TryHackMe ZTH: Web 2

Saikat Karmakar | AUG 3 : 2021

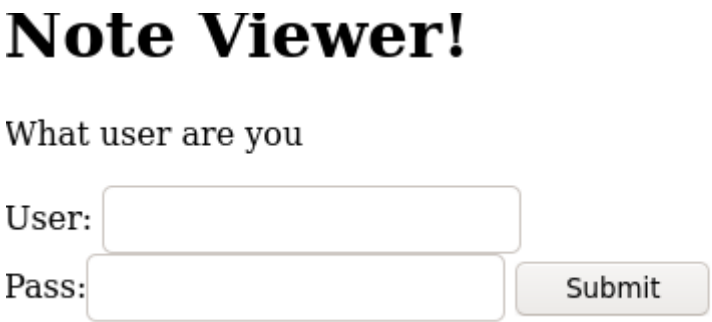
IDOR

IDOR, or Insecure Direct Object Reference, is the act of exploiting a misconfiguration in the way user input is handled, to access resources you wouldn't ordinarily be able to access.

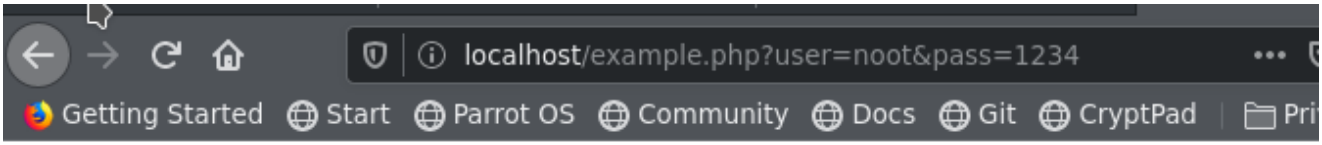
For example, let's say we're logging into our bank account, and after correctly authenticating ourselves, we get taken to a URL like this https://example.com/bank?account_number=1234. On that page we can see all our important bank details, and a user would do whatever they needed to do and move along their way thinking nothing is wrong.

There is however a potentially huge problem here, a hacker may be able to change the account_number parameter to something else like 1235, and if the site is incorrectly configured, then he would have access to someone else's bank information.

We start off with an example page.

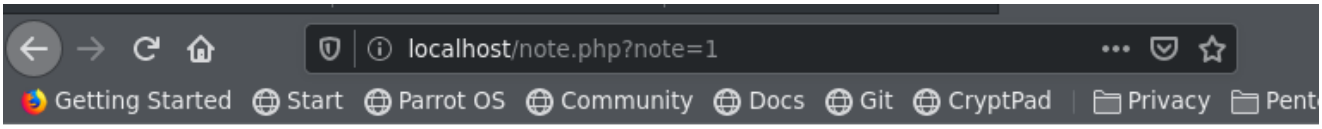


It makes enough sense, if you authenticate with the right user, you get to access that user's note.



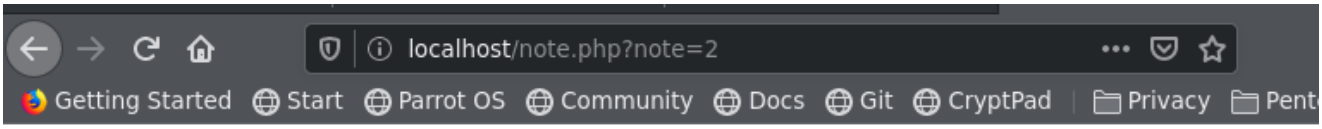
Incorrect Password

If you have the wrong password, you get an incorrect password message. For our purposes the user is noot, and the password is test1234. Authenticating correctly, as expected takes us to our note.



I am noot!

