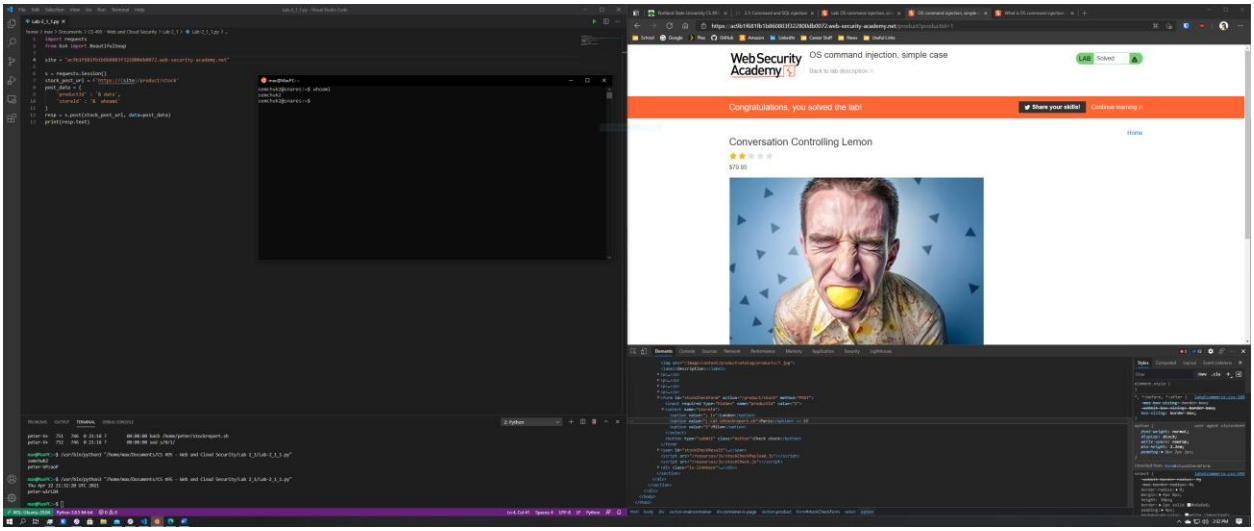
Contents

Labs 2.1.1 simple

1: os-command-injection:

Code Part :

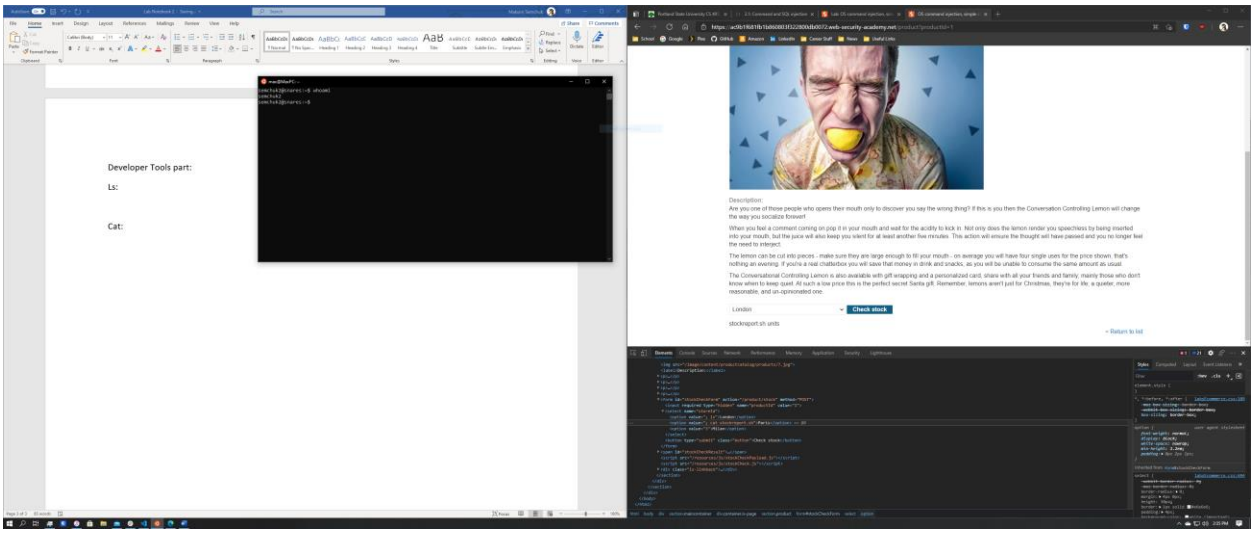


This was an attack on the fact that due to legacy support, there needs to be an easy way to grab products. At the same time, the server needs to execute a shell script to get stock information.

