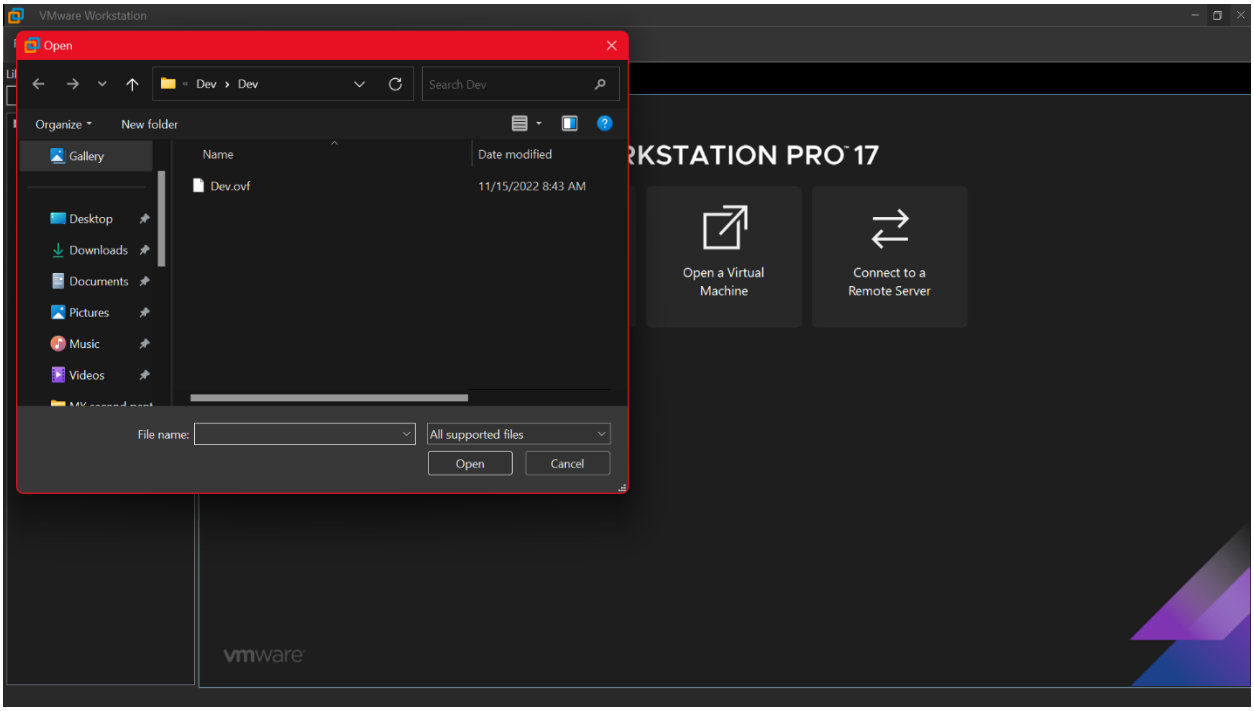
Capstone Report: Hands-On Exploitation of Dev VM – Lab Report

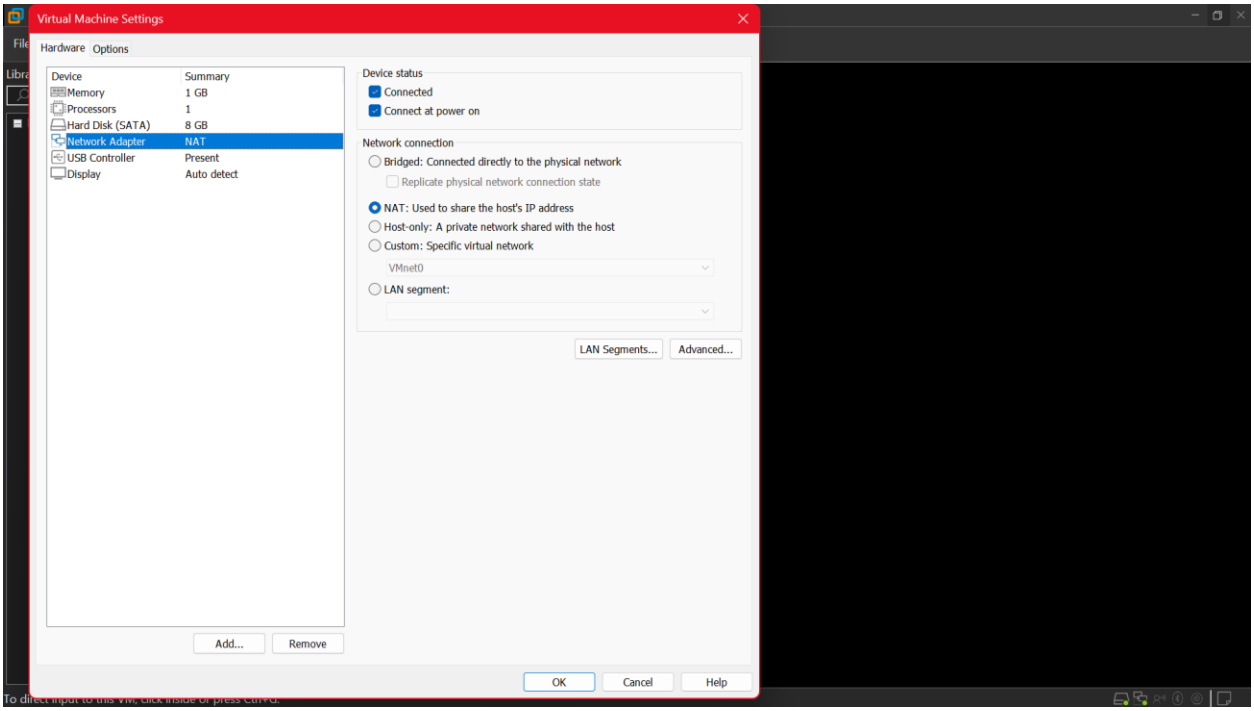
By: Lloyd Ensor Azumah

Overview

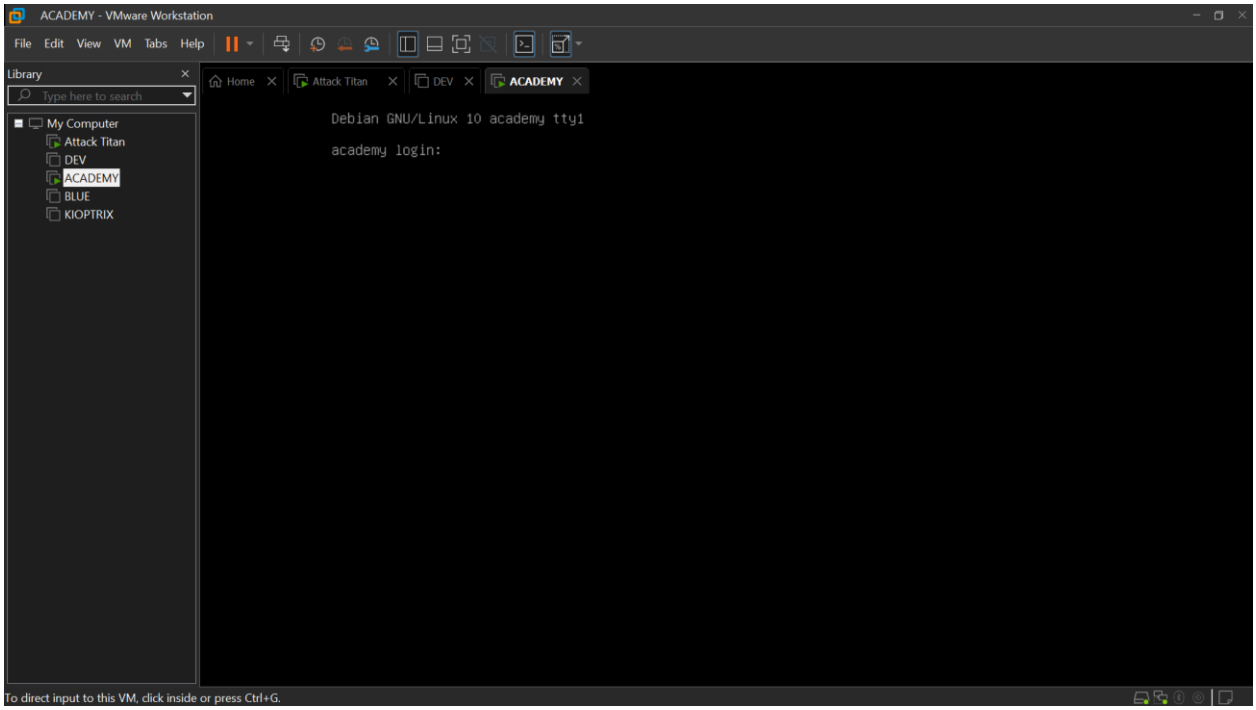
This report documents my hands-on experience compromising the Dev virtual machine in a simulated lab environment. By leveraging open NFS shares, password-protected ZIP cracking, directory fuzzing and SSH key authentication, I gained initial access to the system. I then escalated privileges using a known misconfiguration and a GTFOBins technique, ultimately achieving root access. This exercise reinforced key offensive security concepts such as file enumeration, local file inclusion (LFI) and privilege escalation.



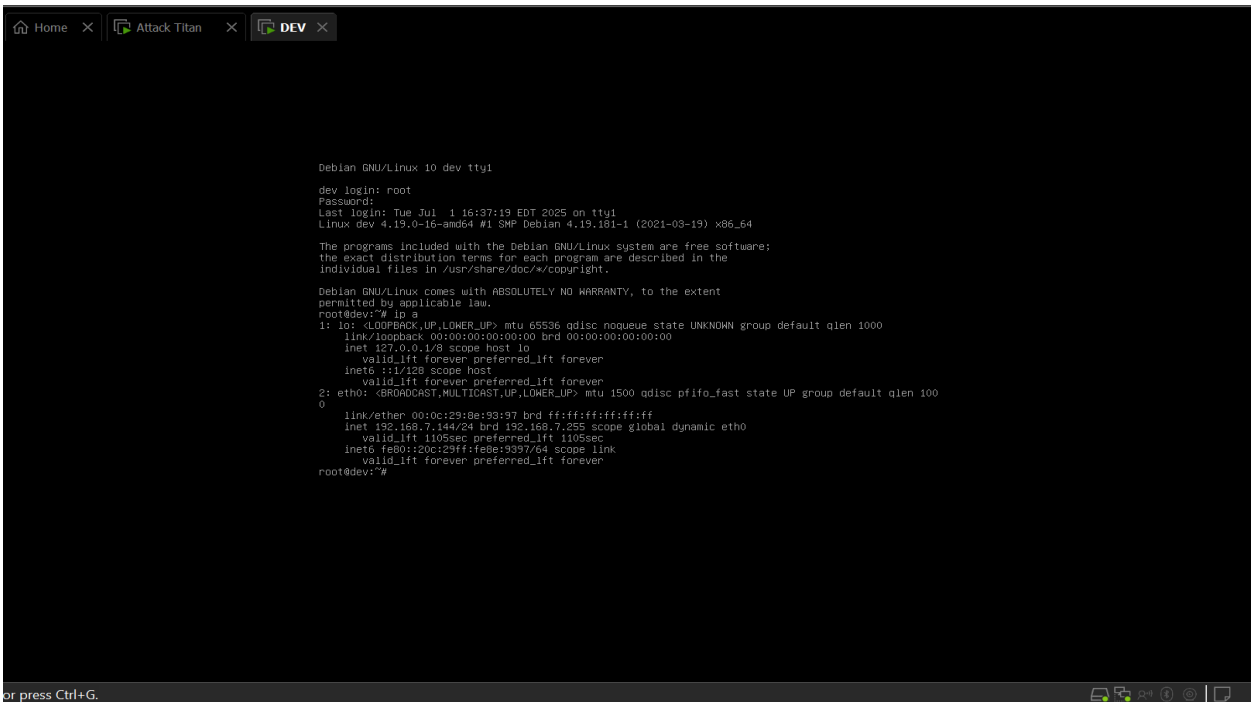
Once we have downloaded our virtual machine, we can click “Open a virtual machine” as this is an already built and configured machine and select “open” to import it.



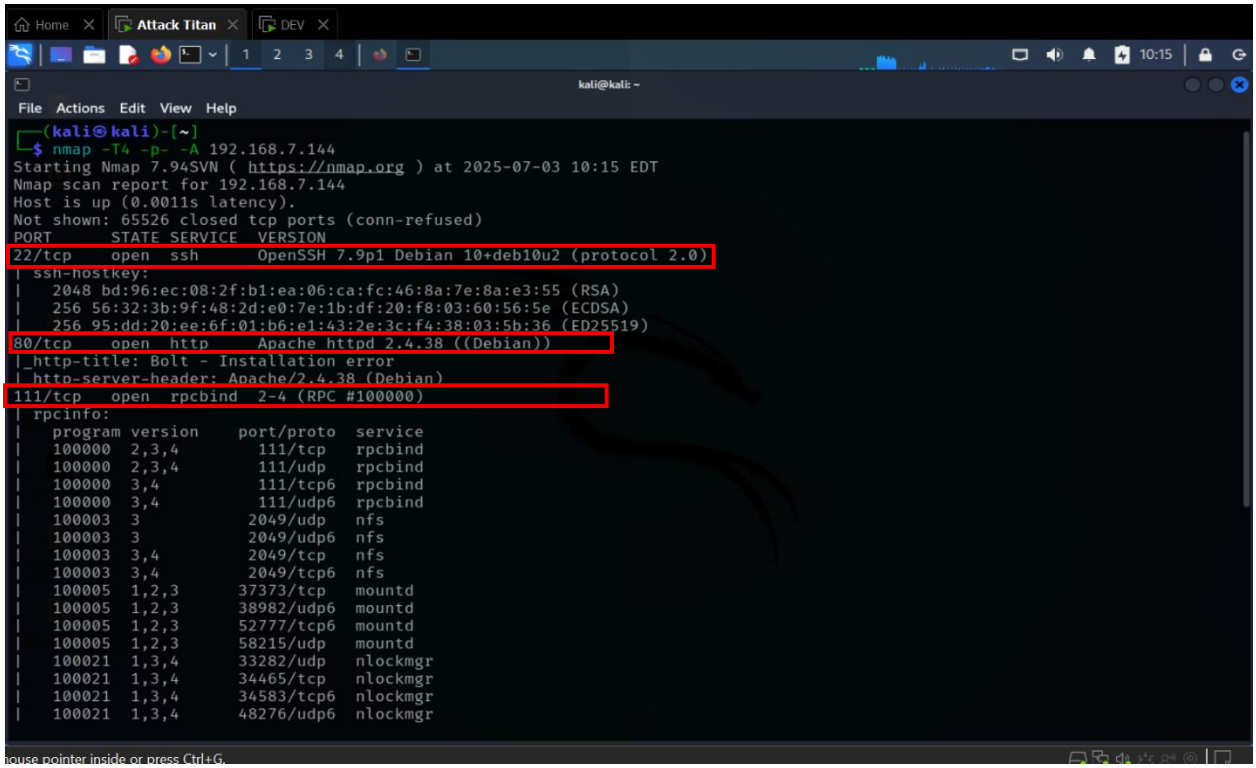
Once it has successfully imported, we will only need to make changes to the “Network Adapter” settings by converting it from “Bridged” to “NAT”. This is to ensure our attack machine is able to communicate with it (on the same virtual network).



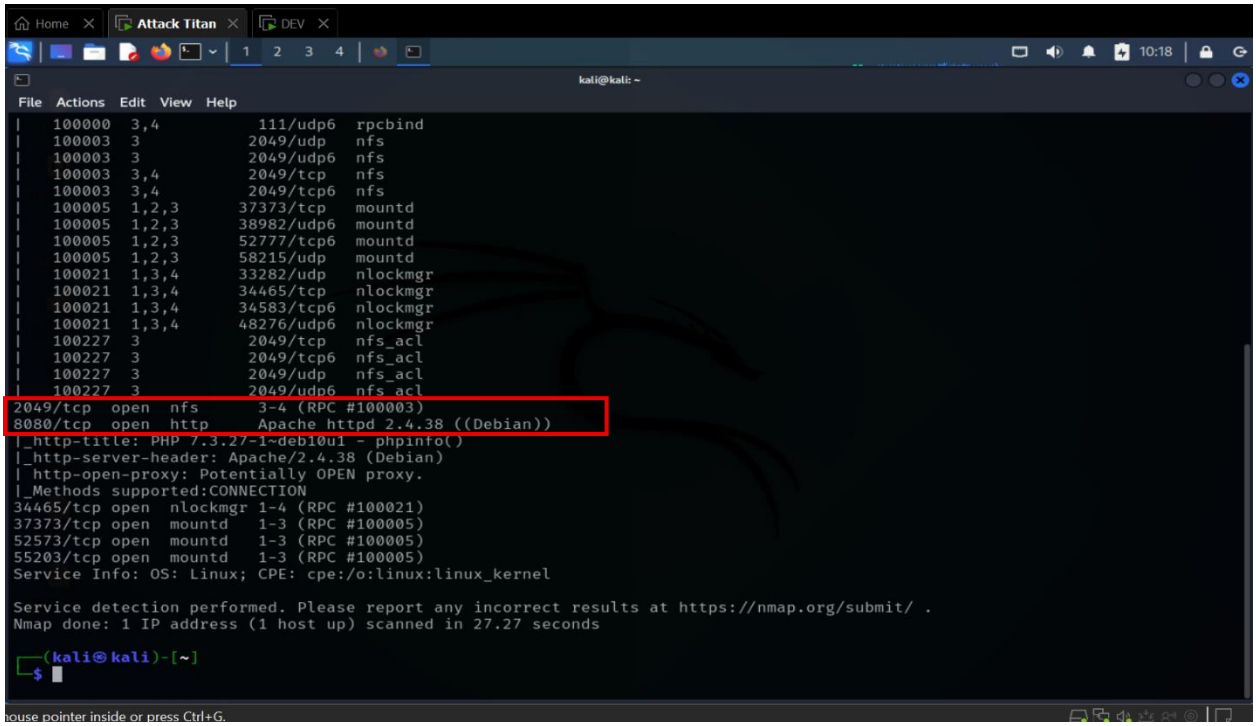
Now, we boot the machine which should to load to this interface



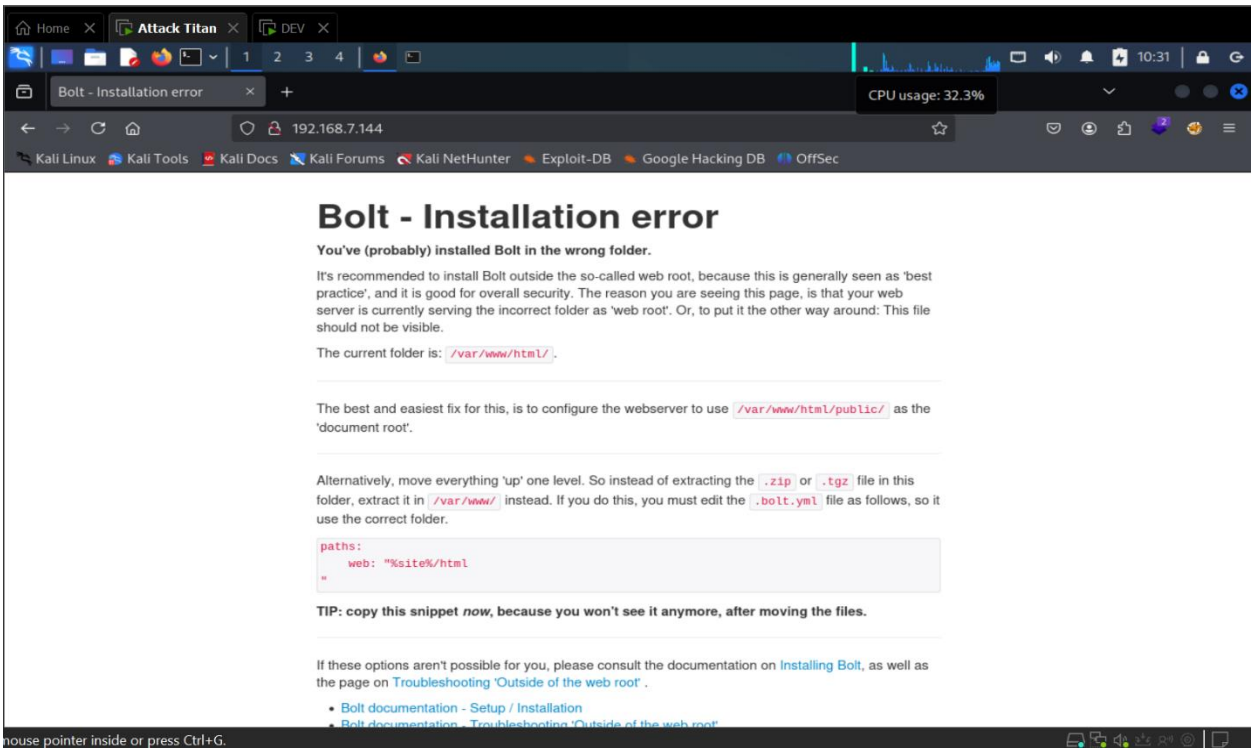
Here, we use the “ip a” command to verify the target’s IP address



