# Vulnhub HackDay: Albania

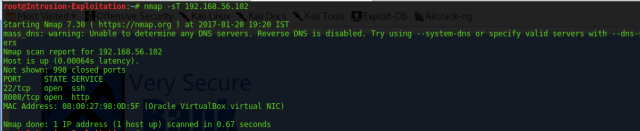
Walkthrough for Hackday Albania ,  the vulnerable image can be downloaded from vulnhub : – <https://www.vulnhub.com/entry/hackday-albania,167/>

The image is hosted on the Virtual Box with VirtualBox Host-Only Adapter.

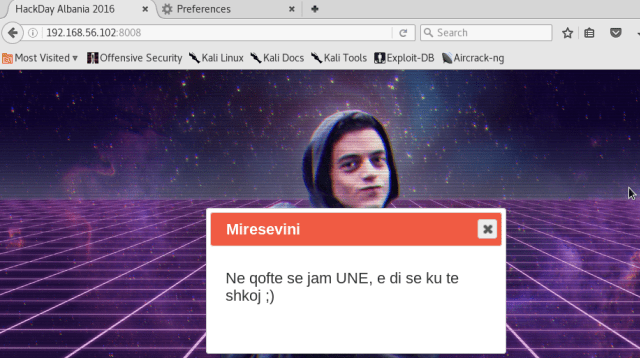
# Walkthrough:-

We will begin with the reconnaissance phase. Fire up nmap to scan for the Open ports on the Host. Nmap query used :

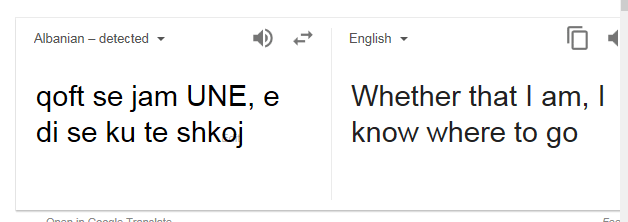
nmap –sT 192.168.56.102



On accessing port 8008 we find out that it’s a web application , it looks like this.

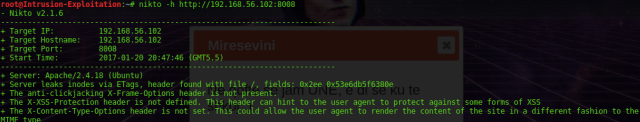


The translation for the same is as follows :

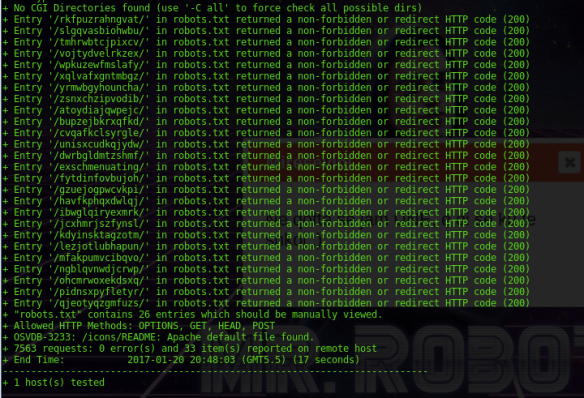


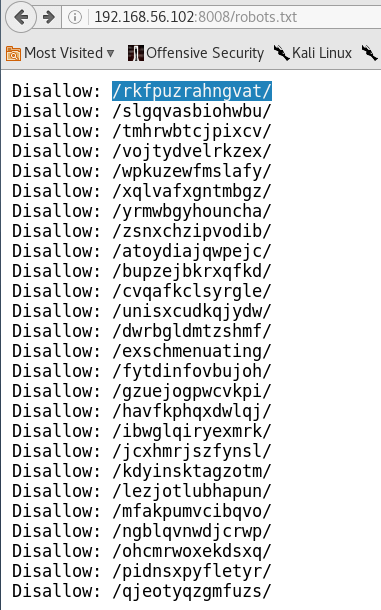
After reaching this page , it’s time to fire up nikto to see if we can get anything useful .