This was exploited to get the echo for my username and whoami to execute since they were passed in as command line arguments.

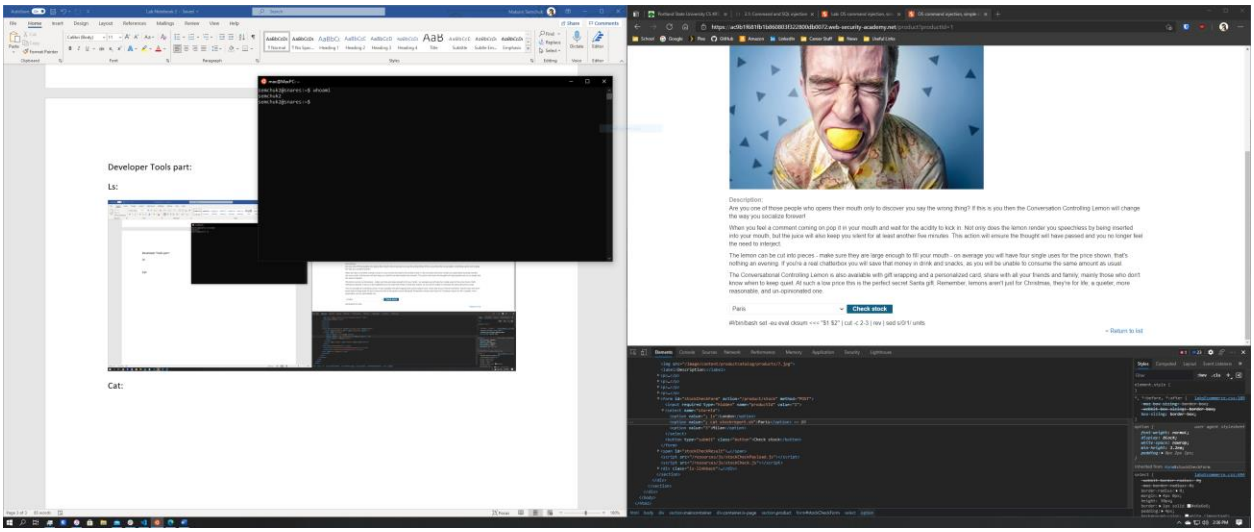
Developer Tools part:

Ls:



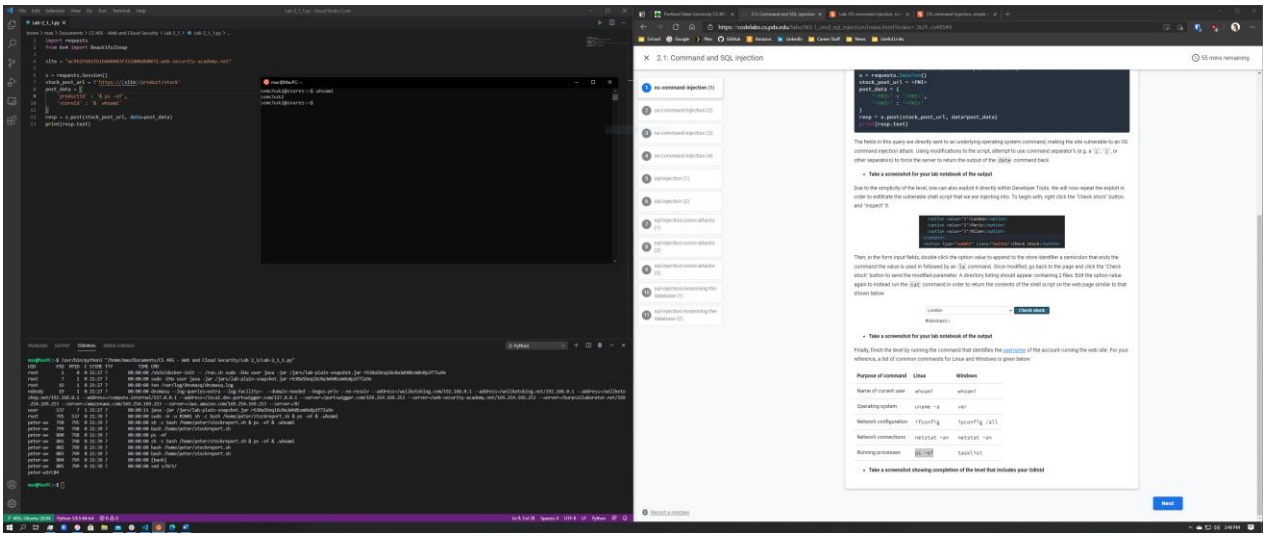
The ‘ls’ command allows us to display every file in the directory, including other directories.