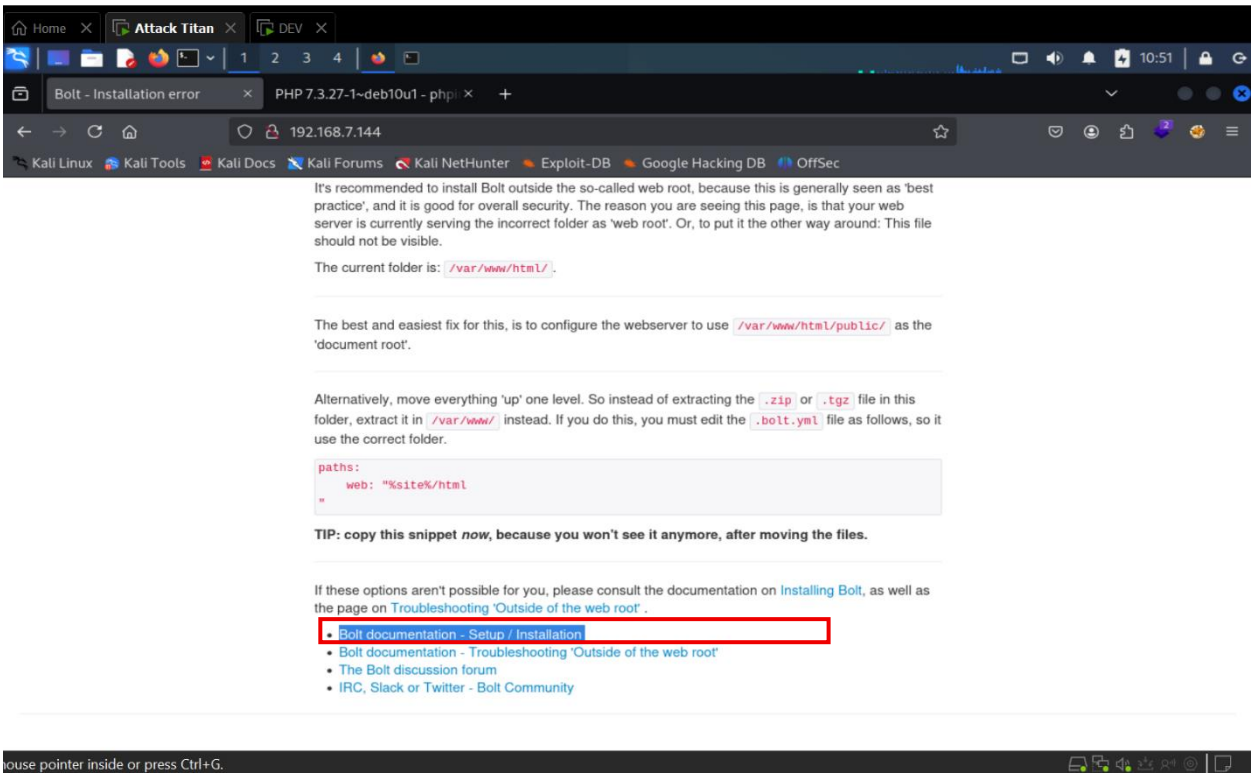
After running our **Nmap** scan, we get some interesting information. Ports such as 22(ssh), 80(http) and 111(rpc) are open.



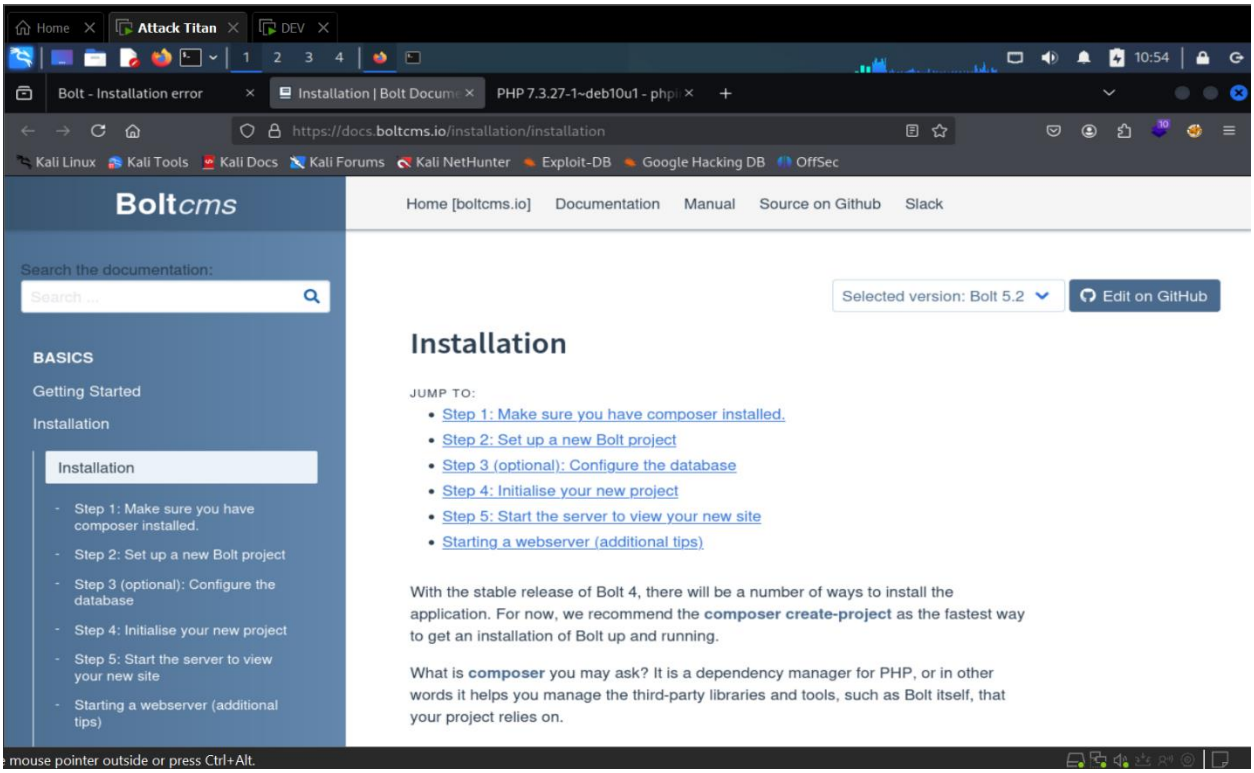
Also port 2049 (nfs) and 8080 (http) are open. So, we start from port 80 as it tends to provide us more valuable information by searching http://192.168.7.144 on our browser.



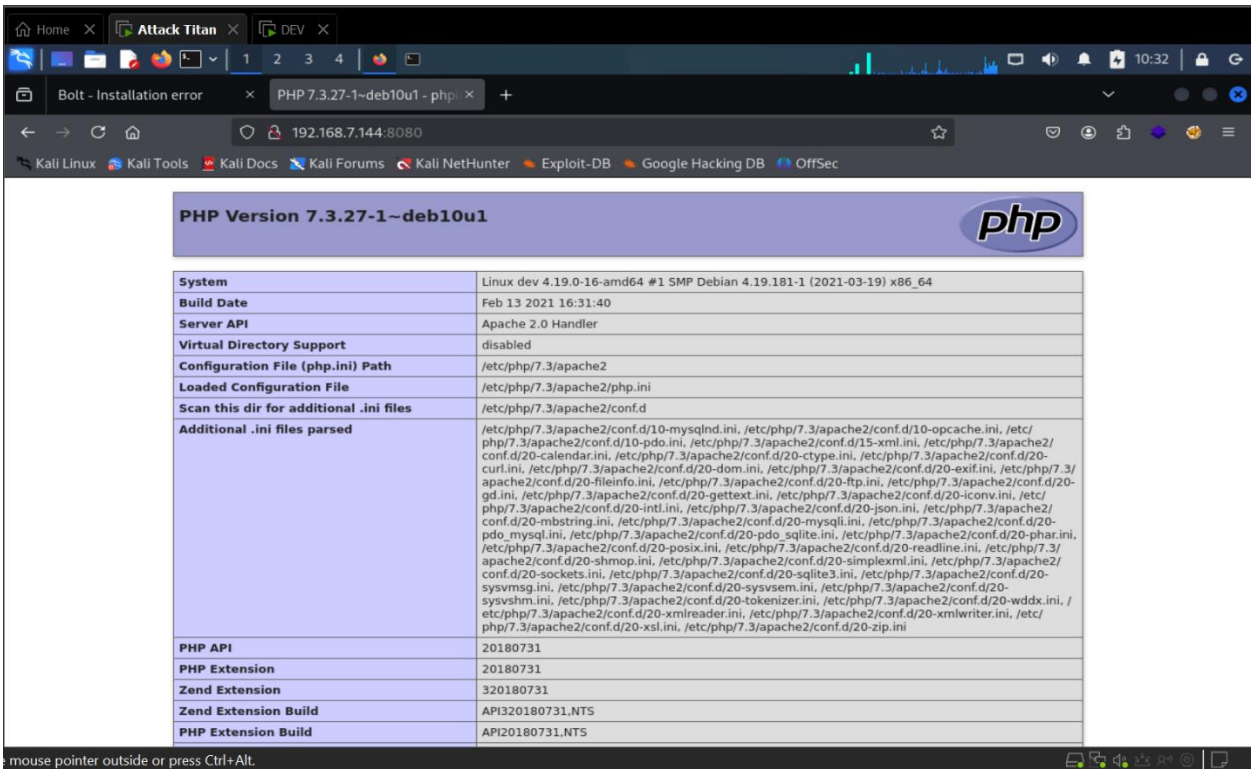
NB: The page we are looking at was opened through port 80 which is by default for all webpages.



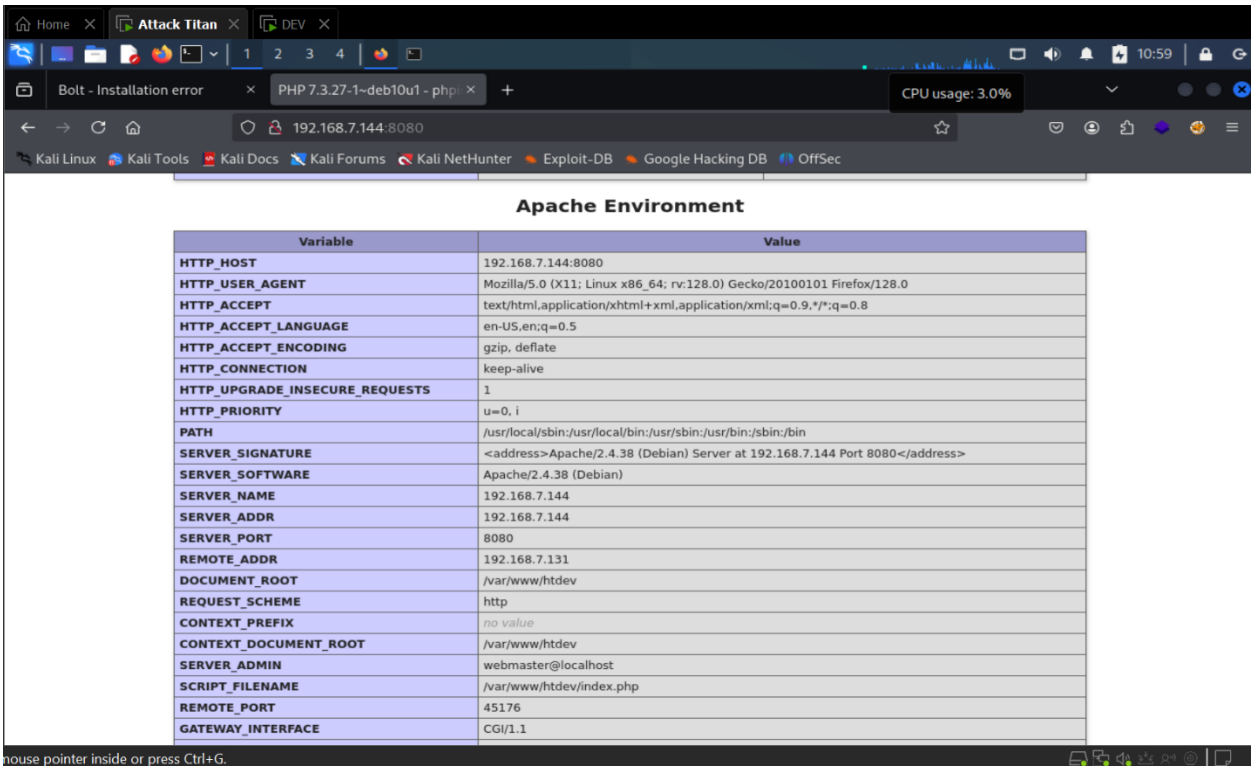
When we read what's on the page, it seems to be talking about how to setup a platform called "Bolt" and it goes ahead to provide a link to the documentation.



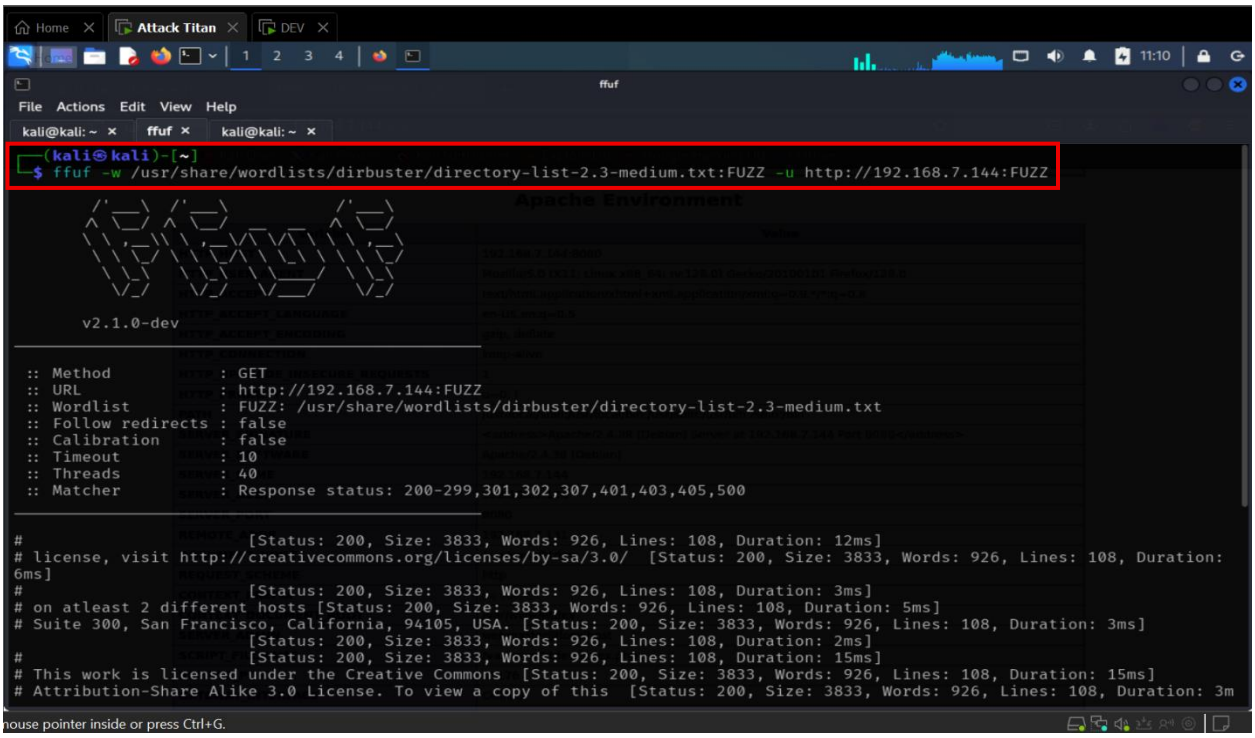
Bolt documentation setup



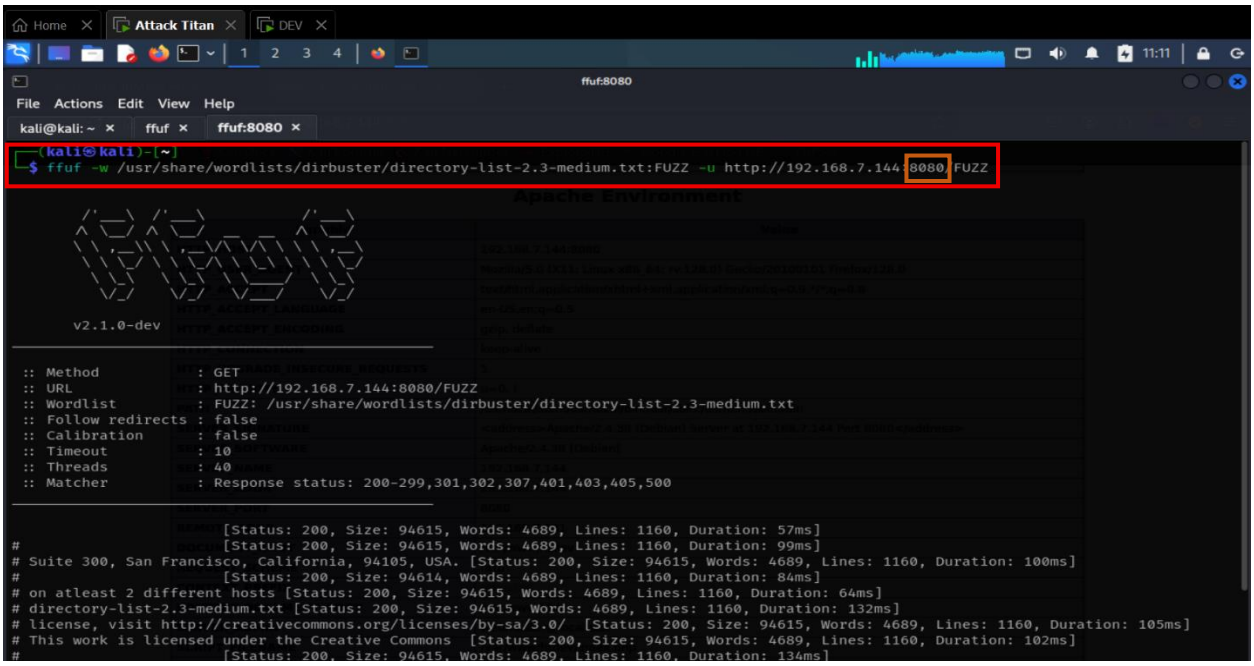
Now we open the same web page but through port 8080.



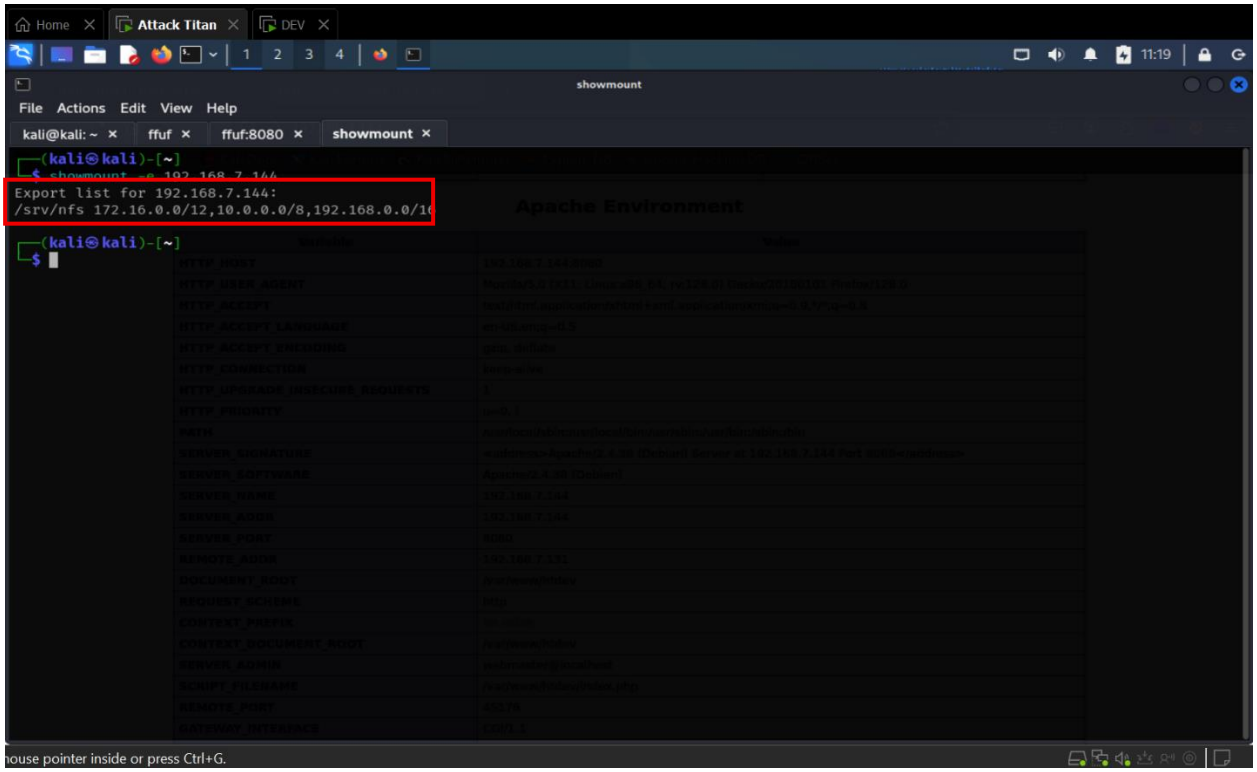
As a result, we get some interesting details as displayed, but this alone isnt enough.



