r-- 37144 fil 2018-04-08 06:57:55 -0400 zeus.jpg
meterpreter > cd ..
meterpreter > cd ..
meterpreter > cd ..
meterpreter > ls
Listing: /
==========
Mode Size Type Last modified Name
---- ---- ---- ------------- ----
100755/rwxr-xr-x 0 fil 2018-04-08 13:50:31 -0400 .dockerenv
40755/rwxr-xr-x 4096 dir 2018-04-08 13:31:45 -0400 bin
40755/rwxr-xr-x 4096 dir 2018-04-07 20:34:13 -0400 boot
40755/rwxr-xr-x 340 dir 2018-08-30 00:02:58 -0400 dev
40755/rwxr-xr-x 4096 dir 2018-08-30 03:48:52 -0400 etc
40755/rwxr-xr-x 4096 dir 2018-04-08 13:31:48 -0400 home
40755/rwxr-xr-x 4096 dir 2018-04-08 06:56:55 -0400 lib
40755/rwxr-xr-x 4096 dir 2018-04-07 20:34:13 -0400 lib64
40755/rwxr-xr-x 4096 dir 2018-04-07 20:34:13 -0400 media
40755/rwxr-xr-x 4096 dir 2018-04-07 20:34:13 -0400 mnt
40755/rwxr-xr-x 4096 dir 2018-04-07 20:34:13 -0400 opt
40555/r-xr-xr-x 0 dir 2018-08-30 00:02:58 -0400 proc
40700/rwx------ 4096 dir 2018-04-15 13:08:50 -0400 root
40755/rwxr-xr-x 4096 dir 2018-04-08 13:50:32 -0400 run
40755/rwxr-xr-x 4096 dir 2018-04-08 06:57:55 -0400 sbin
40755/rwxr-xr-x 4096 dir 2018-04-07 20:34:13 -0400 srv
40555/r-xr-xr-x 0 dir 2018-08-30 03:16:19 -0400 sys
41777/rwxrwxrwx 4096 dir 2018-08-30 06:35:59 -0400 tmp
40755/rwxr-xr-x 4096 dir 2018-04-08 13:31:45 -0400 usr
40755/rwxr-xr-x 4096 dir 2018-04-15 13:08:19 -0400 var
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 > clear
[-] Unknown command: clear.
meterpreter >

After some more enumeration we navigate to **/usr/zeus/airgeddon** to witness the following files. None of them are a flag by the way! 😋 guys the fun has just started 😋. Now for the uninitiated, **Airgeddon** is a multi purpose bash script used to audit wireless networks. Here are the files in the directory:-



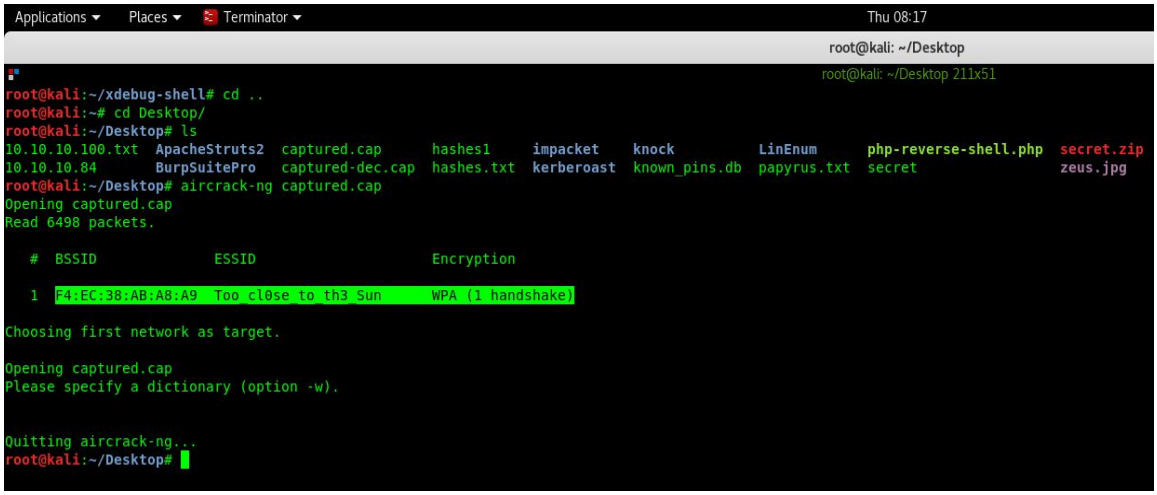
We navigate to **captured** directory to witness two files : **captured.cap , papyrus.txt .** Former is a captured file stored after successful **HANDSHAKE** between a client and a wireless network. Now after some more enumeration , it was evident that we will have to crack the encrypted .cap file to get any idea about the next phase. When we “cat” the text file it says:-