Cat:



Editing the line, will allow the same effect as sending the code. This will get the cat command to display the code running inside of the shell command.

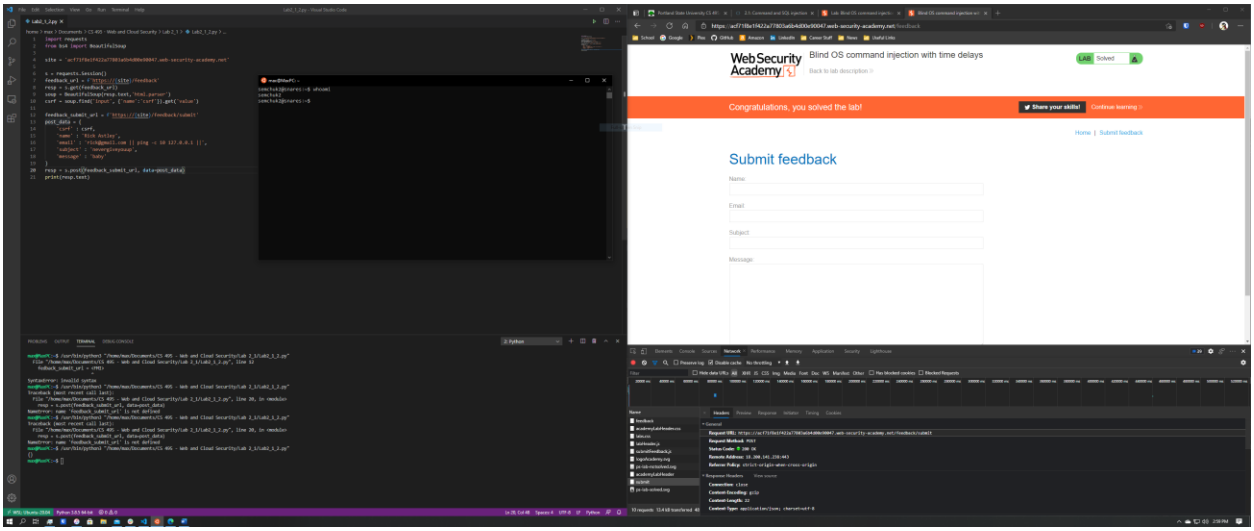
To finish the lab:



The command was running to show it can be executed on the system. This vulnerability relies on the fact that the commands are able to be escaped and run in the server. This is able to prevented by making sure that only certain characters are allowed to be input.

Labs 2.1.2 blind-time-delays

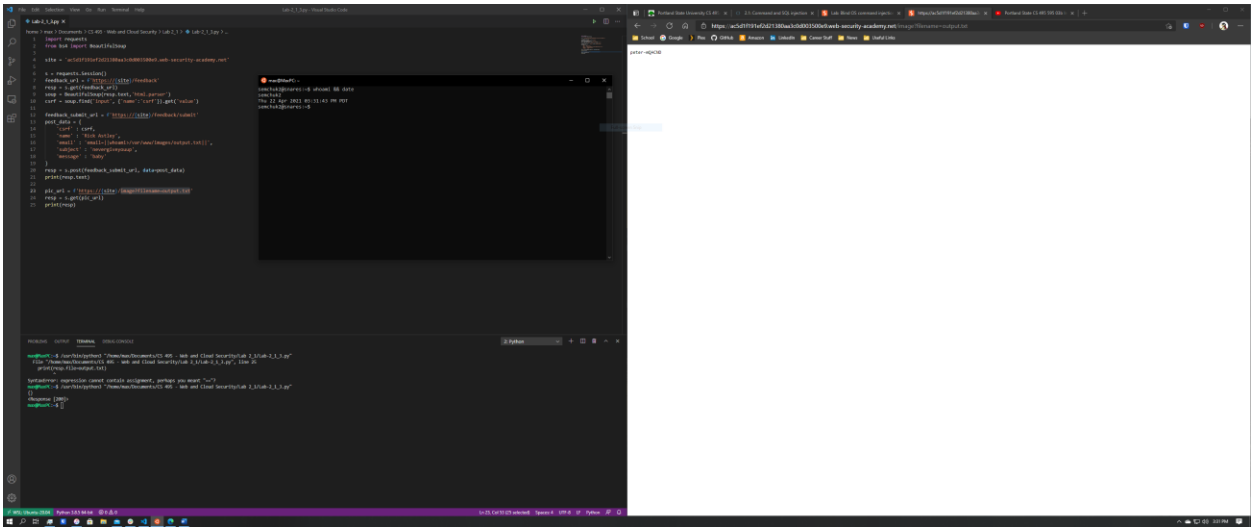
blind-time-delays



In this lab, it does not return anything. A way that a user can find out if a command is running by a command that takes time to execute. In this case, I was able to escape into the command line by using a $ follow by a command.