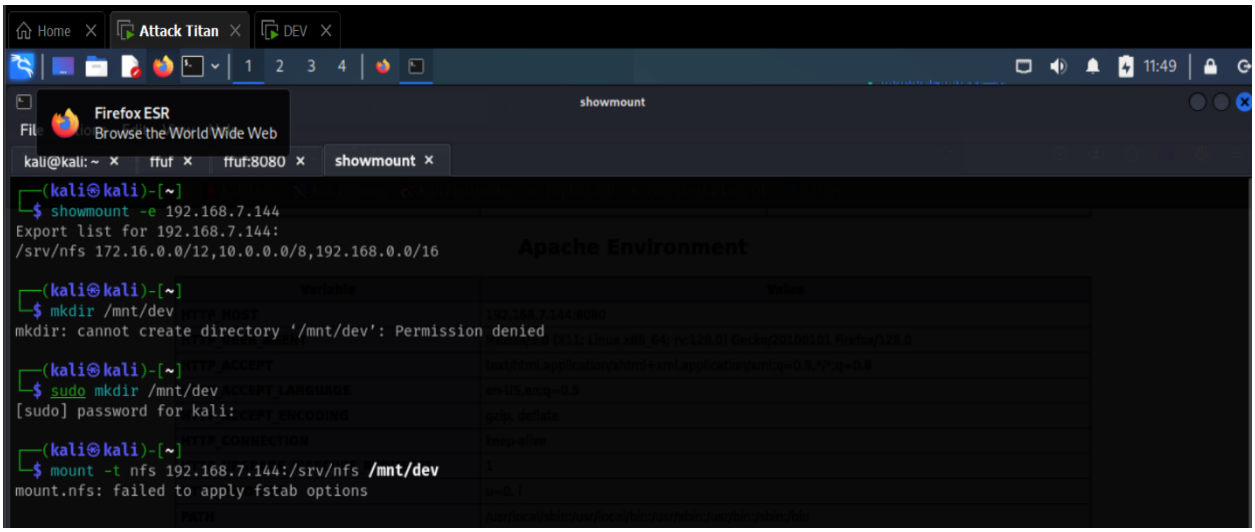
Now its time to expose those hidden directories within it. Using the **FFuF** (Fuzz Fast u Fool) tool, we start busting directories at port 80 first.



While the previous is running, we start another directory busting process at port 8080 simultaneously. These usually a considerable amount of time to complete so we can afford to focus on the other ports.



Port 2049 was open which means a network file share is available. In that case, we can mount whatever that maybe be sitting on this server and in our situation, that is "*/srv/nfs* …….".



We were trying to mount that into the */mnt/dev* directory we created but we encountered an error.

✅ 2. Check Mount Command Syntax

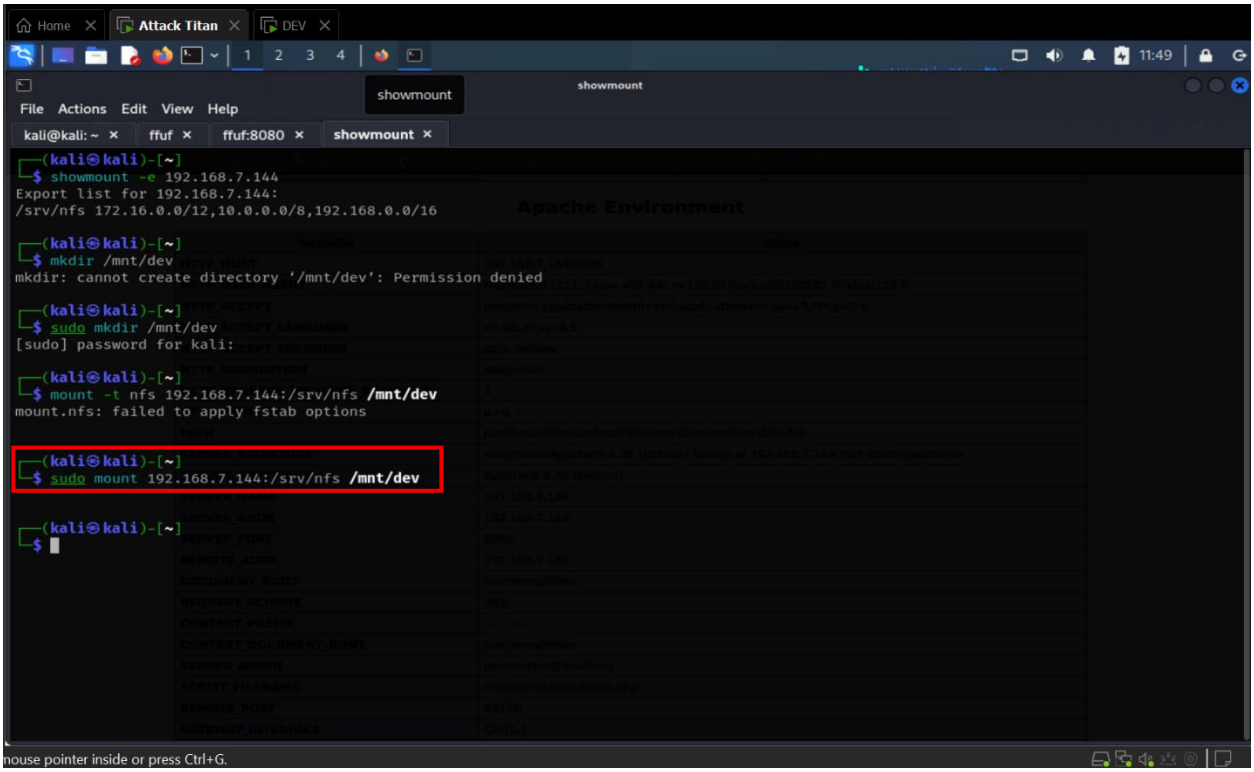
You don't need `-t nfs` if your system automatically detects NFS. Try:

bash Copy Edit

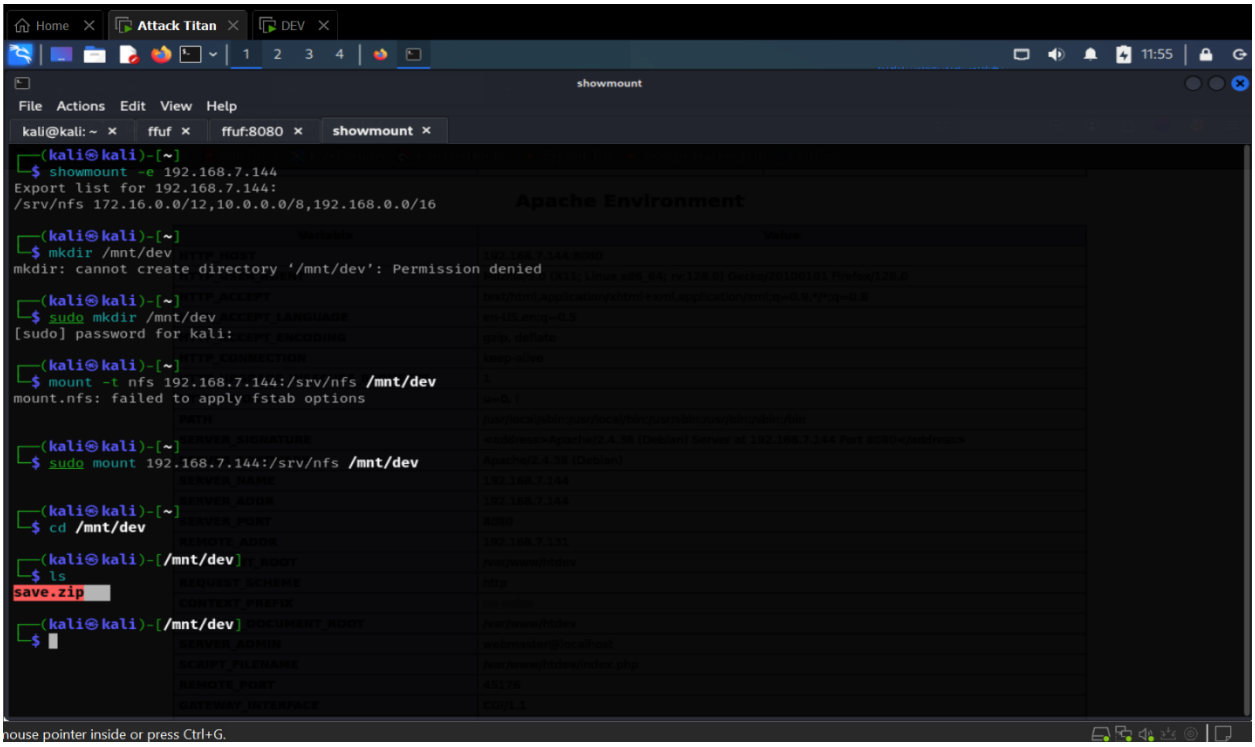
```
sudo mount 192.168.7.144:/srv/nfs /mnt/dev
```

If it works, you can add it to `/etc/fstab` for permanent mounting.

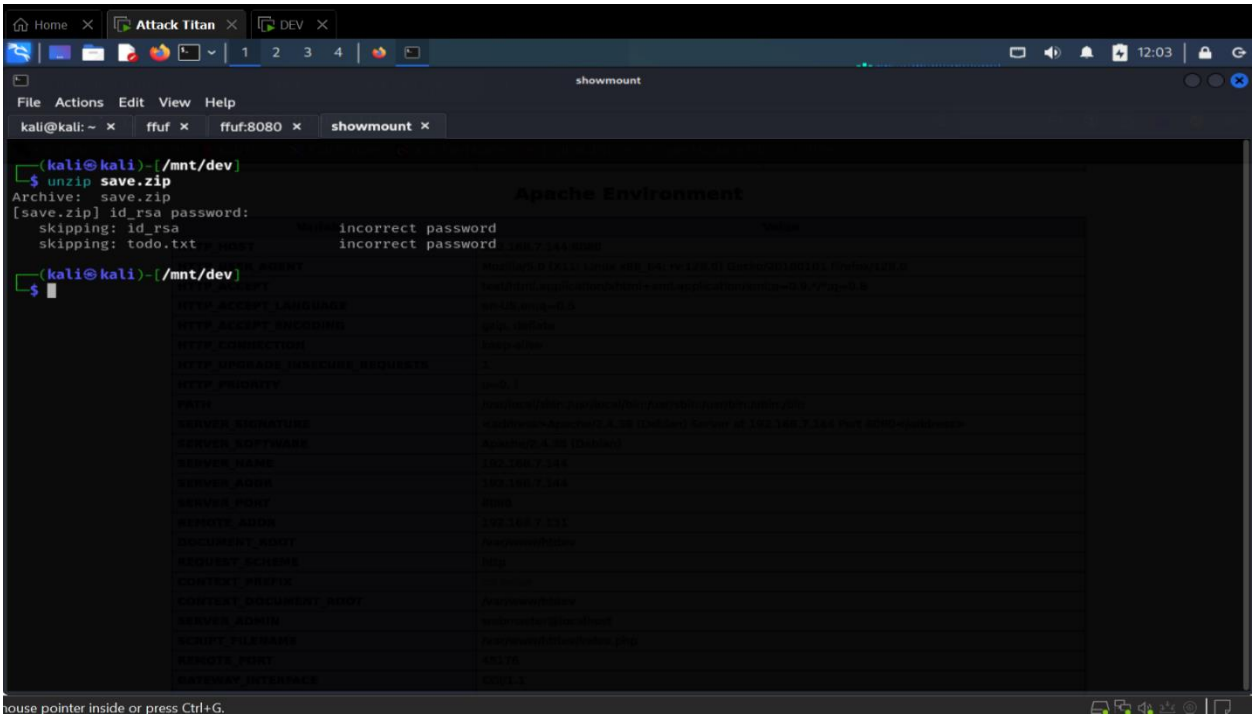
With some research, we learnt that our attack machine could already detect NFS automatically without the "-t nfs" included.



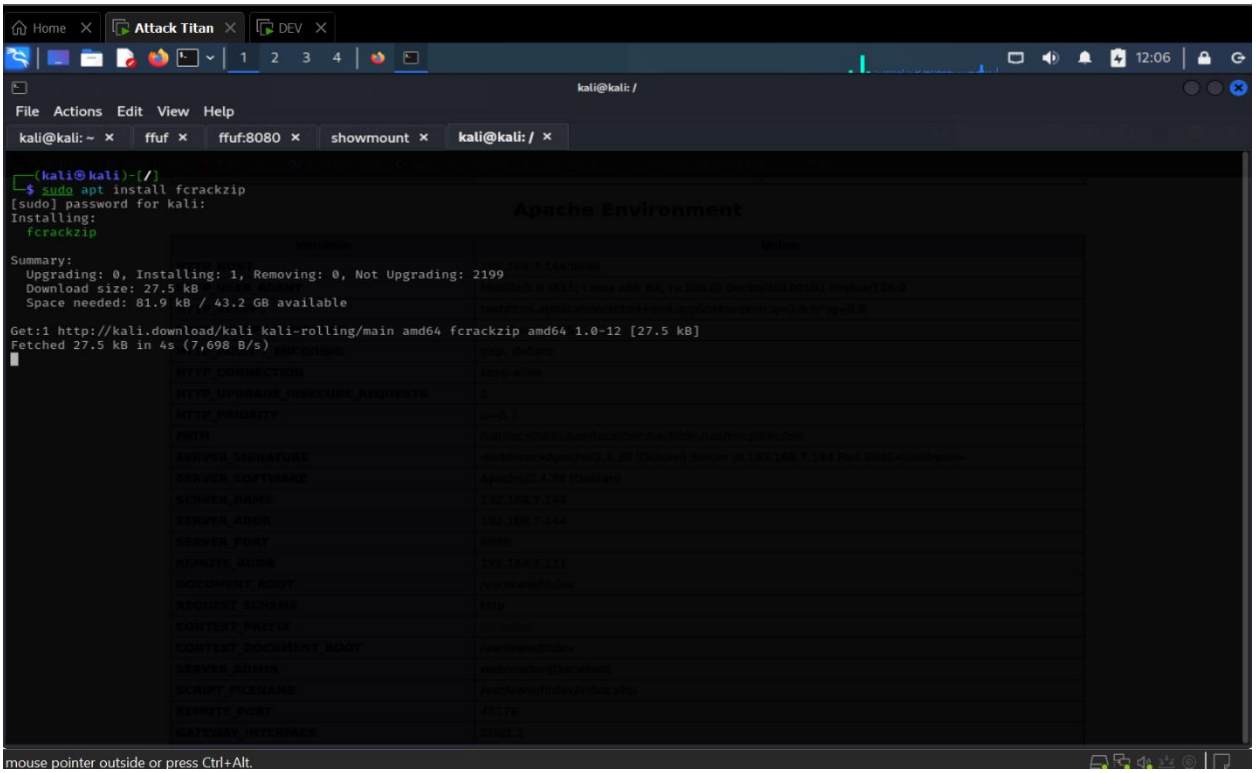
After we applied the corrections, we experienced no error feedback.



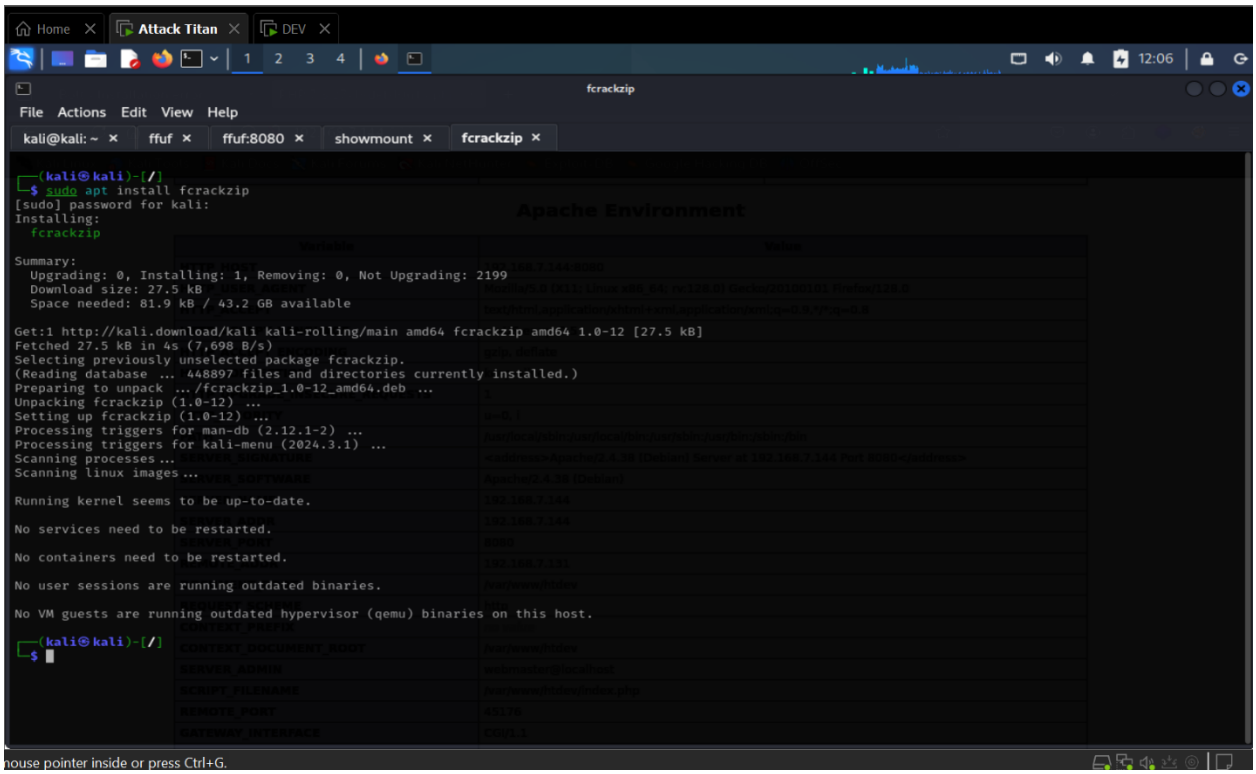
So, we navigate to the /mnt/dev directory and list its content. A zip file by the name “save.zip” is present and in order to know its content, we have to unzip it using the “*unzip*” command.



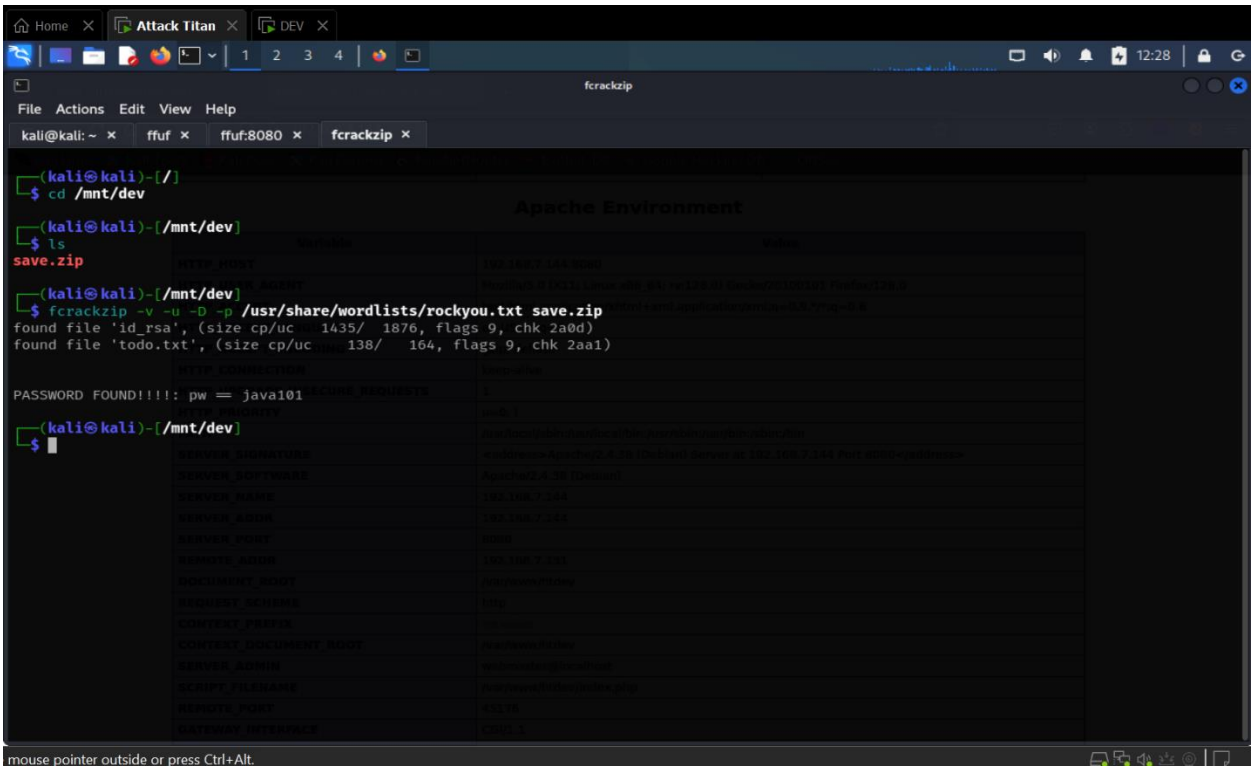
However, when we attempted to extract its content, a password was required. Since we do not have the password, we can crack the zip file with a tool called **fcrackzip**. On our attack machine, we don’t have this tool and so we will have to install it.



