# Initial Access

After Port Scanning the box, we find that port 80 is open with port 22.

Graphical user interface, application

Description automatically generatedWe Then proceed to view the login page of the application on port 80

## First Stage Payload

We Try to login using some credentials, but the login is incorrect!

What can we do?

1. Submit a single quote and look for some errors
2. Submit a SQL statement to look for systematic differences in the application response
3. Submit Boolean conditions and look for application responses
4. Submit time delay payloads and clock the application.

Test the

We try to leverage an SQLi Vulnerability, we can assume that the query is something like this:

SELECT \* FROM accounts WHERE username = '${(username)}' AND password = '${(password)}'

Now if we look at the query carefully, we can see that we as the user have control over 2 fields in this query, one being the ***${(username)}***and the second being the ***${(password)}***

## Breaking out of the payload

We can break out of the payload using the following input on the username field:

‘—

The initial query will end up looking like this:

SELECT \* FROM accounts WHERE username = ''--' AND password = ''--' AND id > 1

The double dashes show a comment in SQL, and this means the value after the dashes is used to comment out the rest of the query and not evaluate it.

# Second Stage

We managed to bypass the login page, now we are presented with a blog. Page.

Graphical user interface, text, application, email

Description automatically generatedWe Then Proceed to inspect the html.

Graphical user interface, text, application

Description automatically generatedWe find an interesting path; we investigate further and find the home.js file to contain an interesting API endpoint that fetches a list of all the users.

Graphical user interface, text, application

Description automatically generatedWe Visit that endpoint and fetch a list of all the existing user accounts.

We find that the user has a hash, we run it through a hash identifier

Graphical user interface, application

Description automatically generatedWe can then Crack the hash using hashcat or use an external service known as crackstation.net

Graphical user interface, text, application

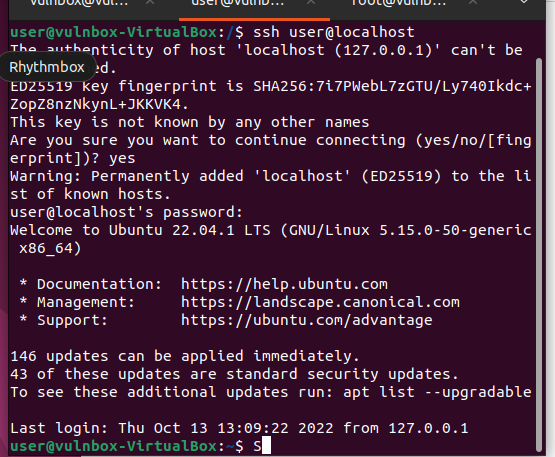
Description automatically generated

The hash was found inside the database, we can now use ssh to login with the username and password combination into the server!

# Third Stage

## Logging in Through SSH

We Proceed to use the current user and password combination (user:hacker) to login using ssh.



After logging in we can look at the current environment and which files we have an SUID bit set. A file with SUID always executes as the user who owns the file, regardless of the user passing the command.

Text

Description automatically generated

We find an interesting script named “cleanup” in the root directory. We can cd to the root directory and find out what it does.

Text

Description automatically generated

It appears that this script is trying to remove files from the /tmp folder using the /usr/bin/rm binary. How about we change the path of that binary to /bin/bash to get a root shell.