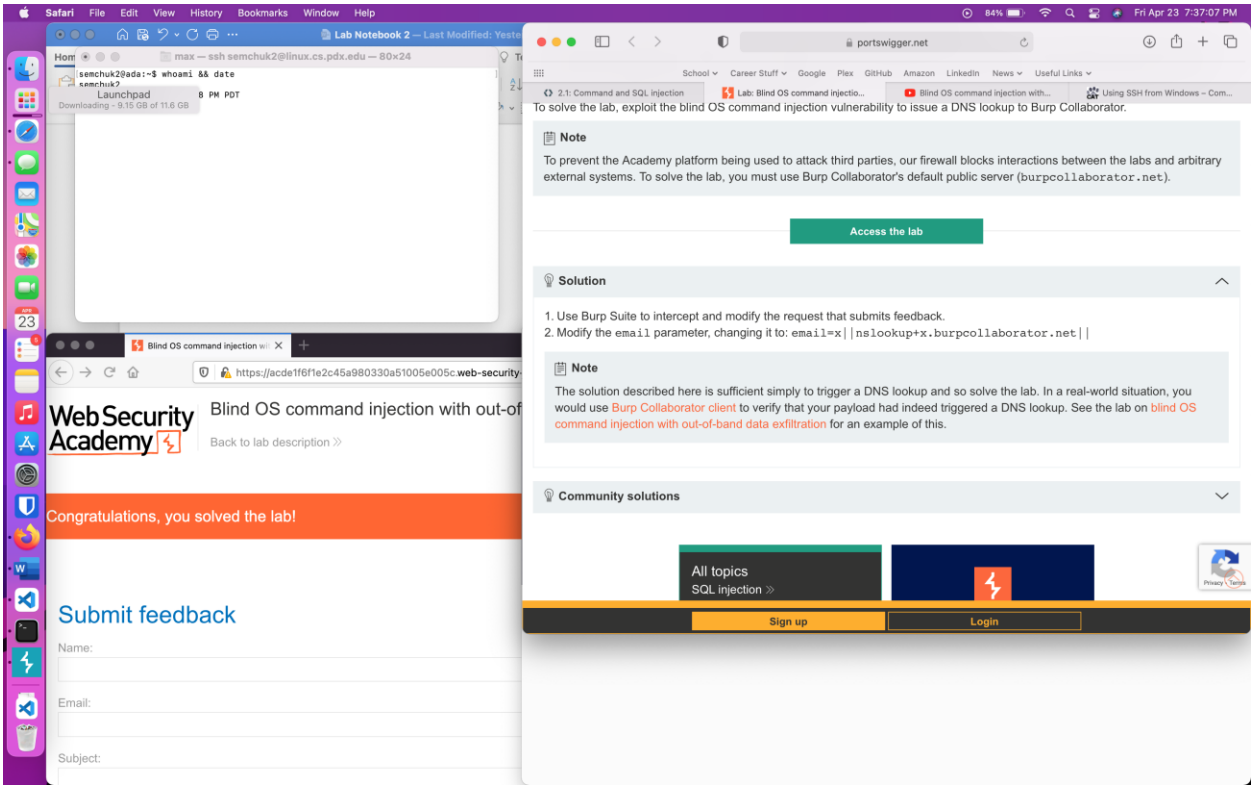
Labs 2.1.3 blind-output-redirection

blind-output-redirection



This allows users to create a file on the images directory, which is bad. A user should never be able to create a file without a specific allow list, in specific permissions. Preventing users from running commands like redirects is really important as well, since you are able to run in the command and put the output into other parts of the system potentially. All those things should not be allowed.