“**Captured while flying . I’ll banish him to Olympia - Zeus**”

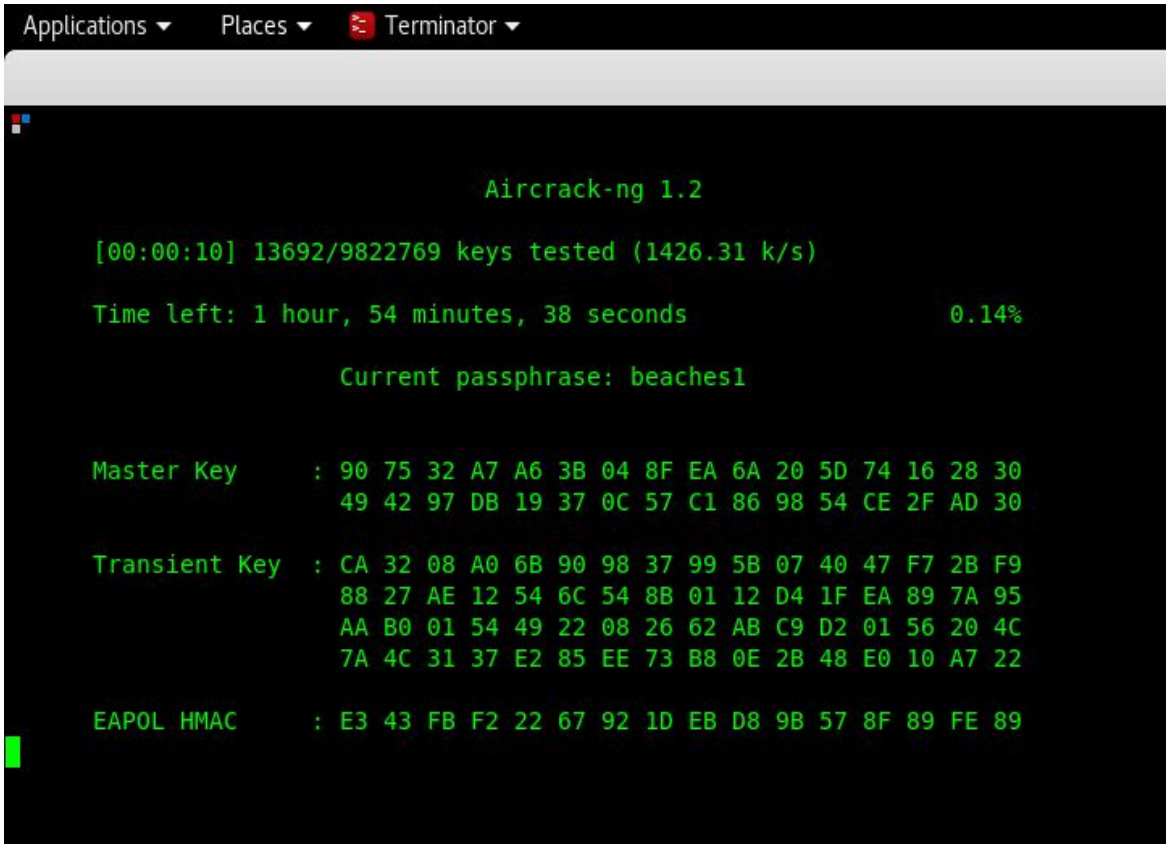
With this message it is pretty clear that we have to find a path to Olympia using the files.