nikto  -h http://192.168.056.102:8008/

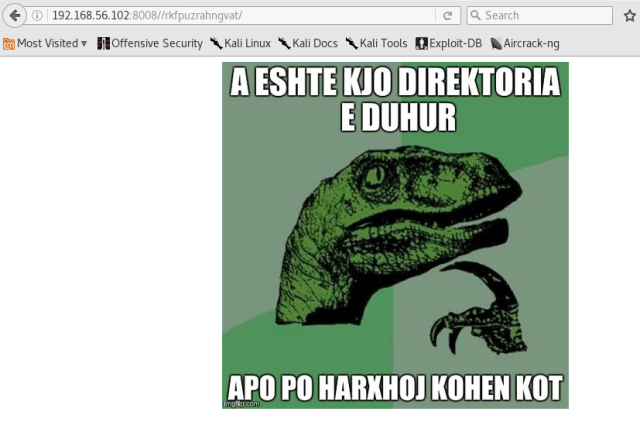


The interesting entry found is robots.txt that contained multiple folders with redirects .

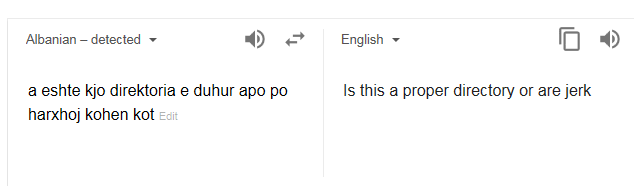




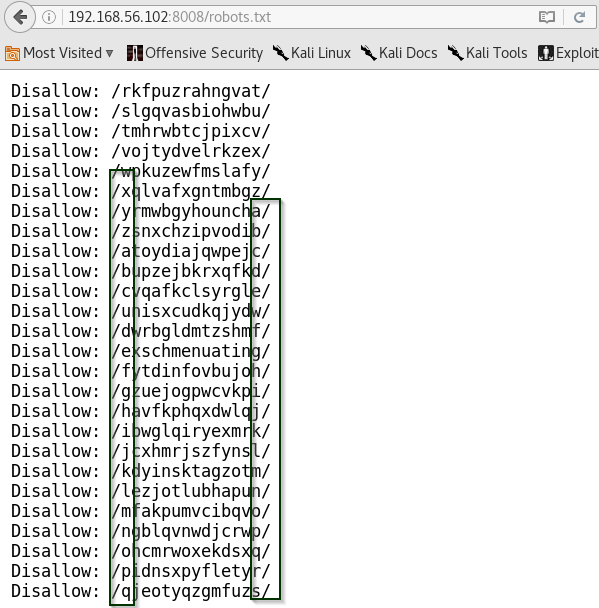
On accessing the first folder we get the following



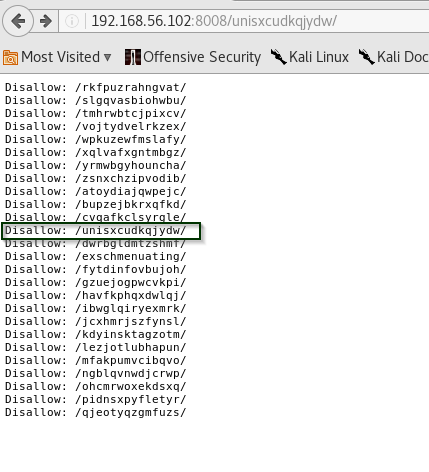
Which can be translated as follows



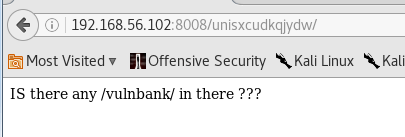
If you look closely there is a pattern that if followed every folder name is 14 alphabets , if we take the 1st alphabet of each folder it is an alphabetical sequence , same is followed for all the columns . however there is one folder that breaks the alphabetical sequence of all the columns , on accessing the same following was observed.