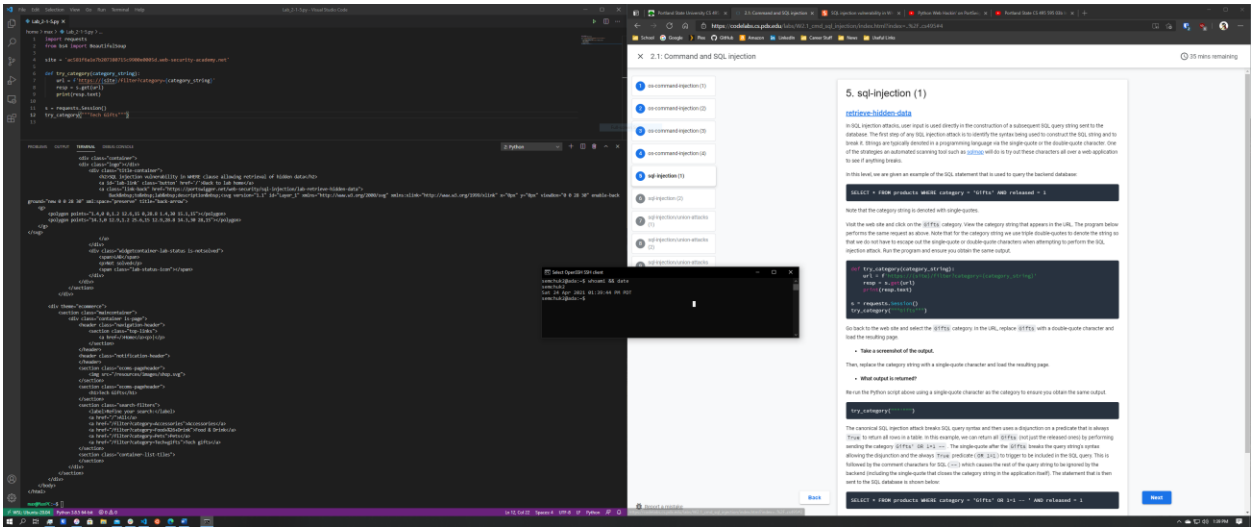
Labs 2.1.4 blind-out-of-band



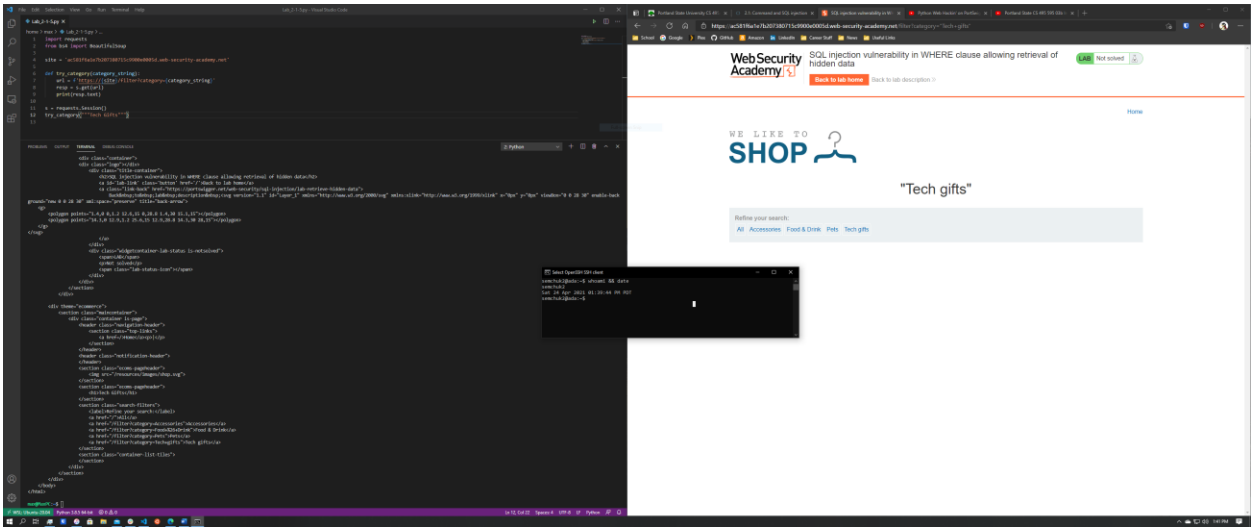
What this does, is redirect the traffic to another server, which is controlled by an adversary. The request allows to see the information flow, and then get it. This is a more complex attack and in our case is simplified but still demonstrates that if a redirect happens to another server, information can be intercepted.