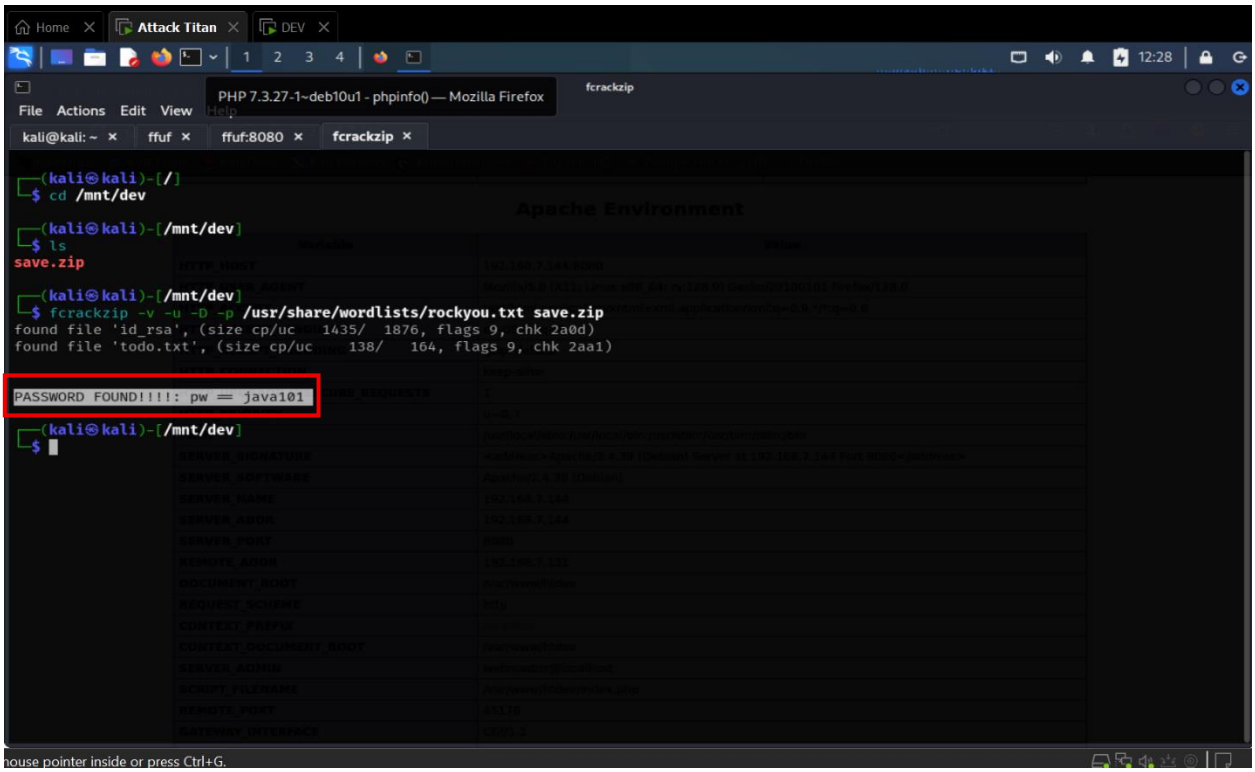
Installation of **fcrackzip** is taking place.



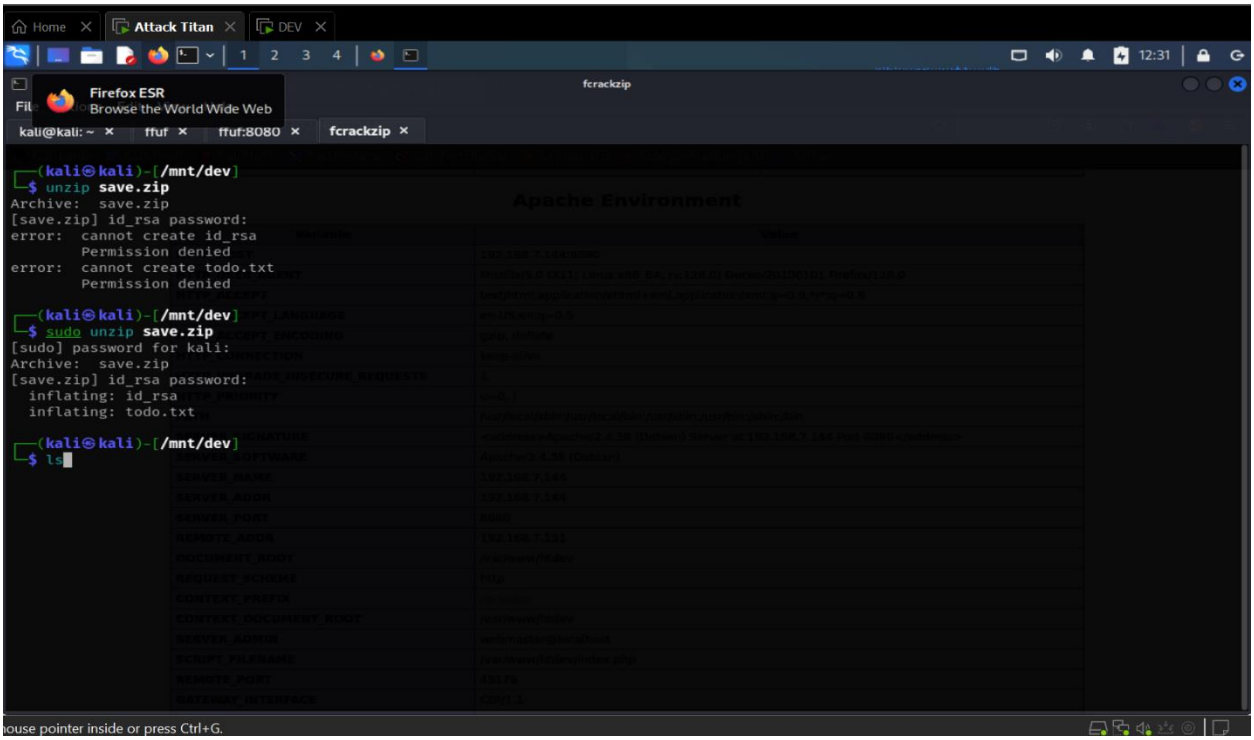
Installation is complete



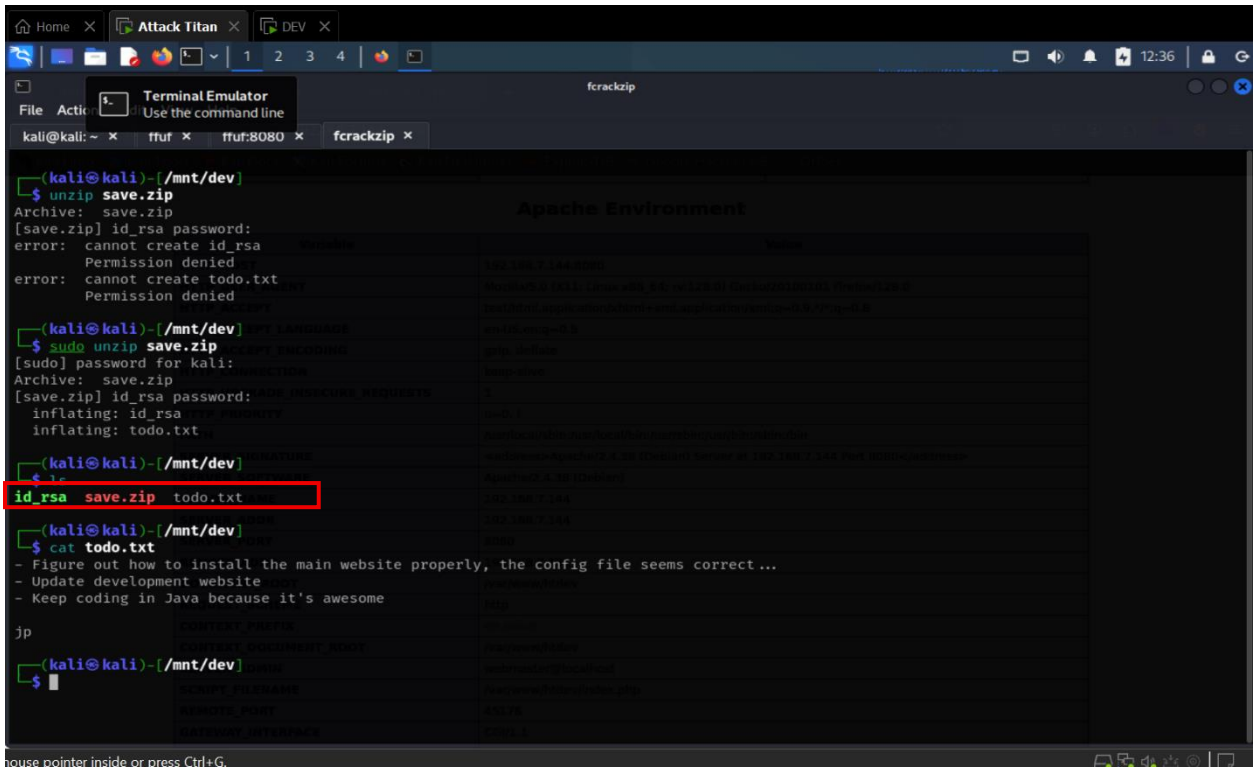
We locate the file we mounted earlier and start cracking out its password using **fcrackzip.**



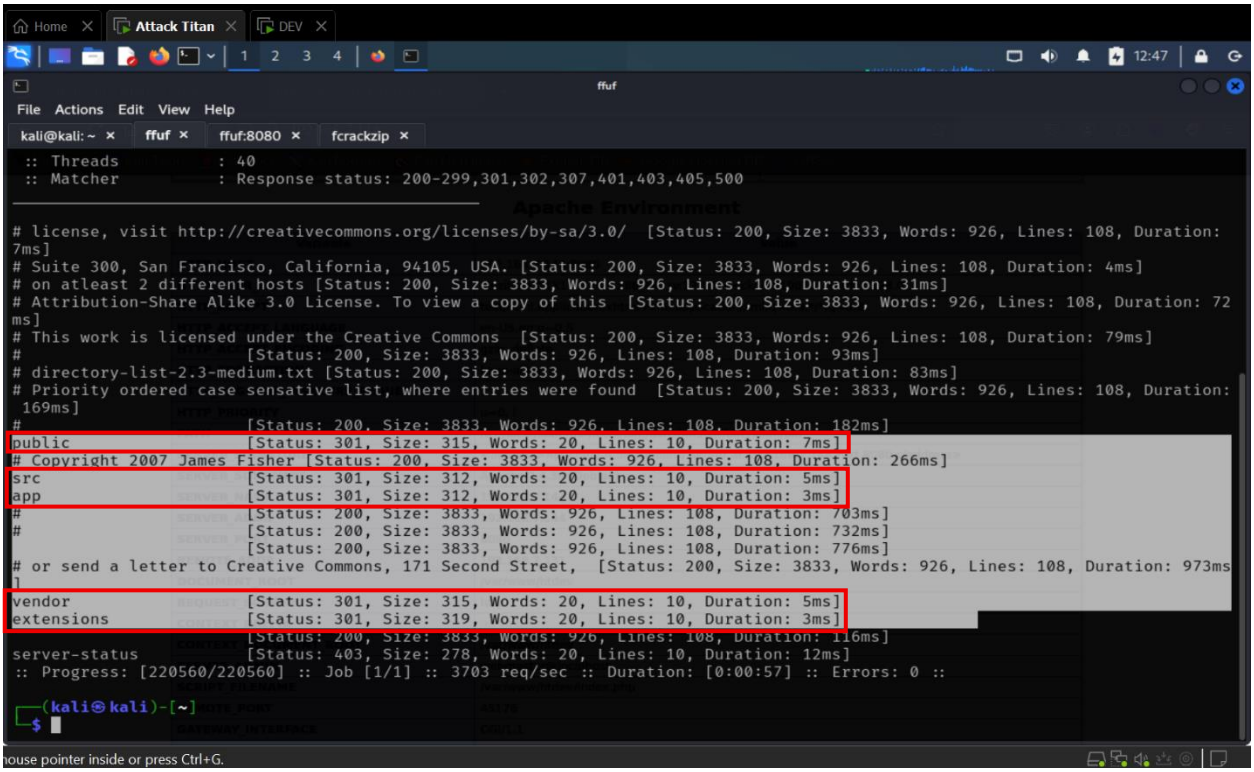
Cracking the zip file was successful and we were able to obtain the password



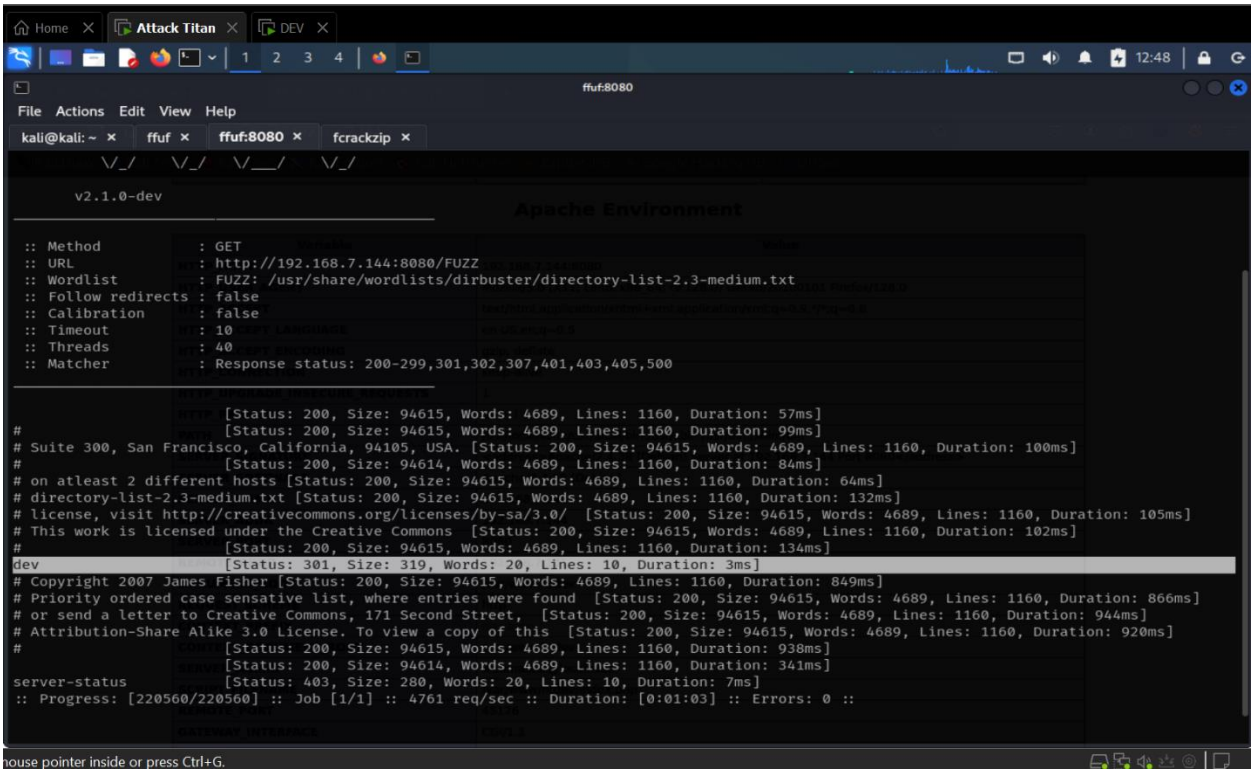
Now we should be able to unzip the file with the password we obtained and view its content.



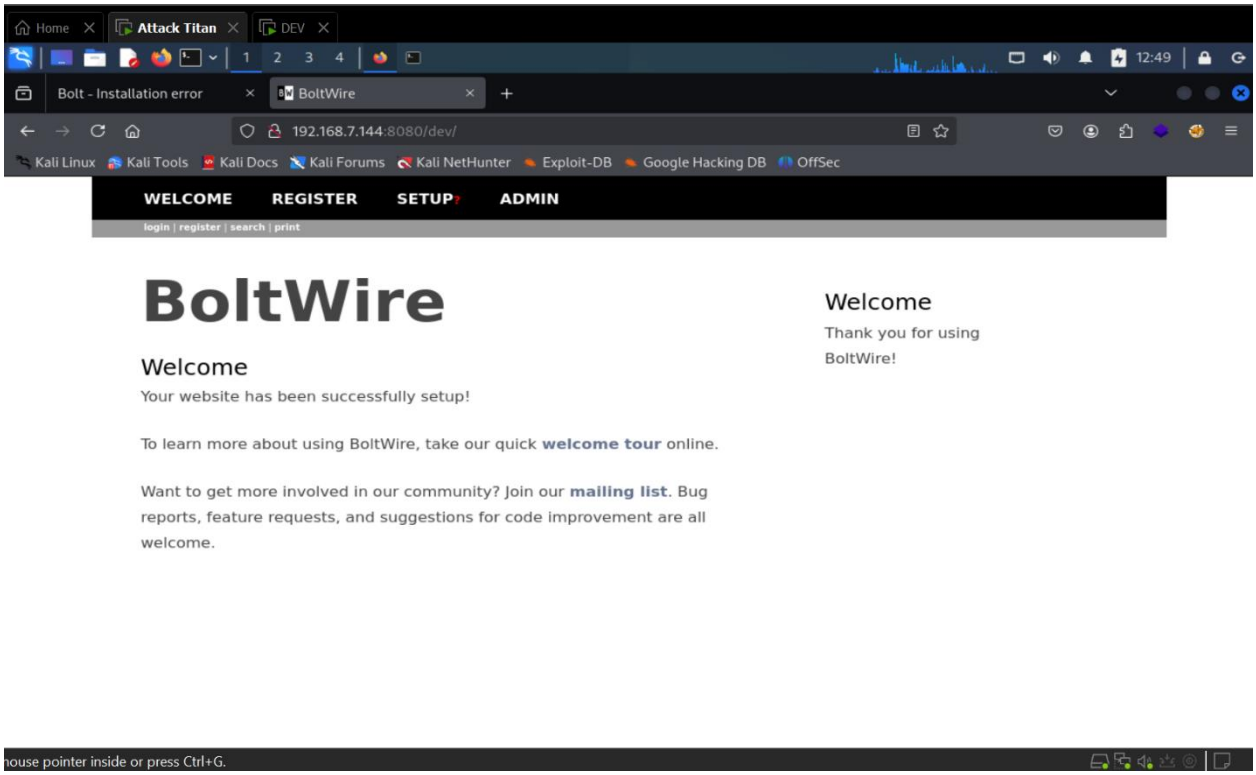
We get two files namely “id_rsa” which suggests we can connect through SSH and “todo.txt” which is, of course, a text file. We view the contents of the text file and get information about somebody called “jp” who talks about a config file concerning a website and seems to love java.