So the particular folder looks interesting



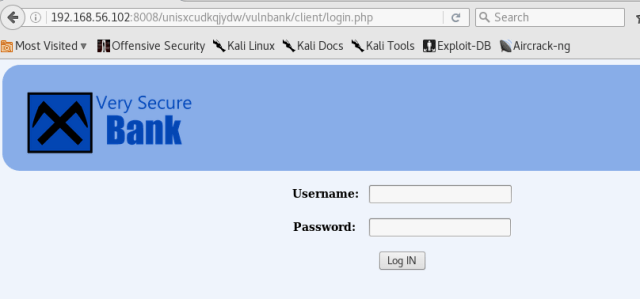
On accessing the folder  the we get a different output not redirecting to the meme that we encountered before .



Pretty Obvious we got to jump in the folder .



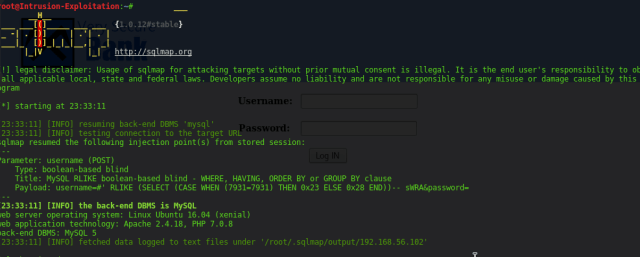
The vulnbank folder has directory listing and the available folder is client .



# Testing:

On clicking on the client folder we get this login , since we have our login page the best bet would be to guess its vulnerable to sql injection . However manual injection attempts using the magic string 1’or’1’=’1 failed nothing really came of it. Time to fire Sqlmap , the post request of the login was captured and fired on sqlmap with high risk and level feature of sqlmap.

sqlmap –u <http://192.168.56.102:8008/unisxcudkqjydw/vulnbank/client/login.php> –data=username=test&password=test –level 5

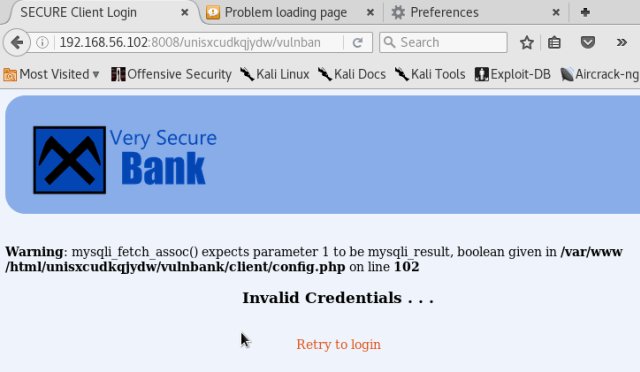


Bingo the username parameter is vulnerable to SQL injection Boolean based blind , and the successful parameter that verifies the same is given in the payload .

Now that we have the same we need to figure out to exploit the same , here we need to remember two things in MYSQL statements, # or —  can be used to terminate a MYSQL query.

# https://i0.wp.com/res.cloudinary.com/dqpbqkydz/image/upload/v1486527259/14_yko6mo.png?resize=640%2C399

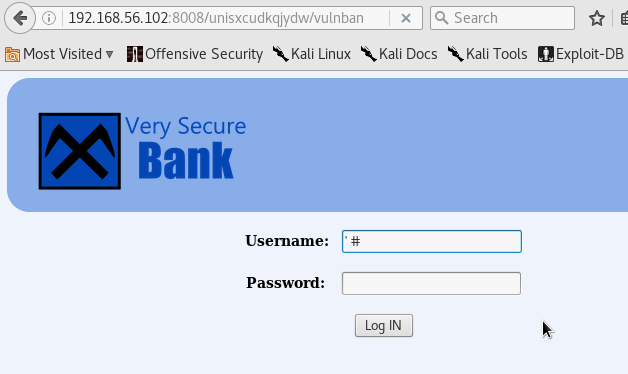
Using the magic string which we do for sql injection authentication bypass , lets see what happens.