Labs 2.1.5 retrieve-hidden-data

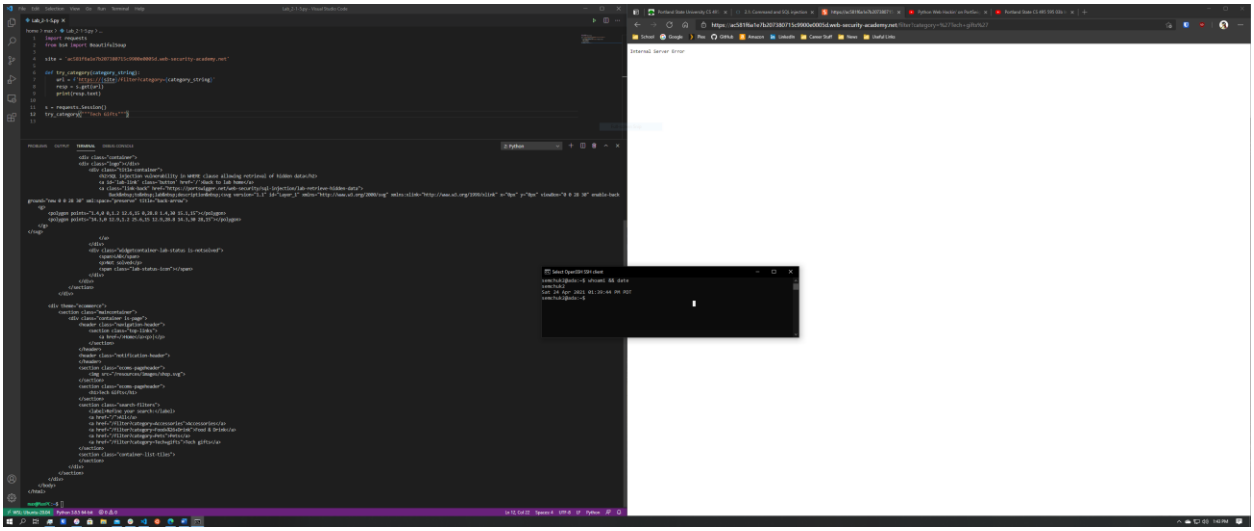
Here I ran the included code and got a good response.



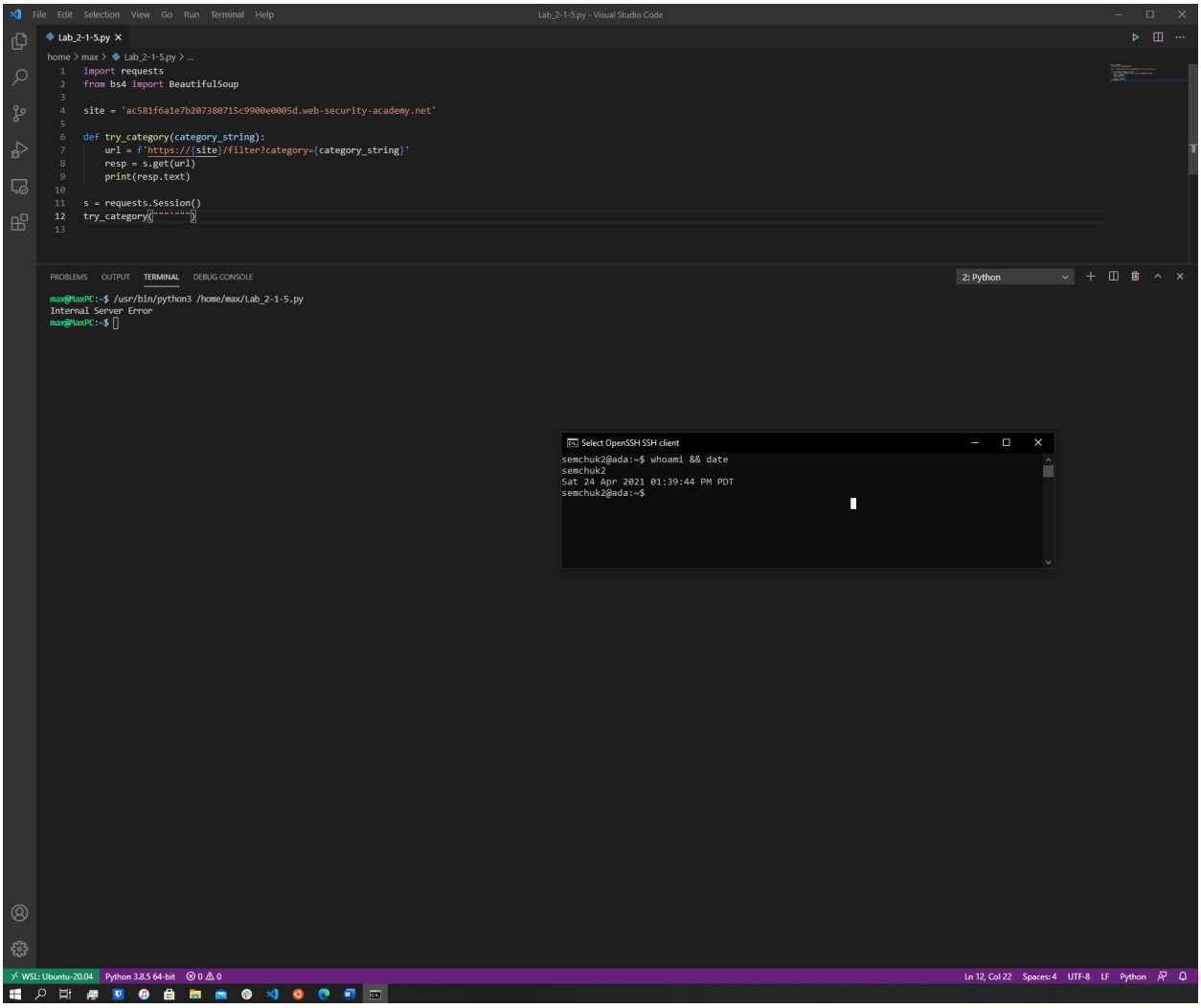
Here I added double quotes to the URL and got an empty page.