---

2) **USER.TXT**: We download the files to local machine and get some more information about it. Since it is a handshake file we can run it with aircrack-ng. Here it is:-



We see the encryption as WPA (which is fairly secure , so cracking it may take some time , be patient) . Now we provide the wordlist - **RockYou.txt .** And let the cracking begin! 😉



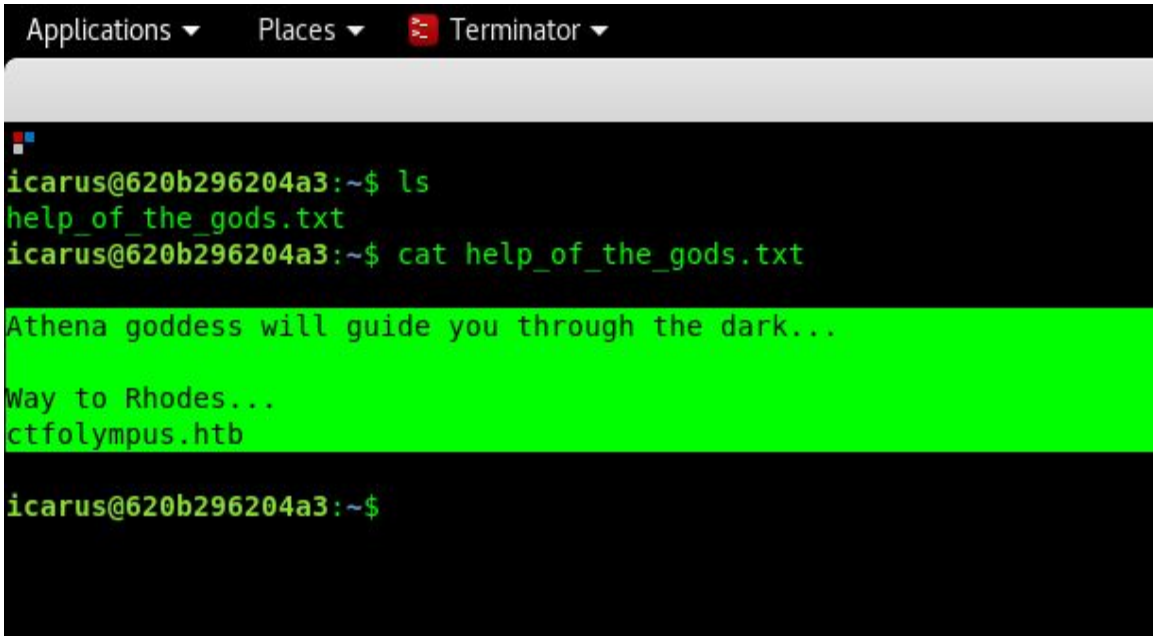
After some amount of time (depending upon the computational power) we will get the passkey as **“flightoficarus”**.

Now we have a typical CTF like situation where we have a lot of information but connecting the right dots is very necessary. The ESSID from above screenshot reads **“Too_cl0se_to_th3_Sun” ,** which seems like a password (Trust your intuition , that’s what CTF is all about).