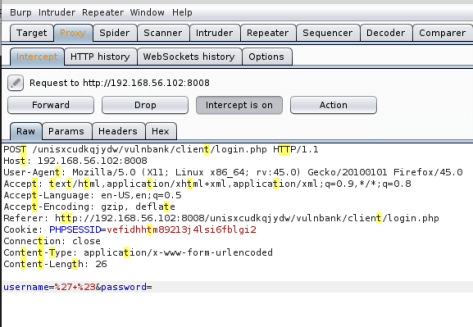
The result is Boolean , all we need to do is figure out a way to get the condition to true.

The usual query for authentication can be assumed to be “ select \* from users where username =’$username’ AND password=’$password’ “

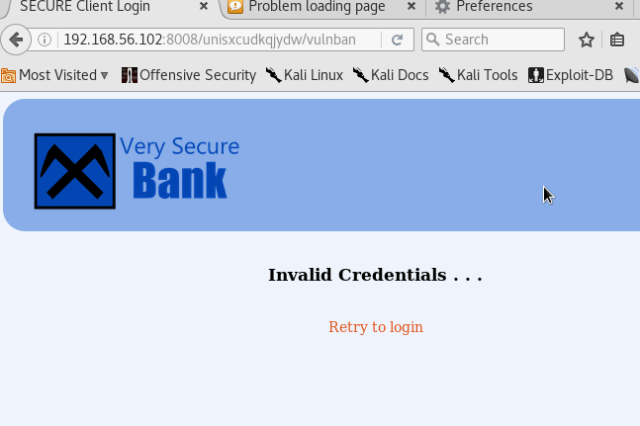
So we need to find a way to terminate the AND condition in the SQL query . we could do so with the help or “ ‘ #” lets enter the same and see how the browser responds.



On submitting this lets check the request in burp .



Forwarding the same we get Invalid Login .

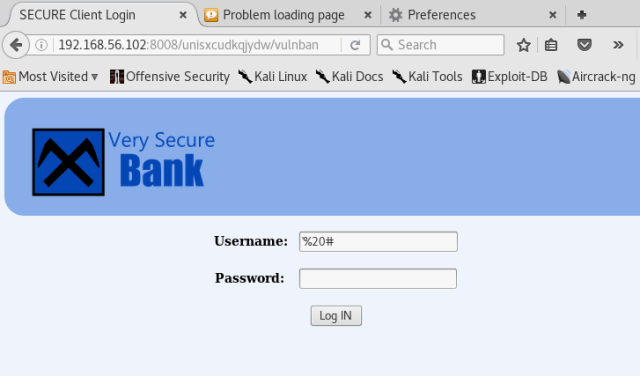


Damn , looks like it did not work . let us try to analyse the situation.

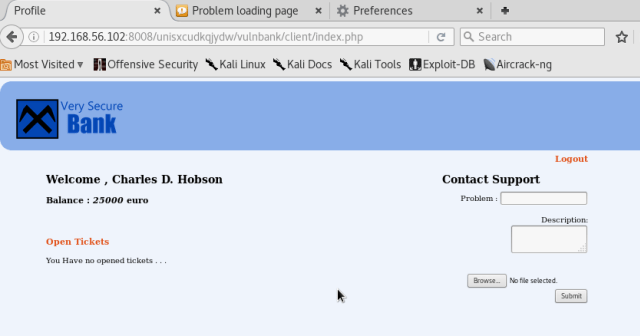
When we entered  “ ‘ # “ it got converted to  %27+%23 assume the same parameter being passed to the sql query of authentication.