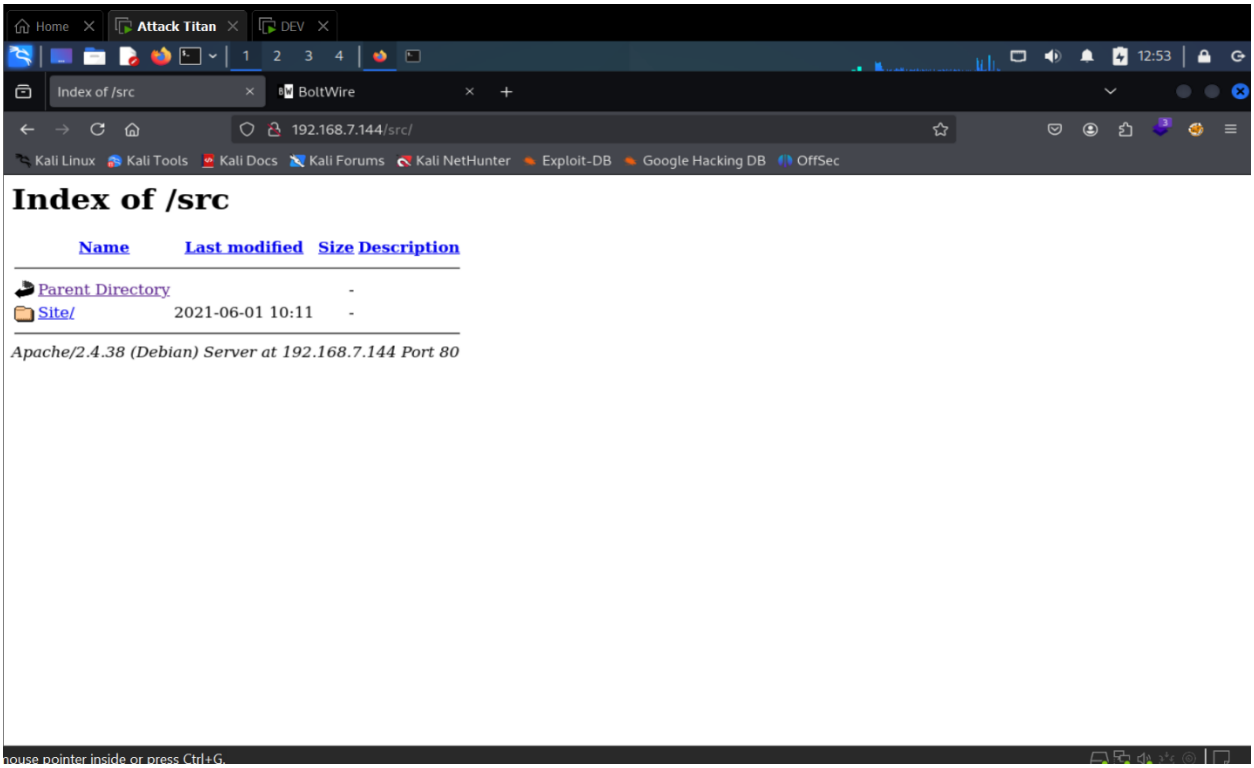
Back to busting directories, we get a number of hidden directories as highhlighted for port 80



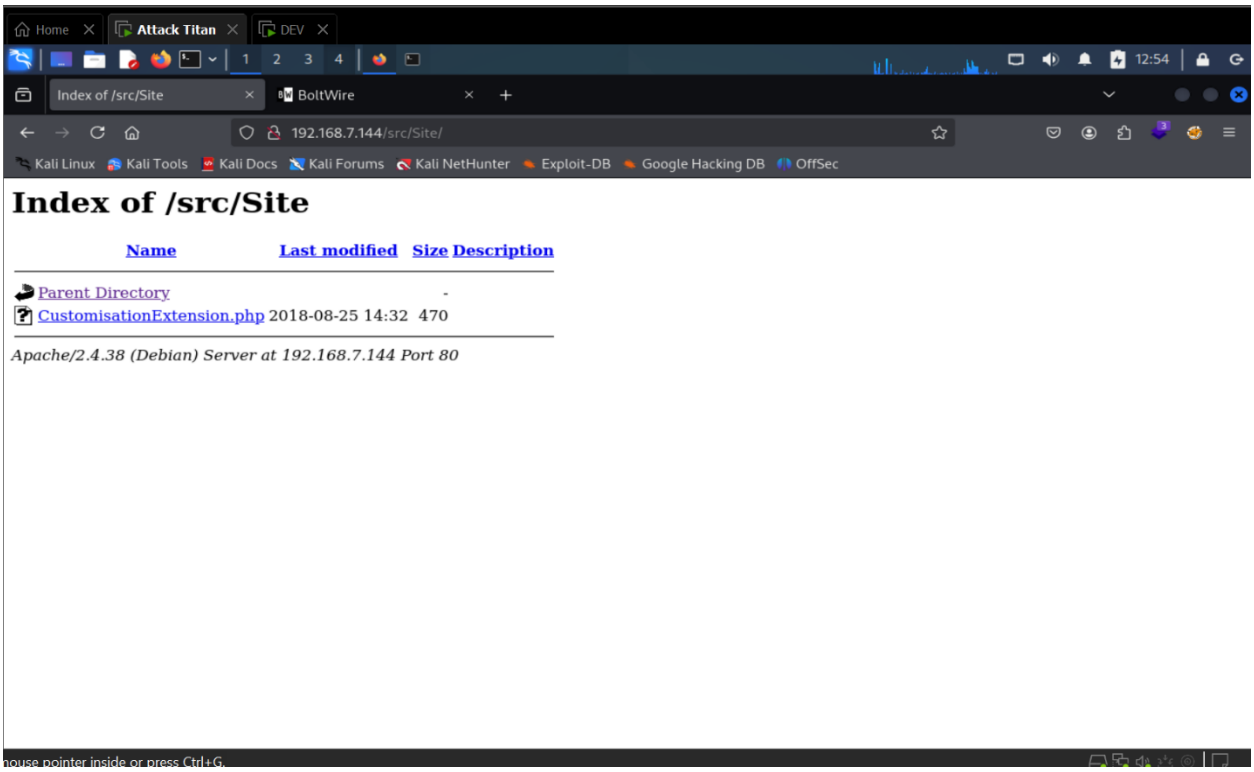
And for port 8080, we get only /dev which should be very interesting.

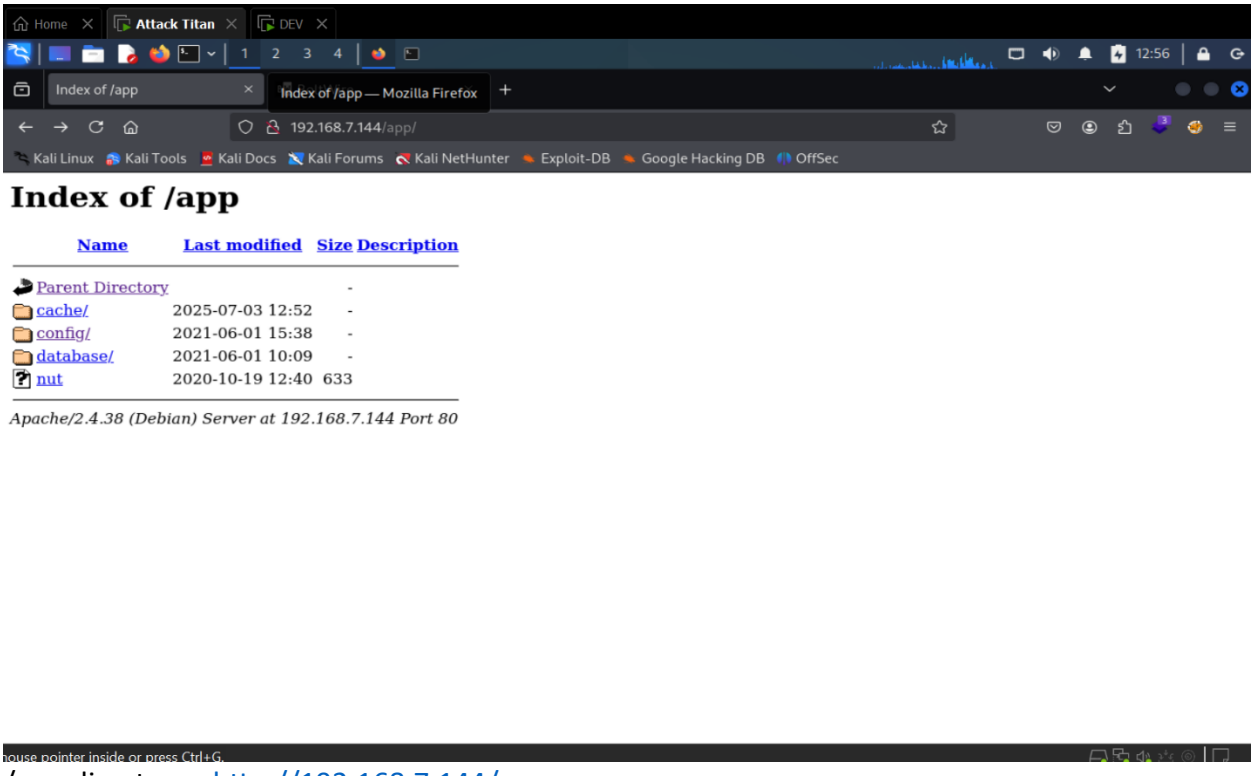


This is the /dev/ directory we opened at port 8080 which finally brought us to the Bolt platform. We explore its features to know if we can take advantage of those.

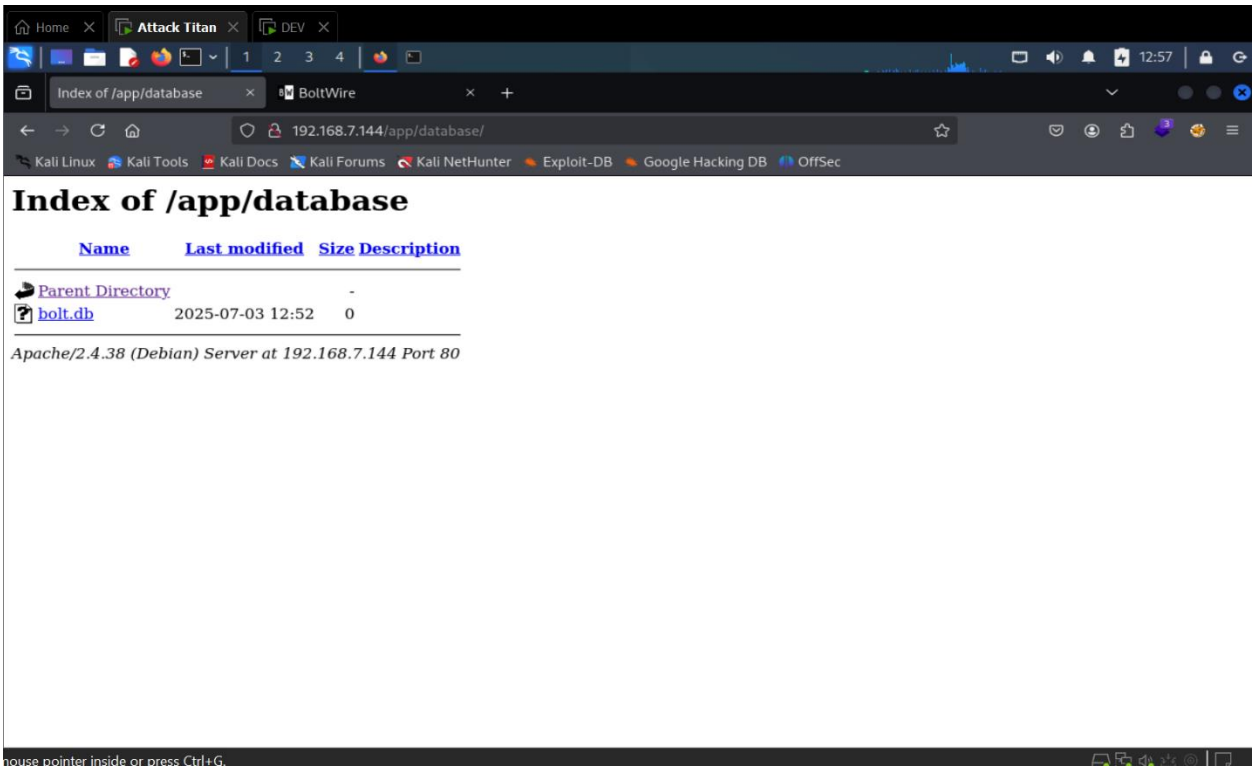


On the hand, we explore the other directories on port 80

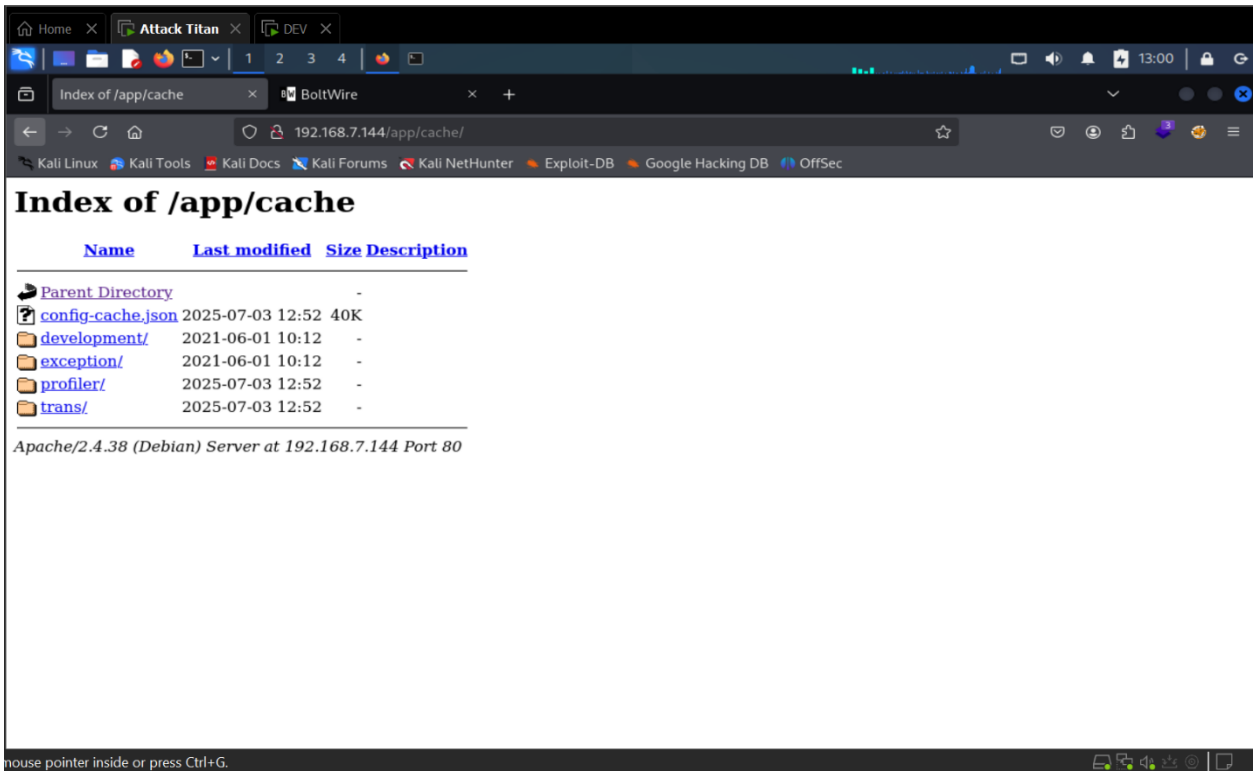


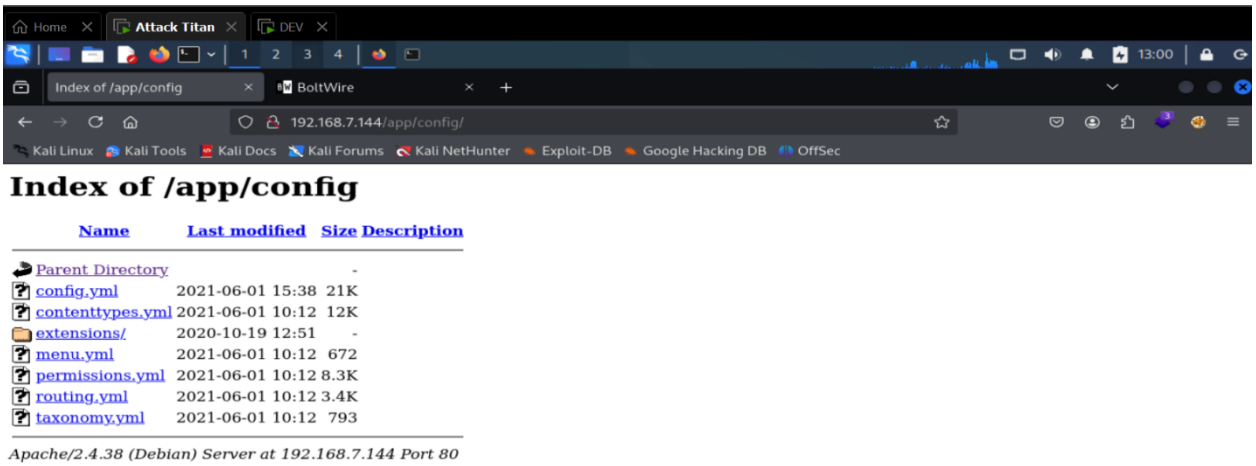


/app directory - http://192.168.7.144/app

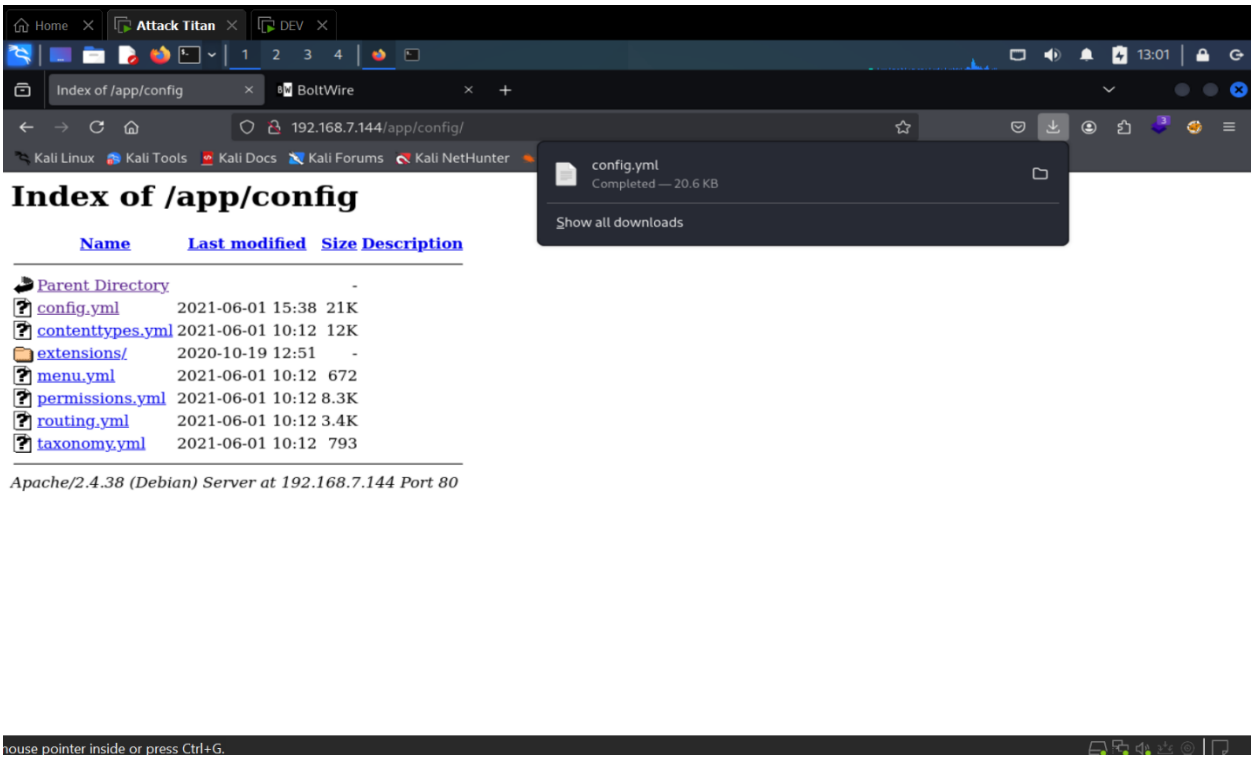


Over here, we have a bolt database file here but the size description indicates its empty.