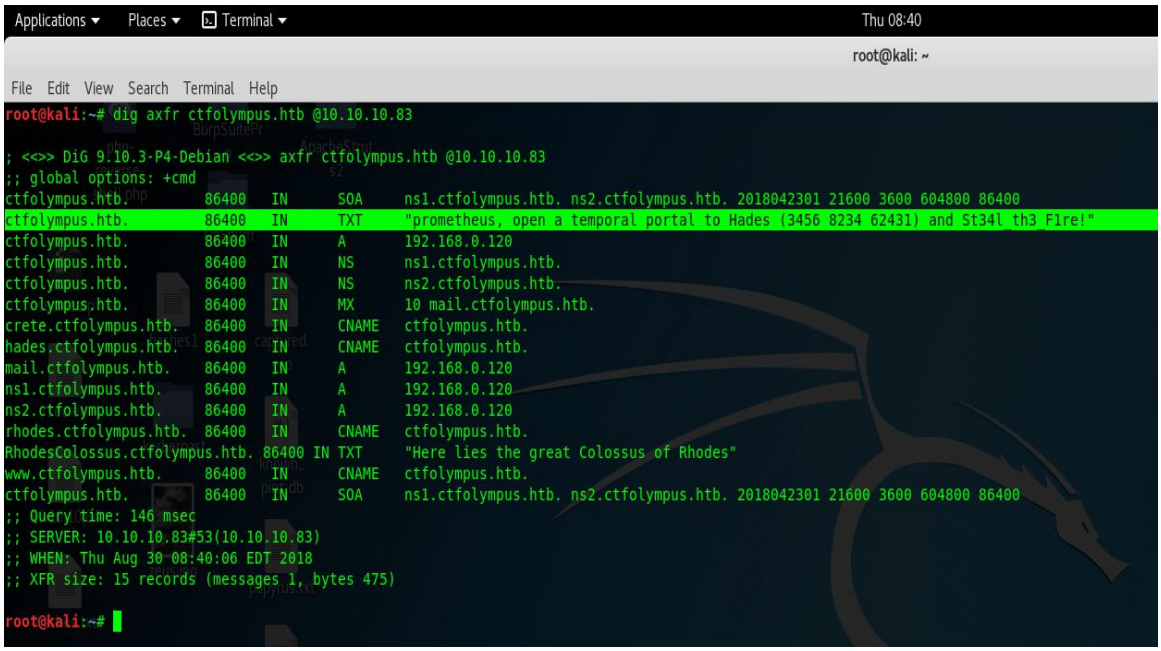
The story of “**Icarus flying too close to the sun”** can help in figuring out the username and password.

Reference:- http://www.mythweb.com/encyc/entries/icarus.html

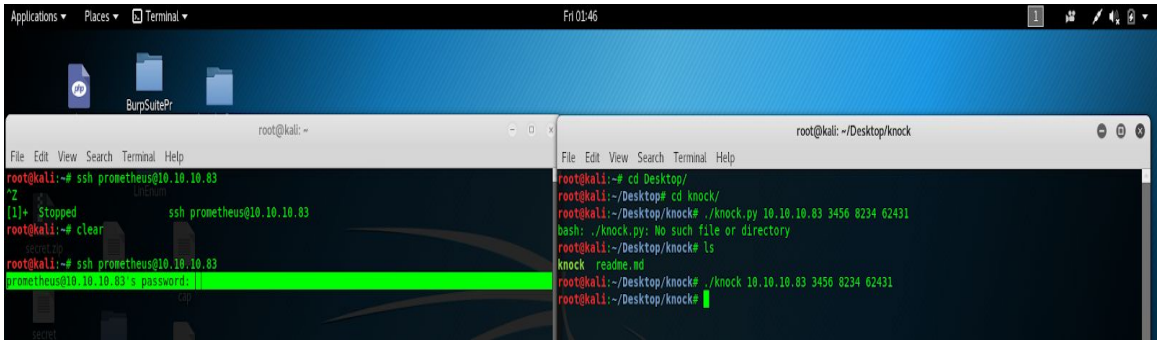
Going for SSH (on port 2222) with gathered credentials.We see that we are still in a container **(icarus@620….)**. Reading the file gives us some information. Here it is:-