“ select \* from users where username =’’+# AND password=’$password’ “

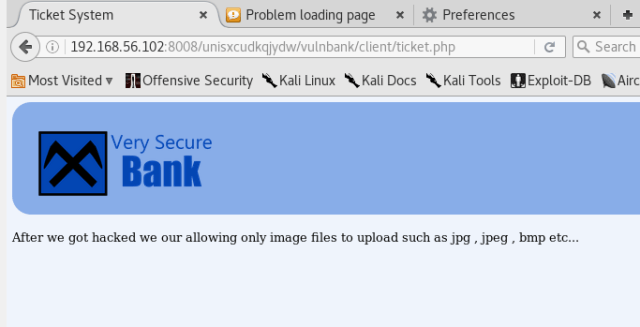
Offcouse this looks malformed we did not anticipate a “+” sign that defeats the query let us go ahead and ensure that the value we want to pass is a space (%20) and not the + .



JACKPOT we are in .

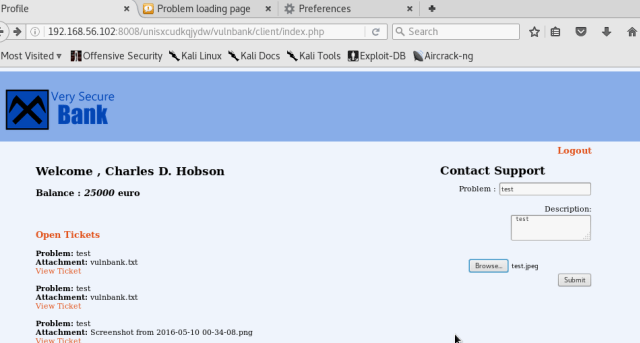


Say hello to Mr.Charles D. Hobson . we have successfully managed to bypass the query and gain access to the application as an authenticated user.  Now there is one feature that looks visible so on creating a sample txt file with practically random content tried to fill details and upload the text file .

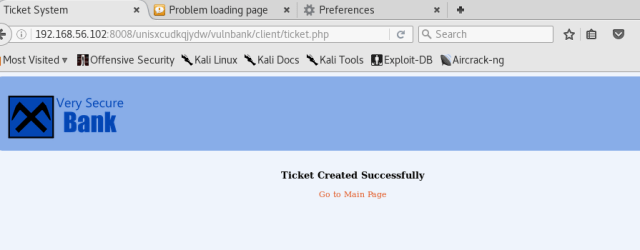


Wait what have we got here  ?

Hmm a ticket system tab , and a message that reads , that they only allowing image file formats to be uploaded .  With the simplest mind-set there are two clues here , there might be a vulnerable ticket system whose exploit is available or there might be a file upload bypass. Let’s go with the flow and upload an image file to check what happens on success condition.



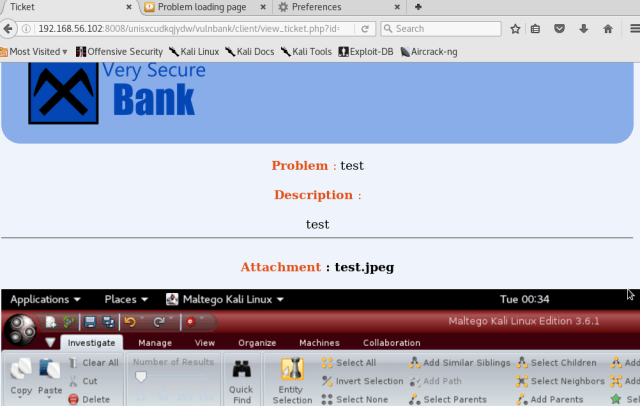
Submitting the same gave us the following result



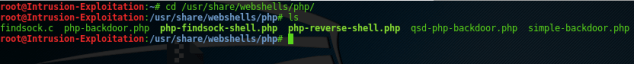
Okay so no other steps ahead of successful submission , however if you look at the main page there are tickets visible even on the error condition let us try to access our test ticket that we created .