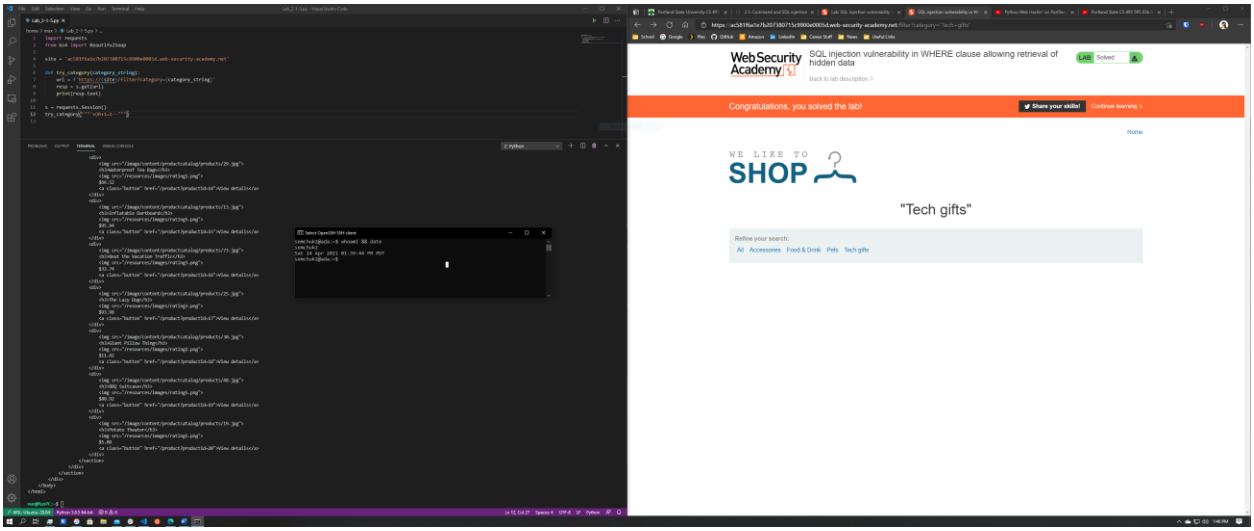
Then I added single quotes to the URL and got a Internal Server Error.



Then I ran the try_category("""""""") option of the script and got an internal server error.