However, as you may have picked up on, there seems to be an interesting part of the URL. It seems that the note that we can view is controlled by a URL parameter, let's check if we can access other notes, by increasing the number to 2.



the password to the server is test12345678

Woohoo! We can access other's notes. While this may seem dramatic, exploiting this is the real world can have drastic consequences. Let's say you found an IDOR vulnerability in a note keeping site, which allowed you to access the notes of others, you could find plenty of personal details, like passwords, usernames, even credit card information.

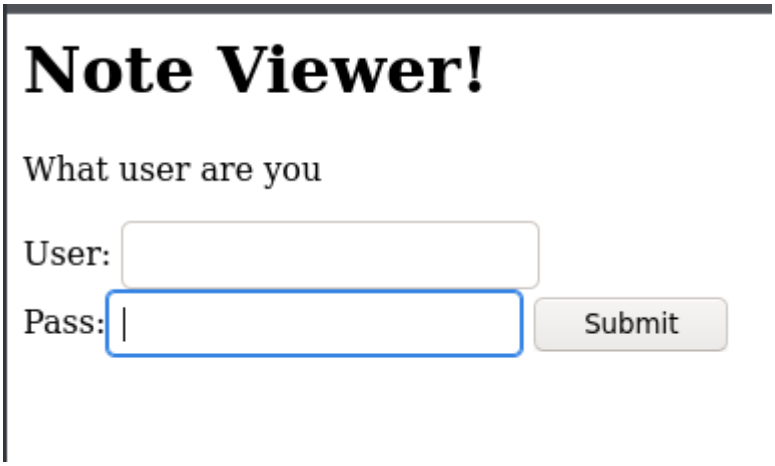
There is no way to automatically exploit this, as the pentester you need to examine the site, and find misconfigurations.

Forced browsing

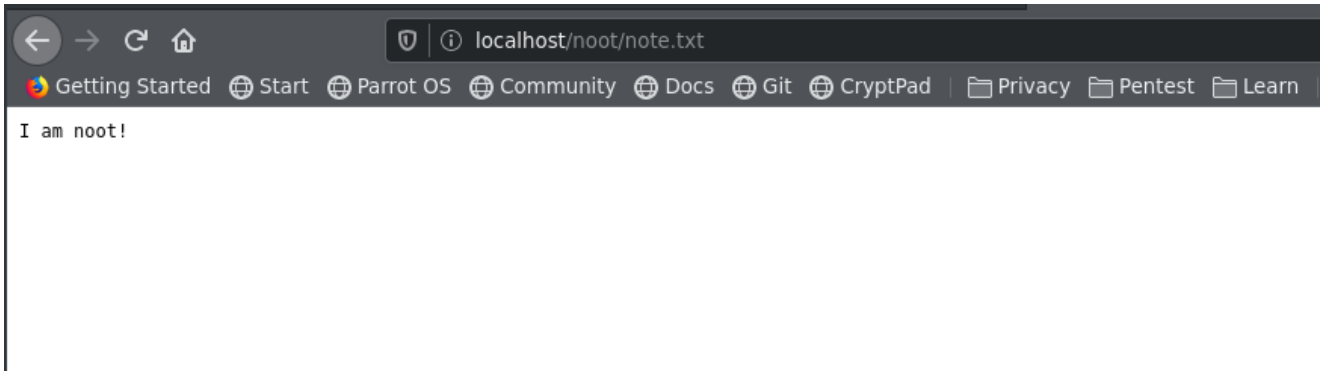
Forced browsing is the art of using logic to find resources on the website that you would not normally be able to access. For example let's say we have a note taking site, that is structured like this. http://example.com/user1/note.txt. It stands to reason that if we did http://example.com/user2/note.txt we may be able to access user2's note.

Taking this a step further, if we ran wfuzz on that url, we could enumerate users we don't know about, as well as get their notes. This is quite devastating, because we can then run further attacks on the users we find, for example bruteforcing each user we find, to see if they have weak passwords.