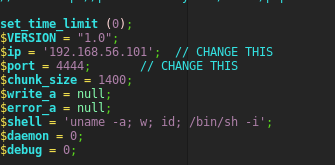
Our text file was not rendered , one can only come to the simplest conclusions i.e the application tried to execute the file but it did not open. How about the one uploaded successfully ?



Amazing the same file is visible and rendered. The application server is running on php so if we load a php reverse shell we will be pretty much in luck and get a reverse shell. Let’s head to uploading a simple php shell . There is a webshell directory available in kali Linux from where php shells are available.

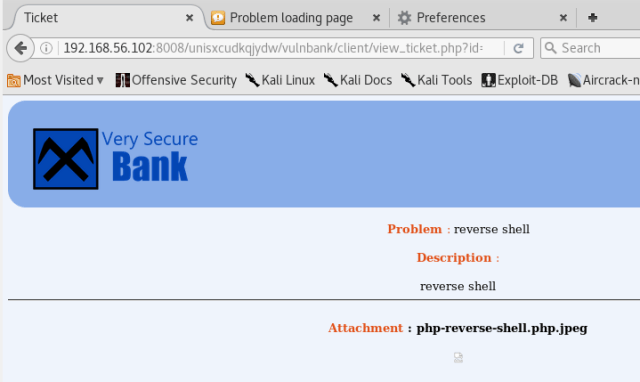


Edit the php reverse shell to the IP and port over your system.

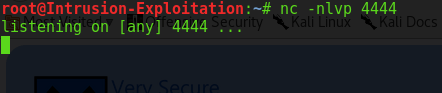


Like this for the instance currently in use.

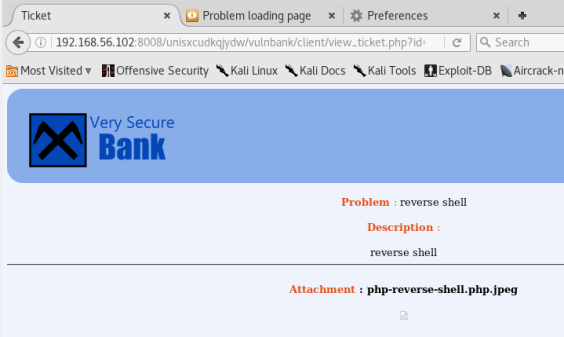
Let’s go forward and upload this .



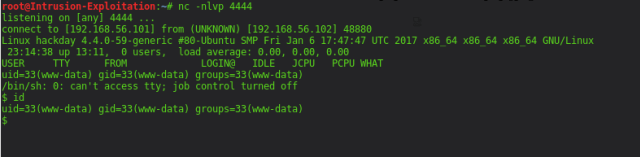
Well okay let’s see if we got a reverse shell .



Nope , doesn’t look like , in order to get the file to execute we will need to have the extension set to the allowed image files . So we go ahead and rename the file adding the .jpeg extension , upload the same , on successful submission we open the ticket .

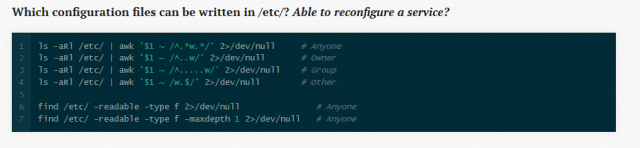


And on checking if we got a reverse shell

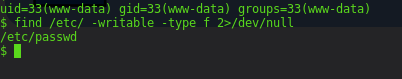


Well well well , what have we got here , a beautiful limited shell . The battle is half won. We are now a www user over the Server . I’d suggest if you are new to Privilege escalation go through Basic Linux Privilege escalation techniques by g0tm1lk , the same is available on <https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/> and <http://security-geek.in/2016/09/01/linux-privilege-escalation-cheat-sheet/>

So one of the basic checks I do is see if I can read the passwd file and the shadow file, just out of sheer luck else run linuxprivcheck.py a good script to give a list of possible entry points. Here is a snippet on in g0tm1lks linux privilege sheet.