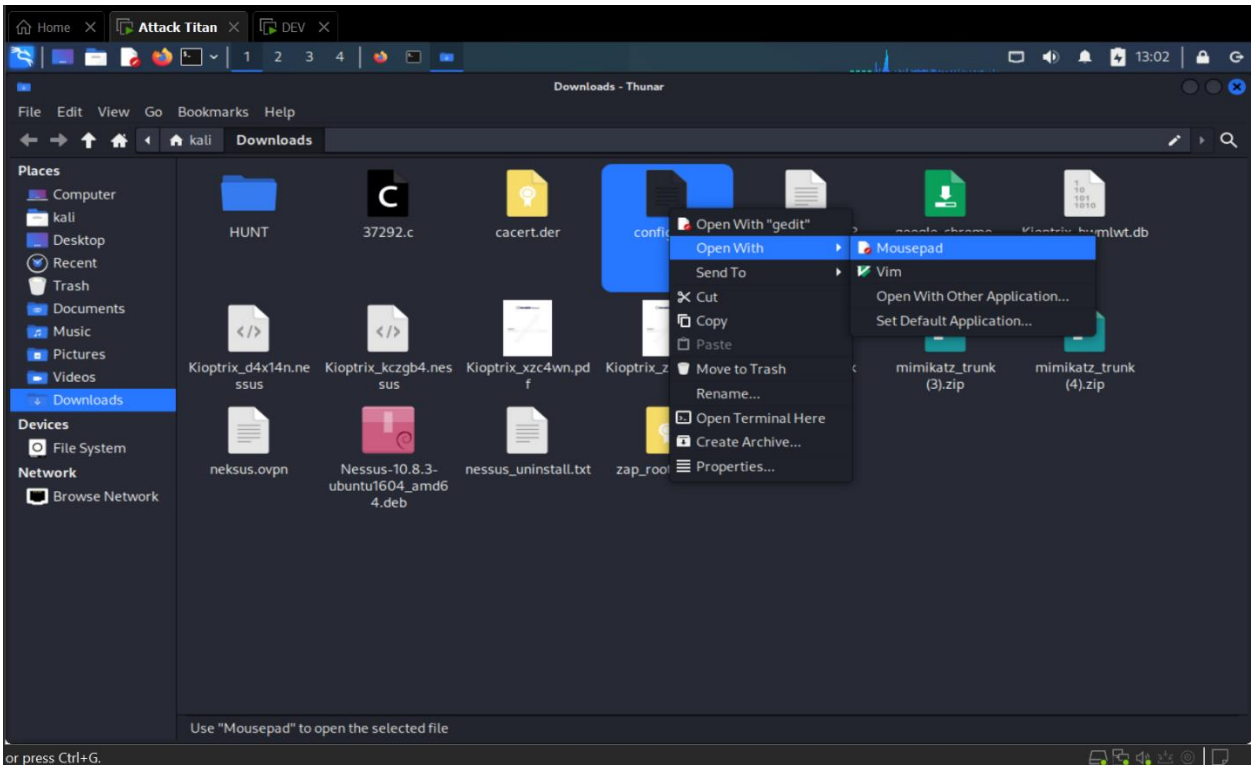


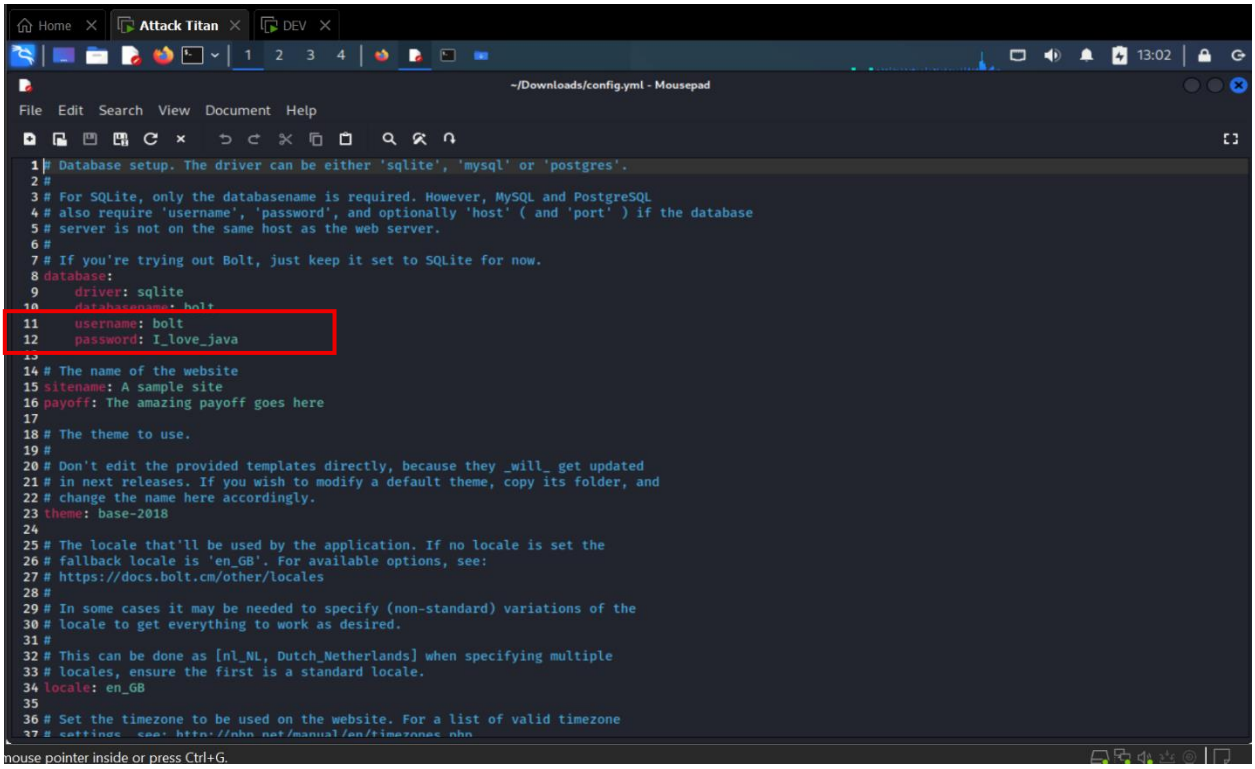


If we remember the information from the todo.txt file, it spoke about a “config” file that was needed to setup the website.

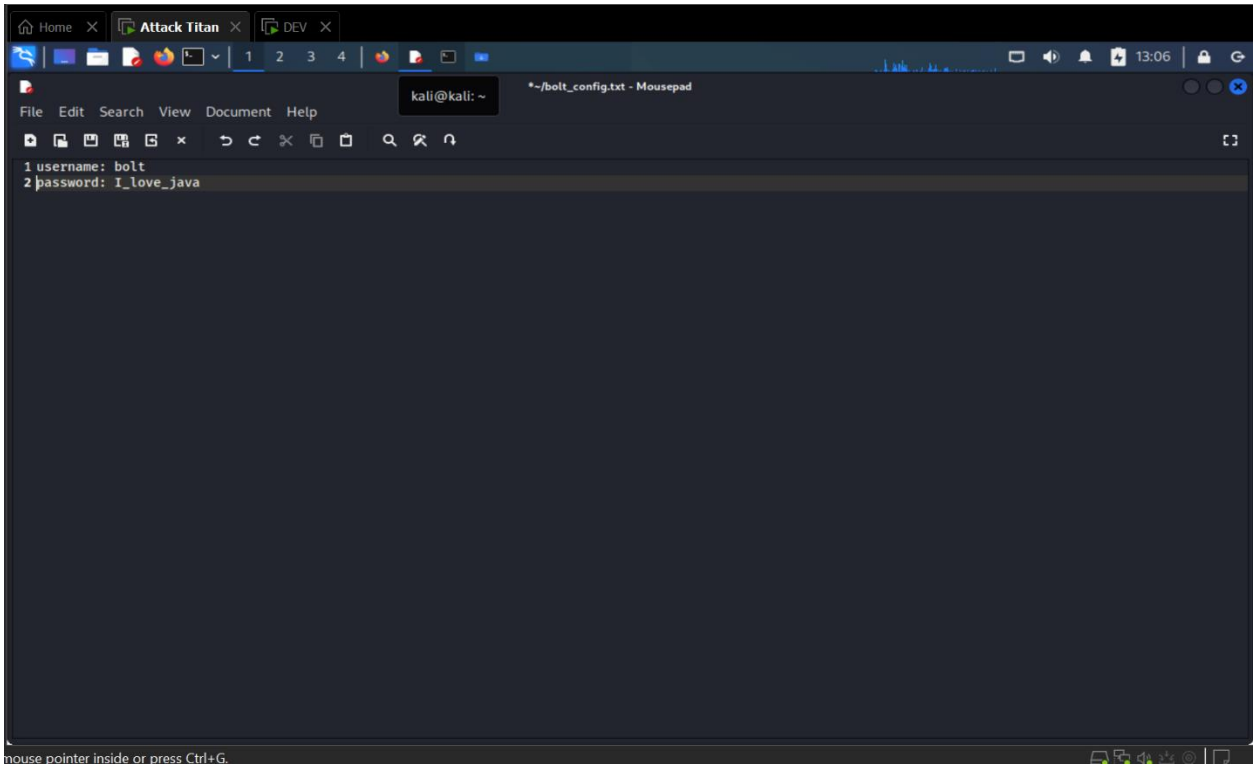


Since we have a “config” file in the /app directory, it will surely guarantee us some precious information and therefore have it downloaded onto our attack machine.

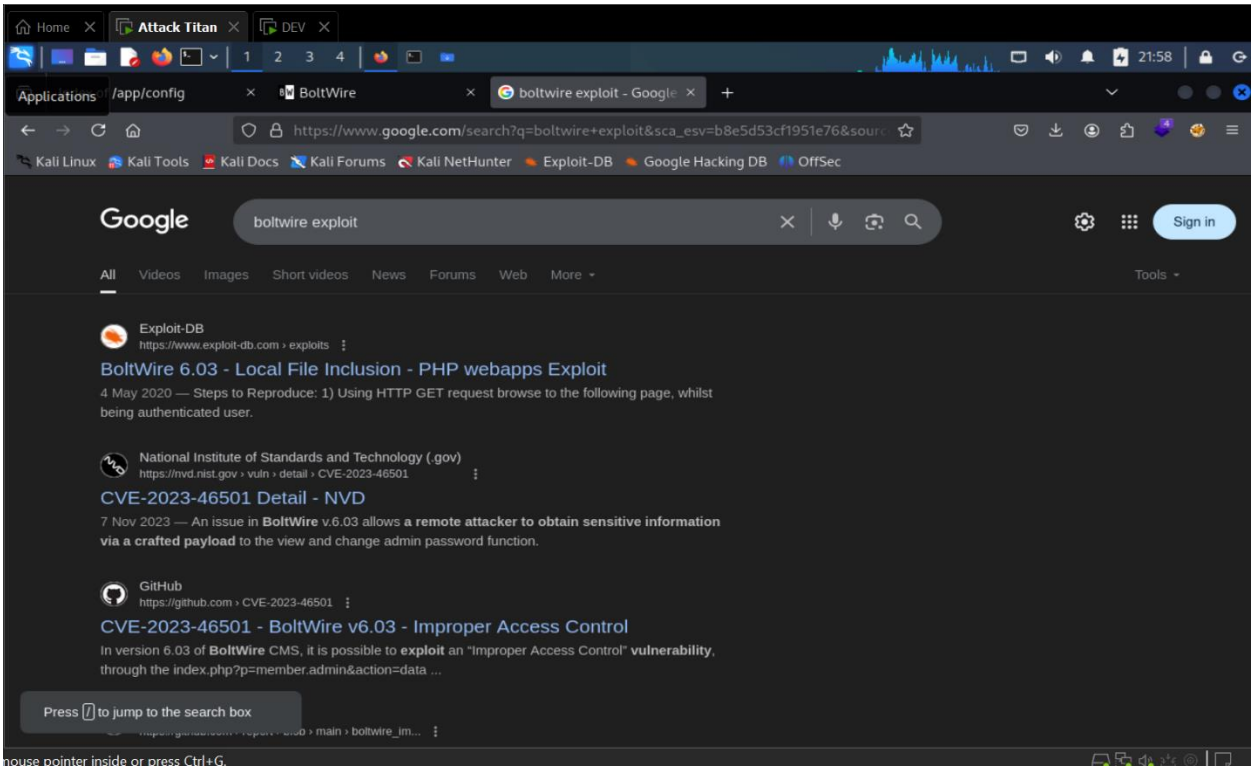




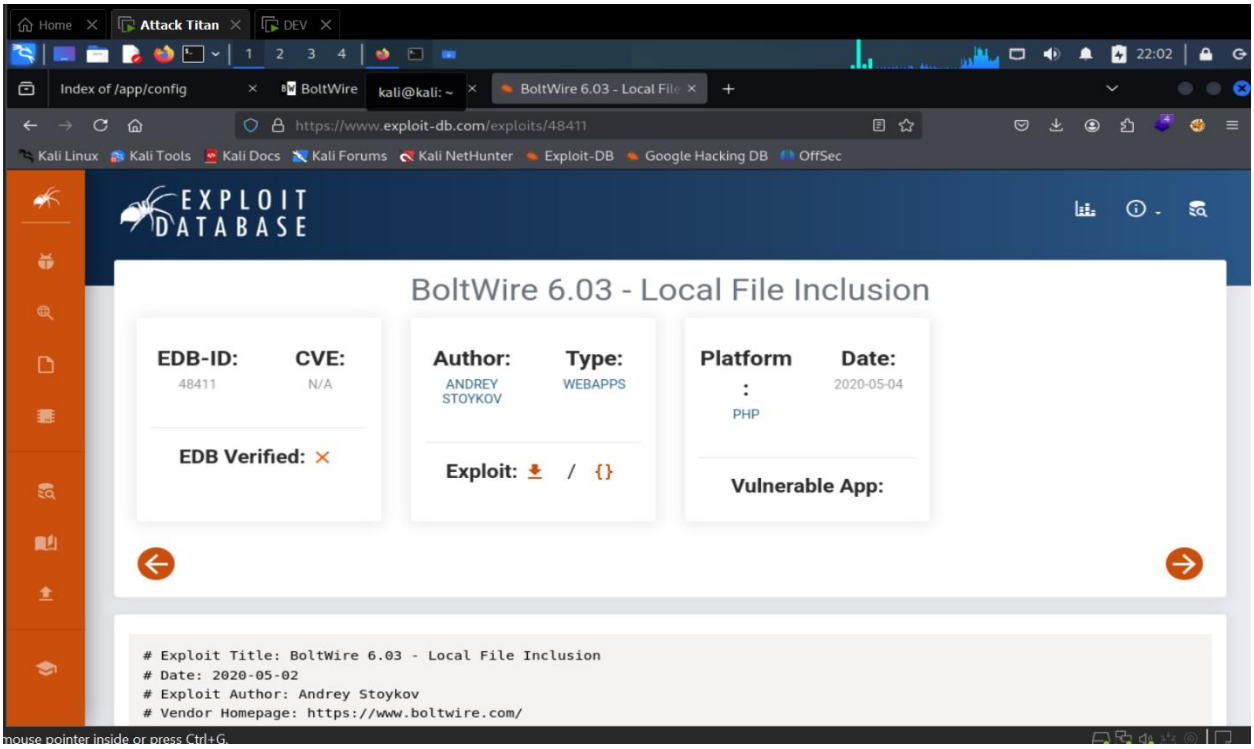
After glancing carefully, we were able to spot some credentials in the file.



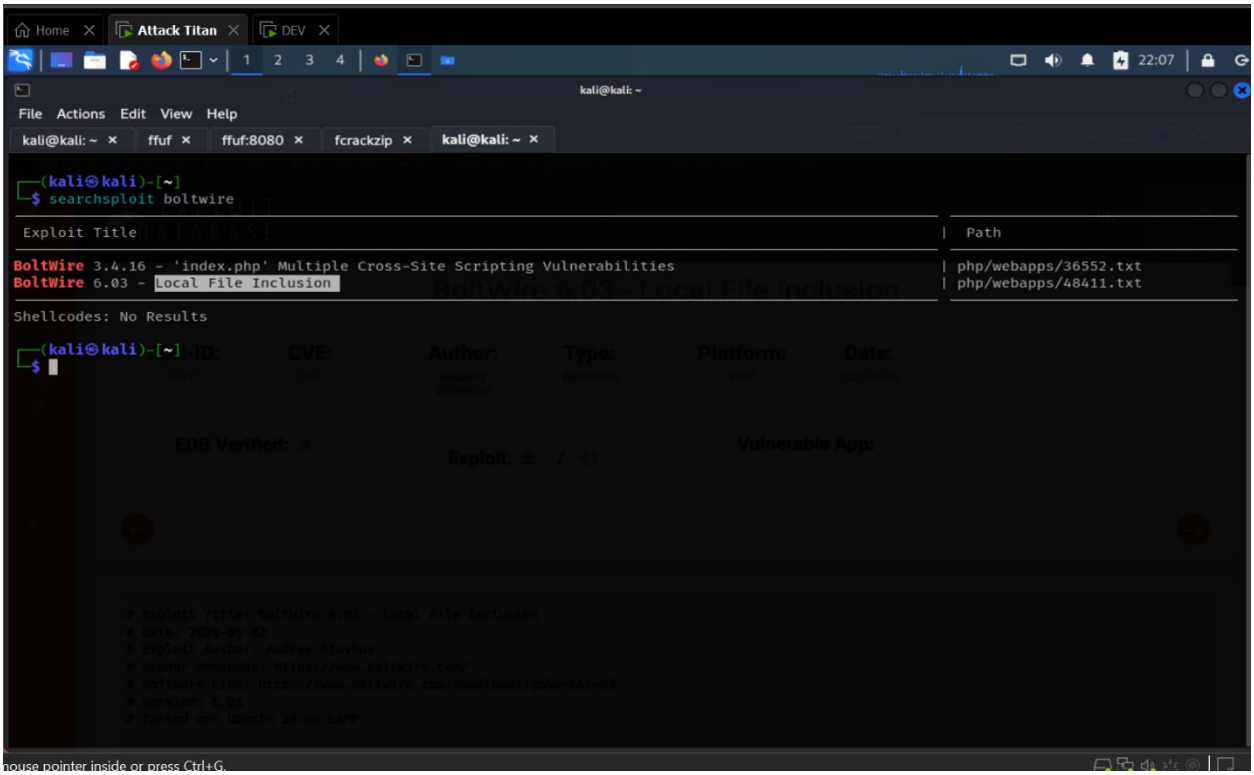
As part of our note keeping, we store these credentials somewhere for later use.



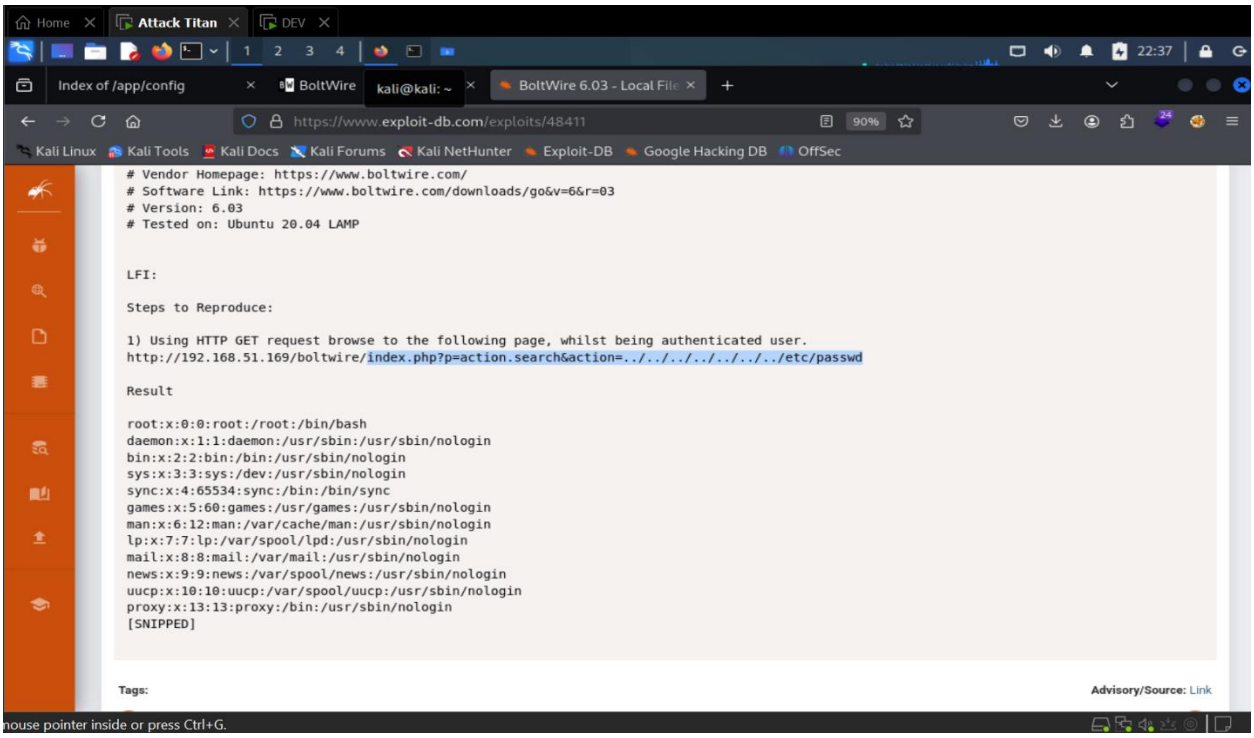
After exploring the bolt platform features, there wasn't much we could use to our advantage. At this, we had to search for existing exploits for that particular platform and look what we got.