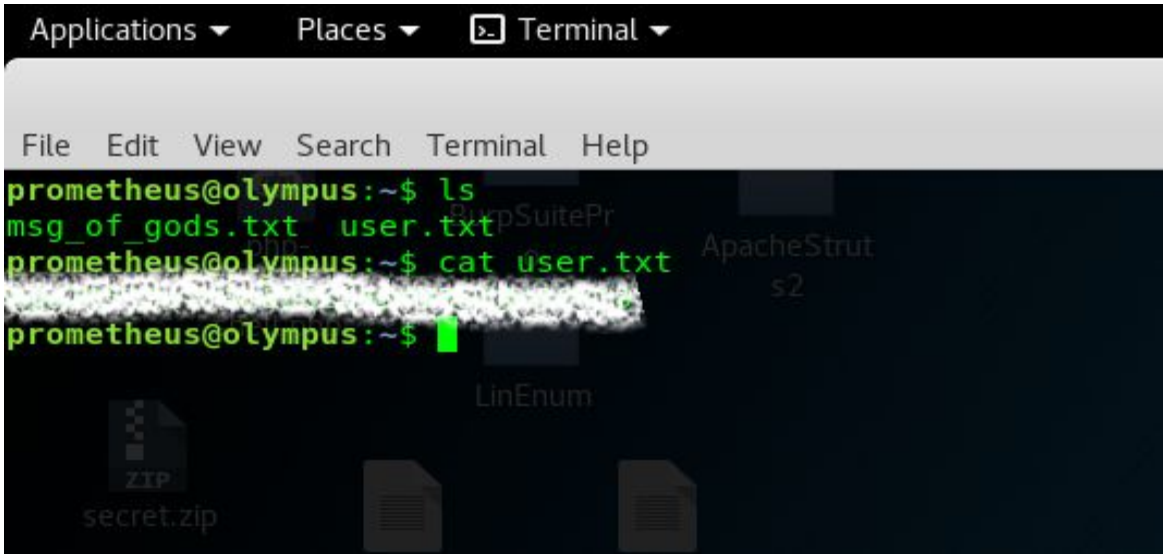
We see that we have got the domain name. Now we can actually connect the dots as to why port 53 was open. It is used for “DNS zone transfers” which happens via UDP but here we have port 53 open to TCP as well. Using the technique to figure out whether any text is present in DNS zone transfers through TCP. Here it is:-



In one of the TXT we can clearly see that a **“temporal portal to hades”** can be opened and we can figure out username and password very easily. Now the numbers (3456 8234 62431) refers to the port numbers. We have to use the **“Port Knocking”** technique to open the SSH port 22. Run the combination of **“Knock”** with SSH to open ssh port.



We can notice that port 22 for SSH is open after knocking the given series of ports. Now we can login in the machine using the credentials we got from **“dig axfr”** command. Now we are in olympus and we can easily read the **“user.txt”** flag.