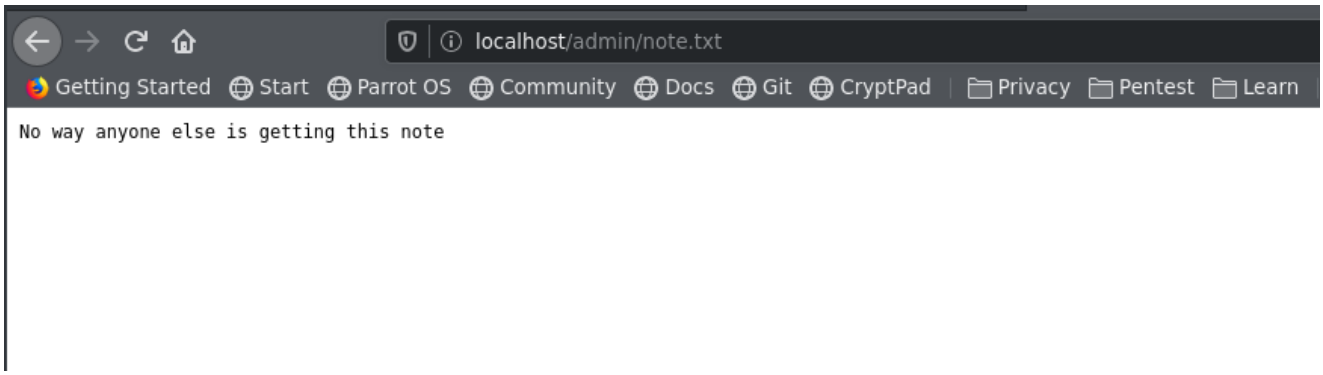
We start off with our basic note taking application.



For our purposes the user and password are noot, and test1234. Properly authenticating shows us our note.



Let's look over what we have. The user we authenticated as is noot, and the path for the note is noot/note.txt. Therefore, it seems notes are stored in `username/note.txt` meaning that if we find other usernames, we may be able to read other's notes! Often times on any application there is an admin user, so let's see if we can view his note.



Woohoo!

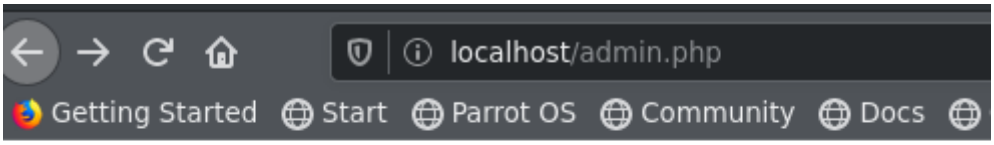
Forced browsing will often require some logic on the part of the hacker. To properly exploit this, you should keep notes on everything you find, building a picture of how the web application functions is essential.

API bypassing

This is a bit of a unique one, as it can basically be anything. APIs are by definition incredibly versatile, and finding out how to exploit them, will require a lot of research and effort by the hacker.

We start off with a basic login.

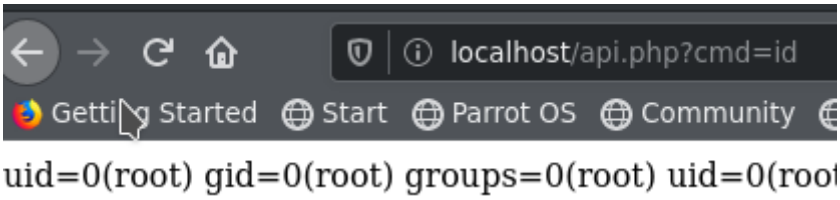
Logging in gives us an admin panel.



Command:

Command:

It seems we can run system commands here, so let's try running id.



Woohoo! It seems we can run commands. But let's think for a second, we didn't really need to login at all, if we found the api.php page through dirsearching, and a cmd parameter through fuzz, we would never have needed to use the login panel.

If you find an api.php file, it's important to check it out on its own. It's probable that it won't be as easy as wfuzz to get command execution, but we may still be able to find useful information.

There is no way to automatically exploit this, we may be able to use tools we already know to get additional clues, but it will always take manual research