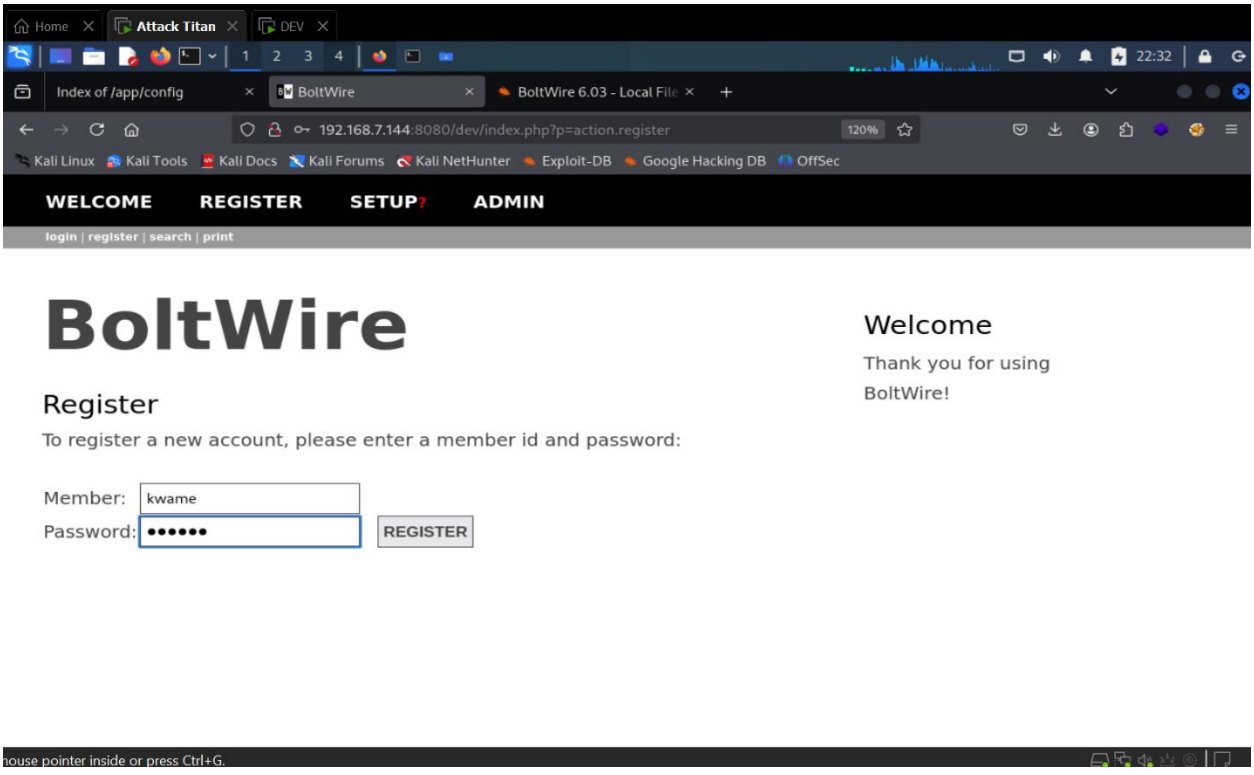
Local file inclusion, known as LFI, allows us to execute files that are already present on the server.



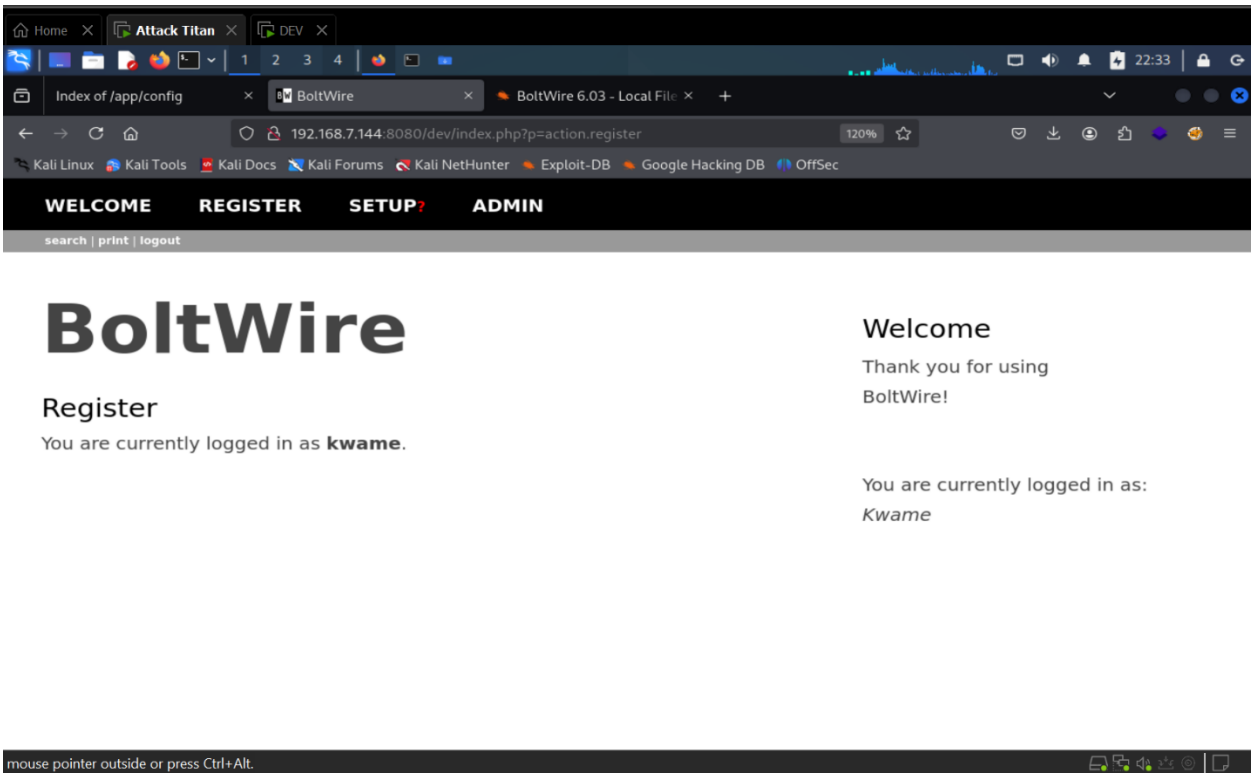
Kali linux has its own local database of exploits known as **Searchsploit** we can utilize without having to search online. For this scenario, we will stick to the “Exploit Database”.

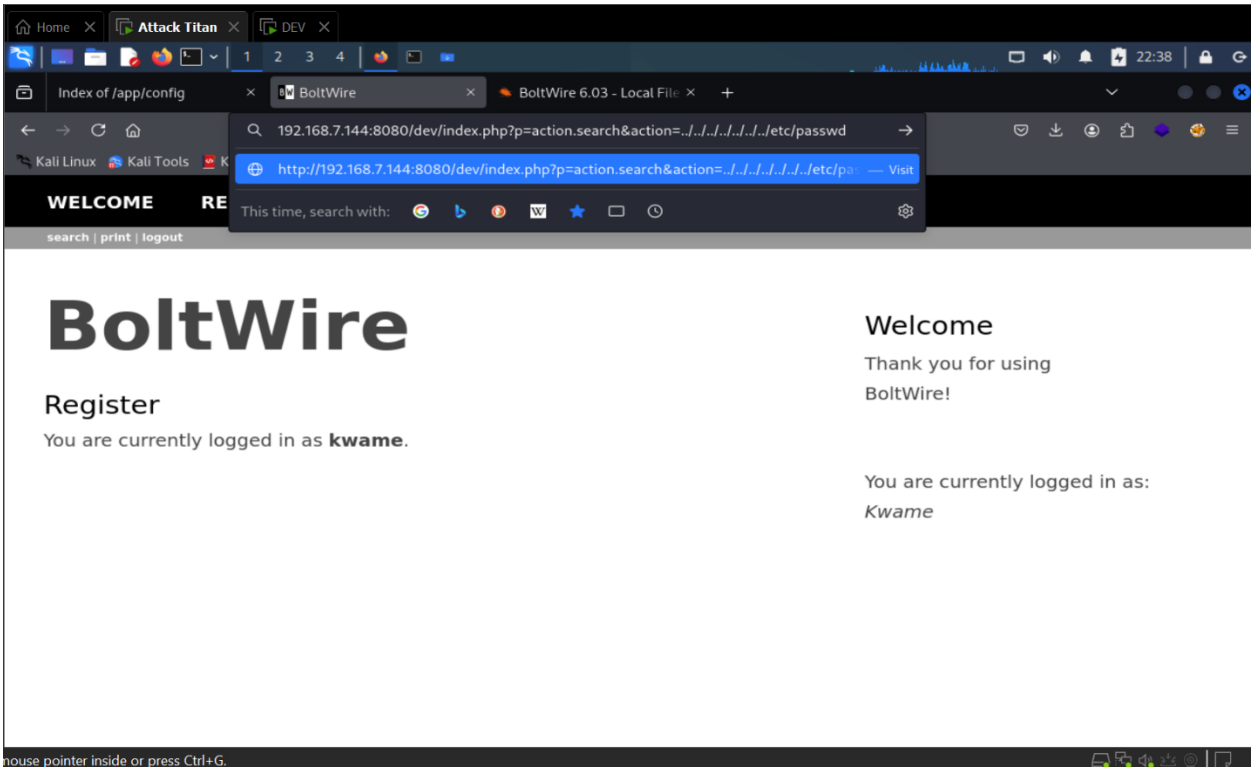


In order to run the code and achieve the results as demonstrated, we are told to be an authenticated user first before this attack is successful.

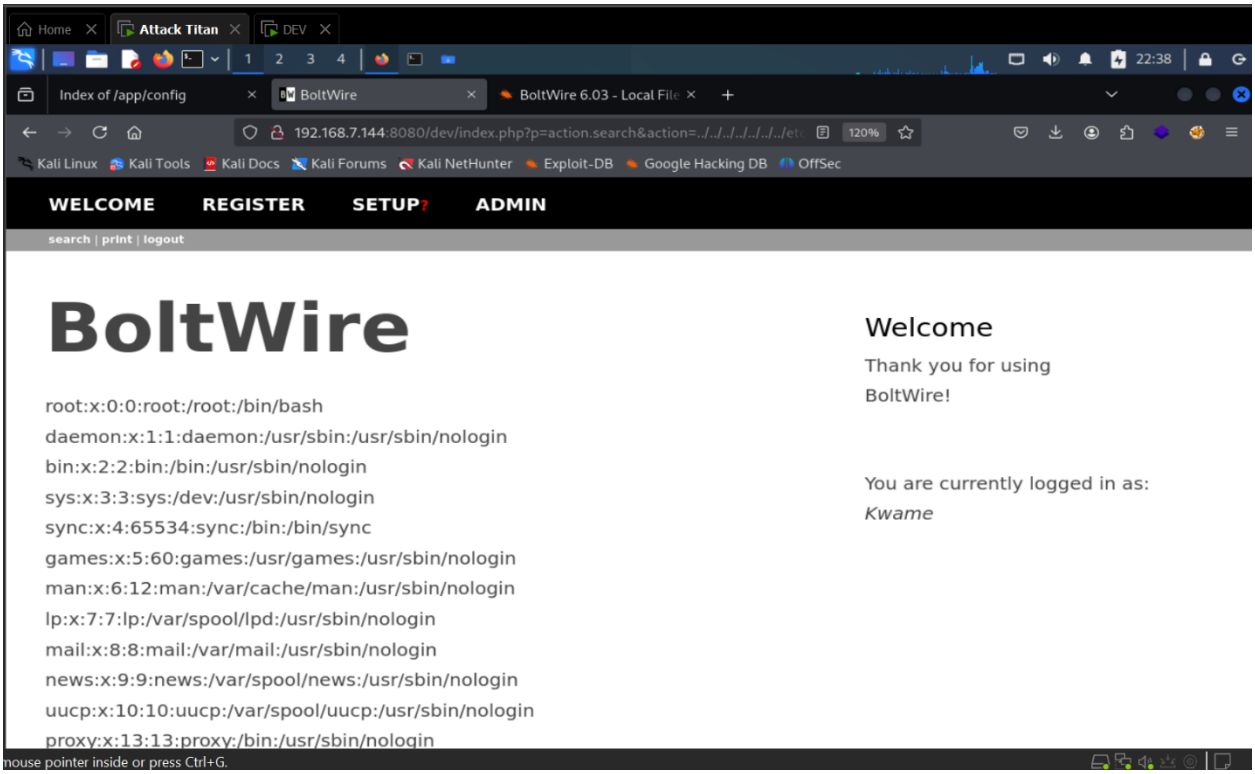


So we create a user by providing a username and password.

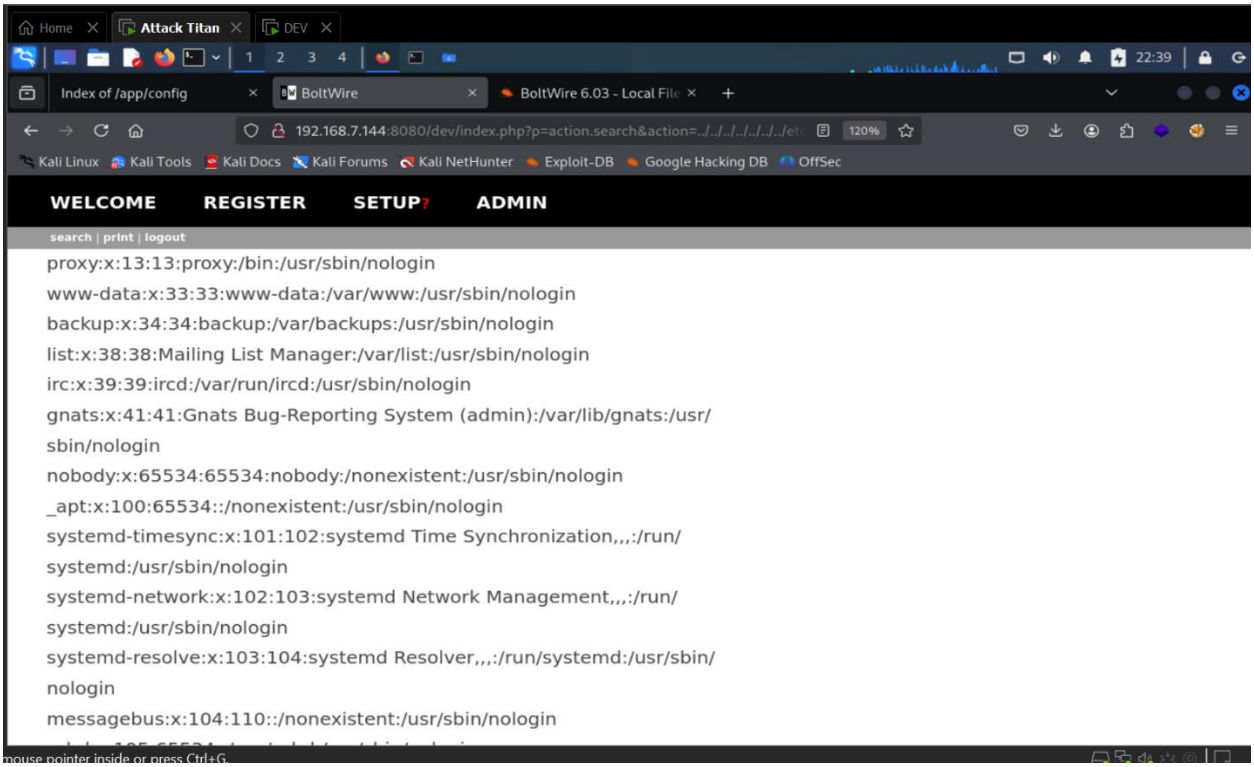


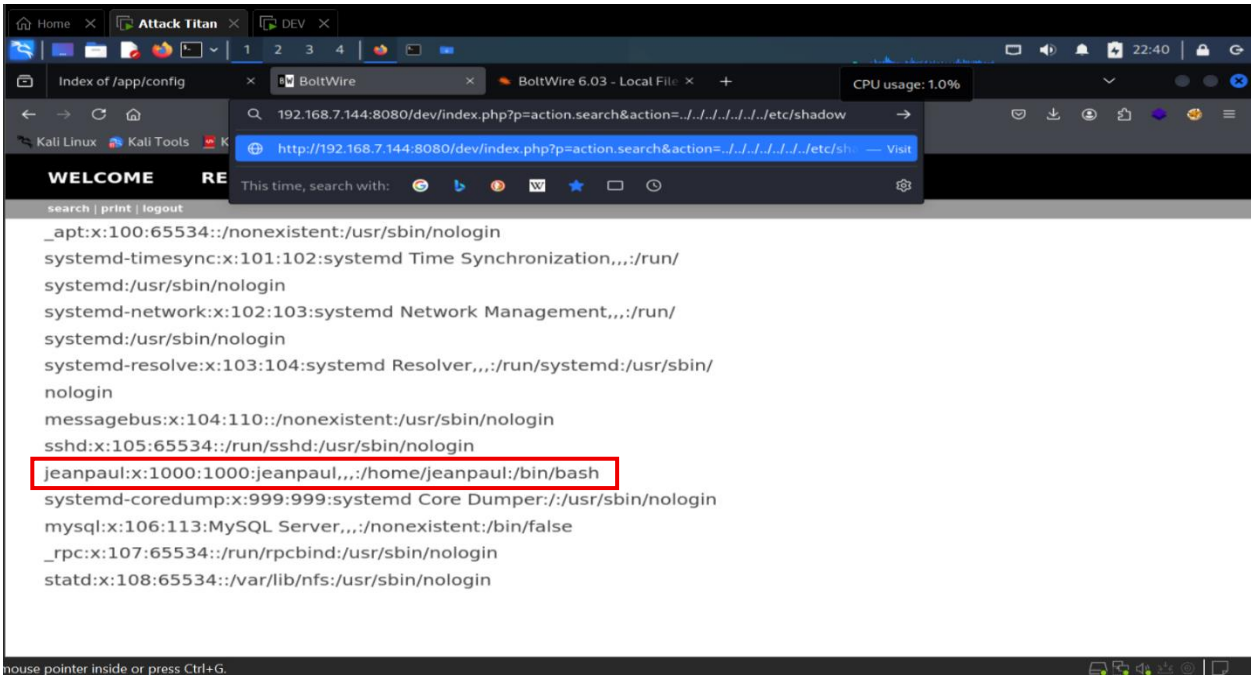


And now we insert the line into our URL search bar and hit “Enter”

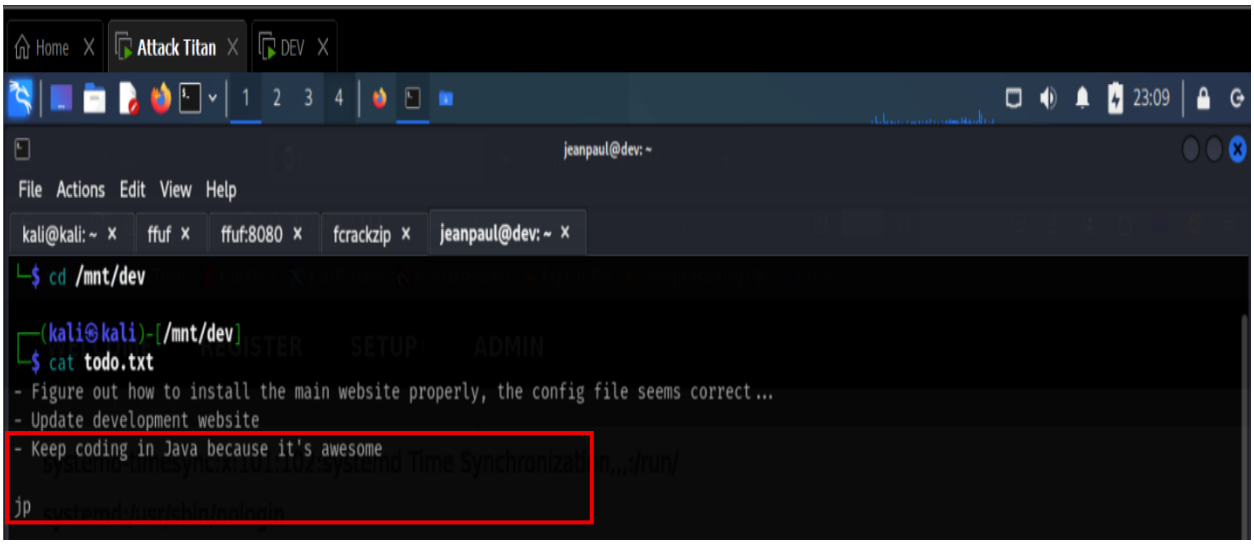


Once again, we got something interesting to work with as we have been presented a list of users, services, etc. on the system.