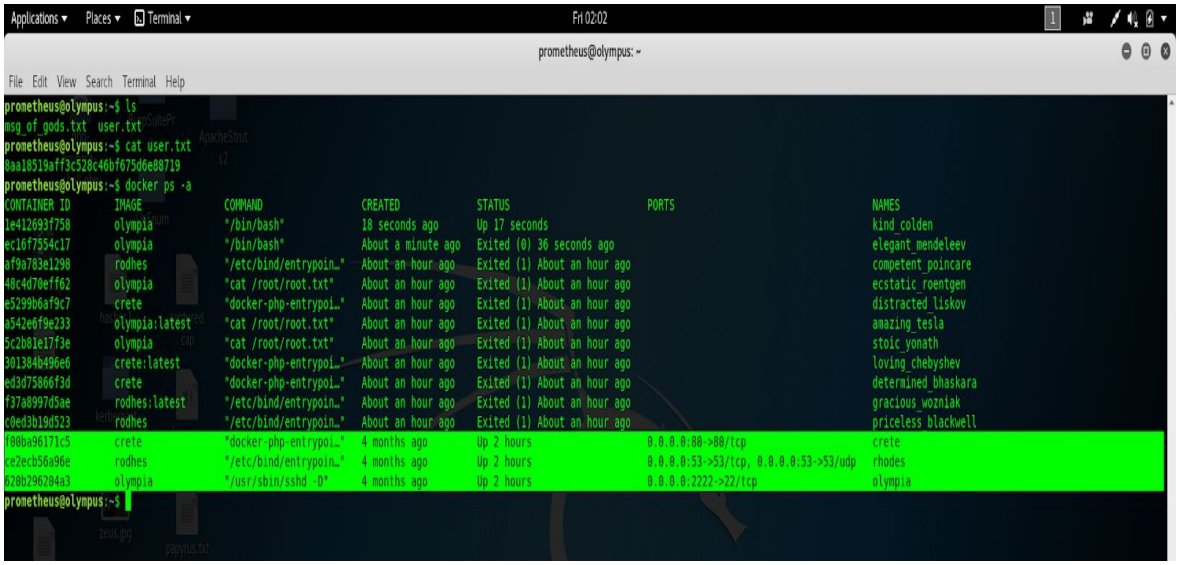


---

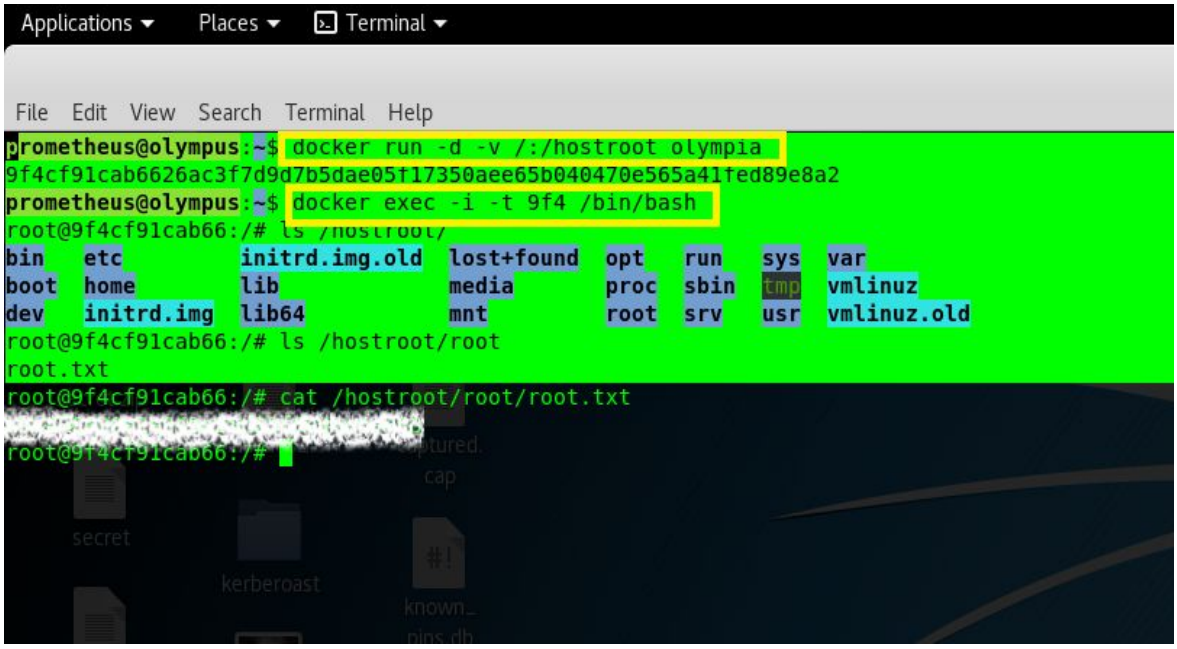
3) **ROOT.TXT**: Now we move towards getting administrative privileges in the machine to read the all important Root flag. We already got to know that docker is installed on the machine. Let's run the following command.

"prometheus@olympus:~$ docker ps -a"

This command will list all the containers present in the machine. We can see there are basically three images : **crete** , **rhodes** , **olympia** .



We know that Docker always runs with administrator (root) privileges , so giving docker executable permissions to a user is actually giving root permissions to the user. Let's exploit that:-



We created a simple docker container using olympia as a template and mount the root of the host in to the container. And we can easily read the **root flag** using this technique.

Thank you to all my fellow culprits, more write-ups are coming your way. If you like this let me know through **HTB respect** 😋.