Can you see the command in the 6th line that shows any file that has read access on any of the configuration files ,lets edit that a bit and change it to writable that’s where we hit the jackpot .



World Writeable on passwd file , how vulnerable can that be lol , now all one needs to do is replace the password of the user available inside .Copy the password in your own shadow file for your user or create one using openssl  or Use the following command to generate a salt password.

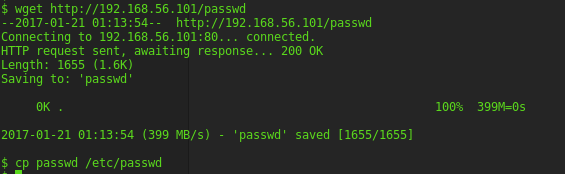
openssl passwd -1 -salt snypter hacker

https://i1.wp.com/res.cloudinary.com/dqpbqkydz/image/upload/v1486528736/36_tbry3b.png?resize=638%2C43

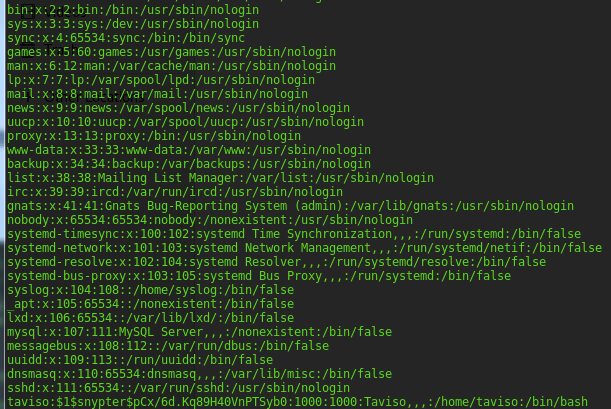
And replace the x in the passwd right after the username with the obtained hash .

NOTE: the x is usually there since the passwords are mentioned in shadow file , however if one overwrites the same in the passwd file that is the password that is chosen.

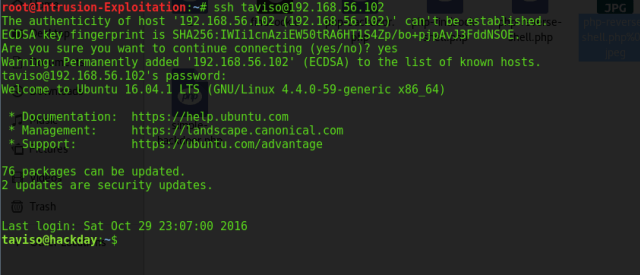
On the limited shell download the file and replace it with the original one.



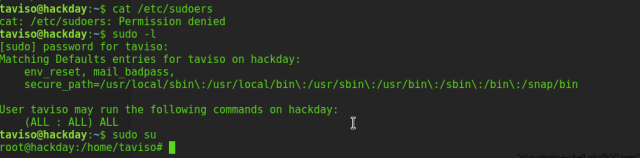
Tavis password has been replaced .



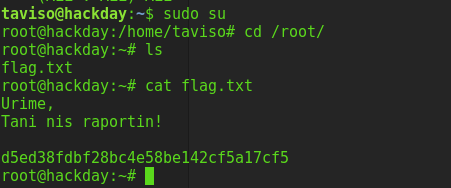
Login to taviso using ssh since ssh is open and use the new password that we just created as the password for taviso .



Now just check for sudo commands taviso is allowed to execute. And as we can see that taviso has all:all just run a ‘sudo su’ .



Where is my flag ?



And that’s how the server crumbles .