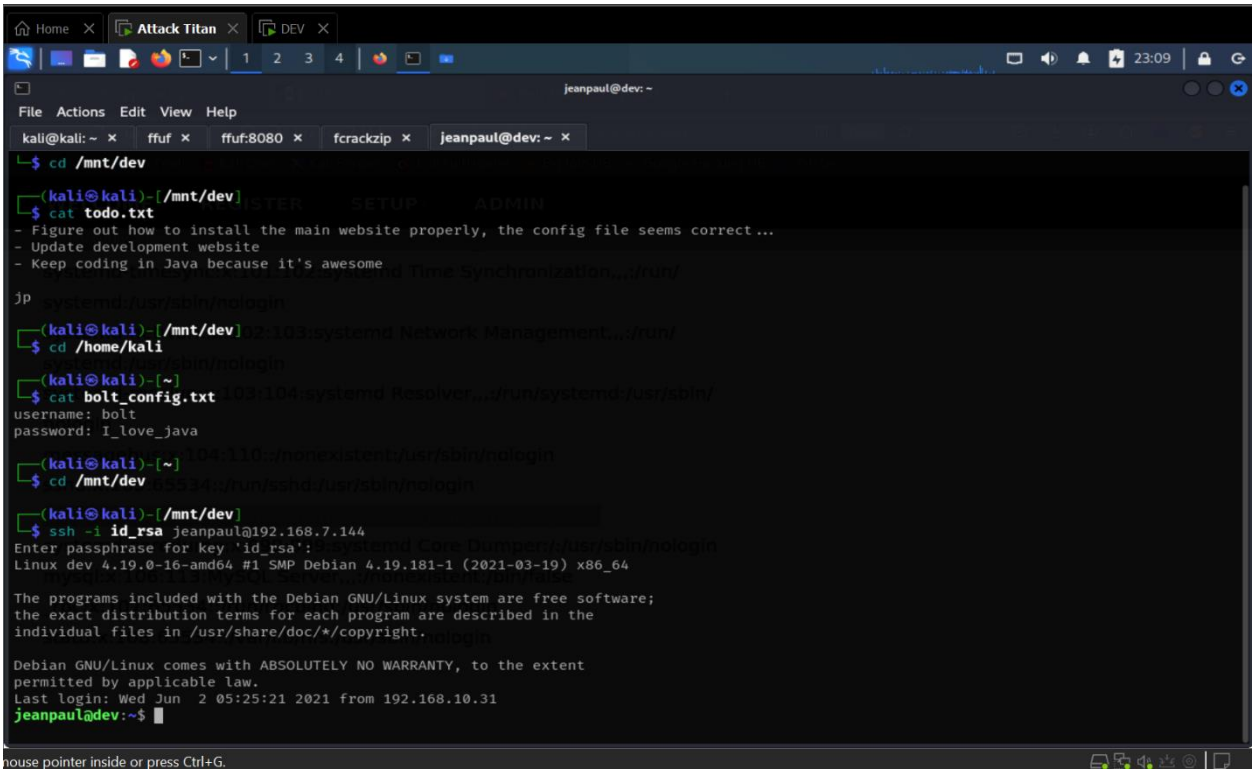




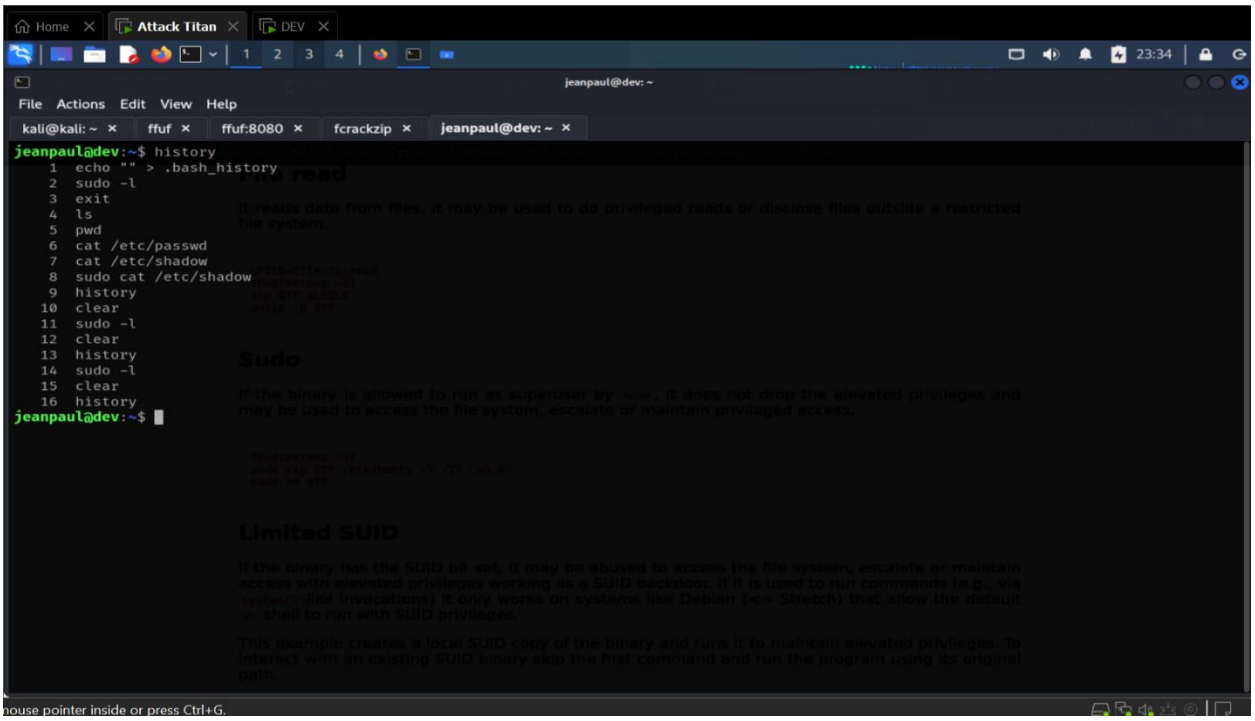
Now, we can spot an interesting name and it seems to tally with the “jp” abbreviation from the todo.txt. If that is the case, we could use this as our username to log through SSH (last port to touch on).



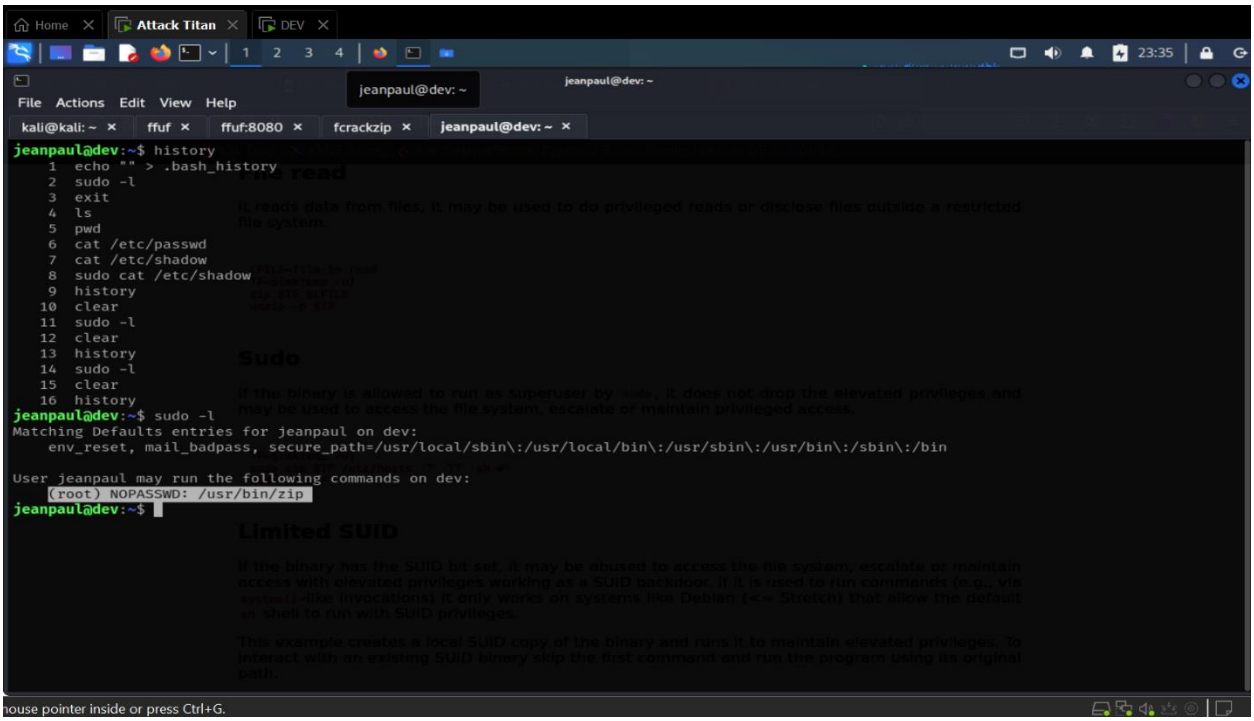
Also, if we can recall, we stored some credentials from a config file. The password we got suggested something interesting, I_love_java. By now, we all know one guy who loves java on this system and it’s our very dear friend, jp.



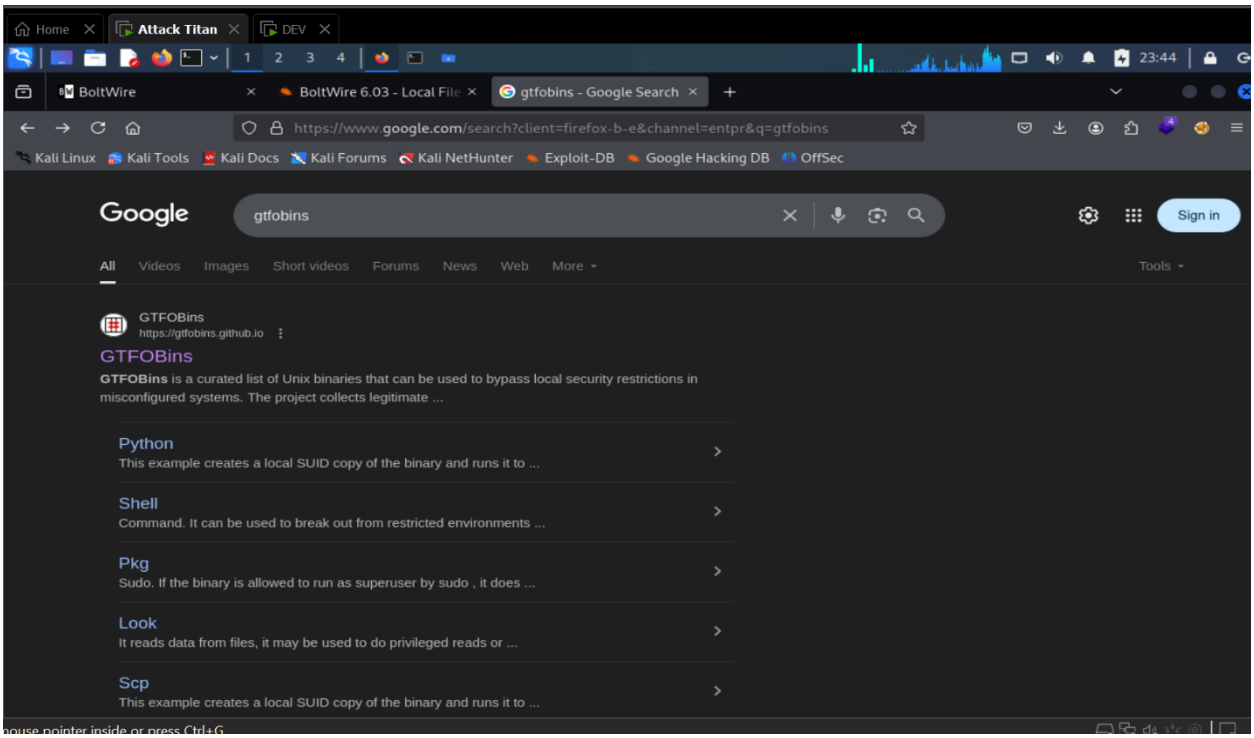
Using the id_rsa private key along with the username “jeanpaul” and password “I_love_java”, we successfully authenticated and gained access to the system.



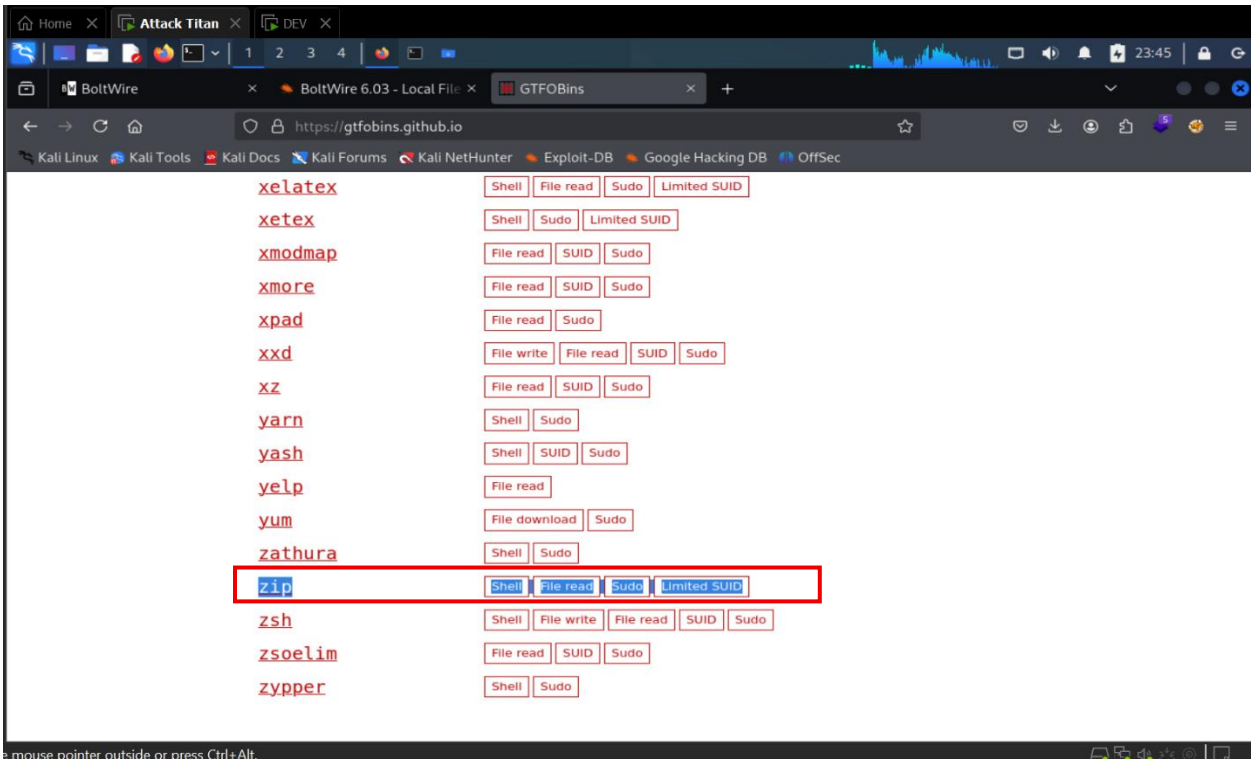
Now that we are on the system, a normal user won’t cut it if we want to do some high privilege tasks. So, we explore avenues we can abuse to our advantage.

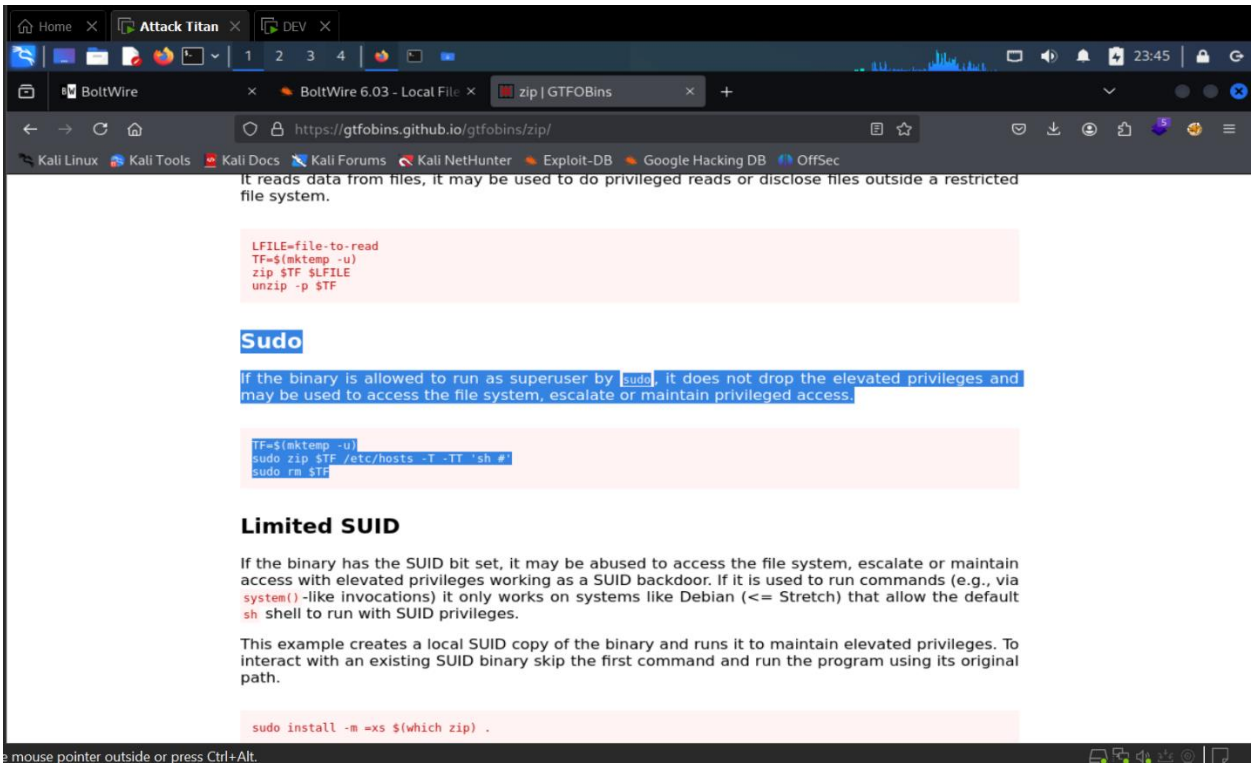


“sudo – l” allows to identify binaries we can run with “sudo” in our current user account. Here, /zip is the only binary we can use with “sudo”.

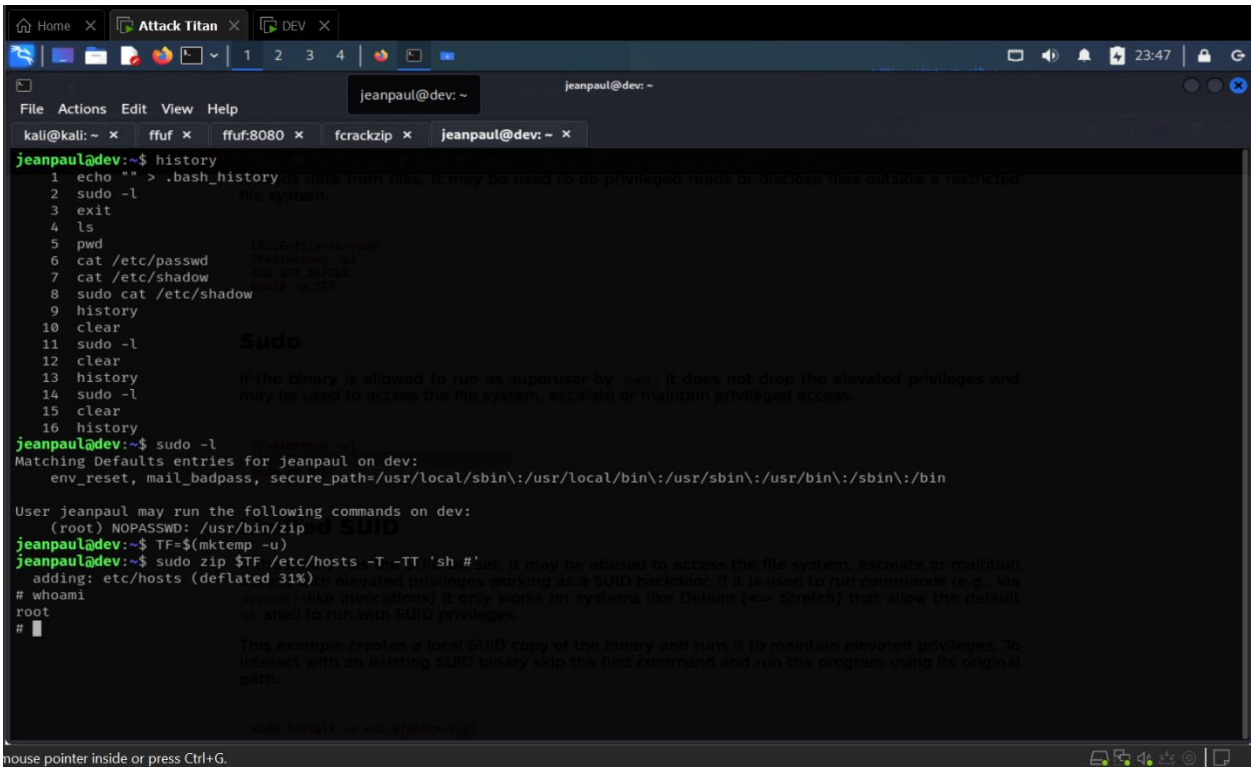


With that said, we can escalate our privileges with the help of the GTFOBins site which is known for providing a list of linux binaries that evade security implementations.





In summary, we are told to run the following code while using “sudo”.



After executing the code accordingly, we finally achieved root privileges.