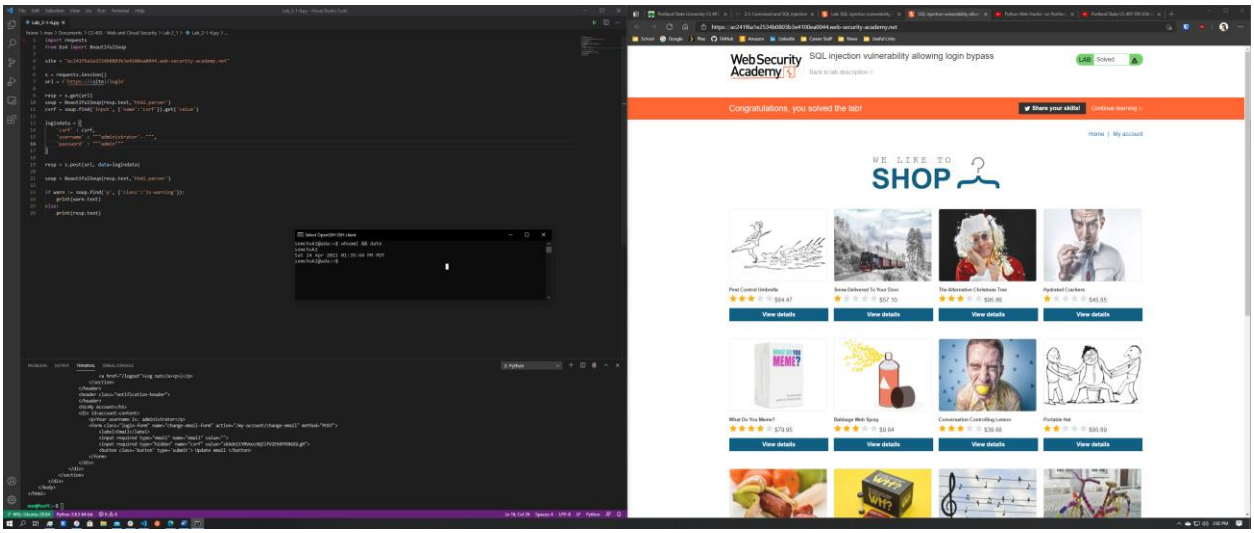
Finally, I was able to solve the level by providing: '+OR+1=1—into the category argument.



This worked because the server interacts with the sql database and if the server simply trusts all things that are passed in via URL to the query, this can open a vector of attack. This is important to prevent by only allowing certain standardized and whitelisted commands, as well as not returning errors in the returns.

Labs 2.1.6 sql-injection

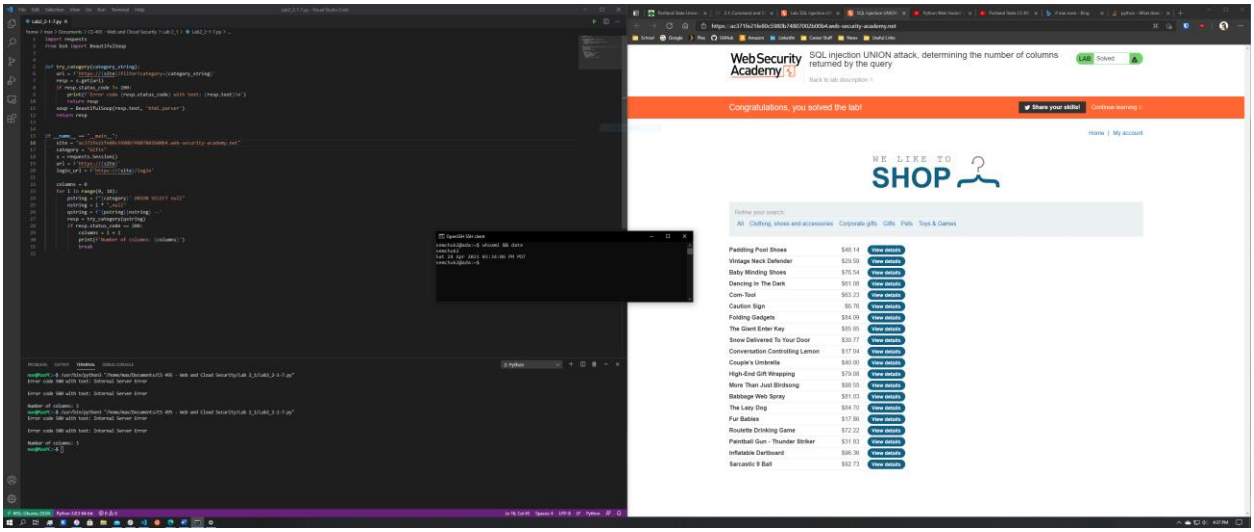
- Is the username field vulnerable to SQL injection? If so, what character breaks syntax?
    - Yes, it is vulnerable. What breaks it, is the '-- string at the end of the administrator.
- Is the password field vulnerable to SQL injection? If so, what character breaks syntax?
    - No, since the same thing cannot be used on the attack as above. The administrator username passed in does not get this to work.



The lab is solved by inputting administrator'-- into the username field.

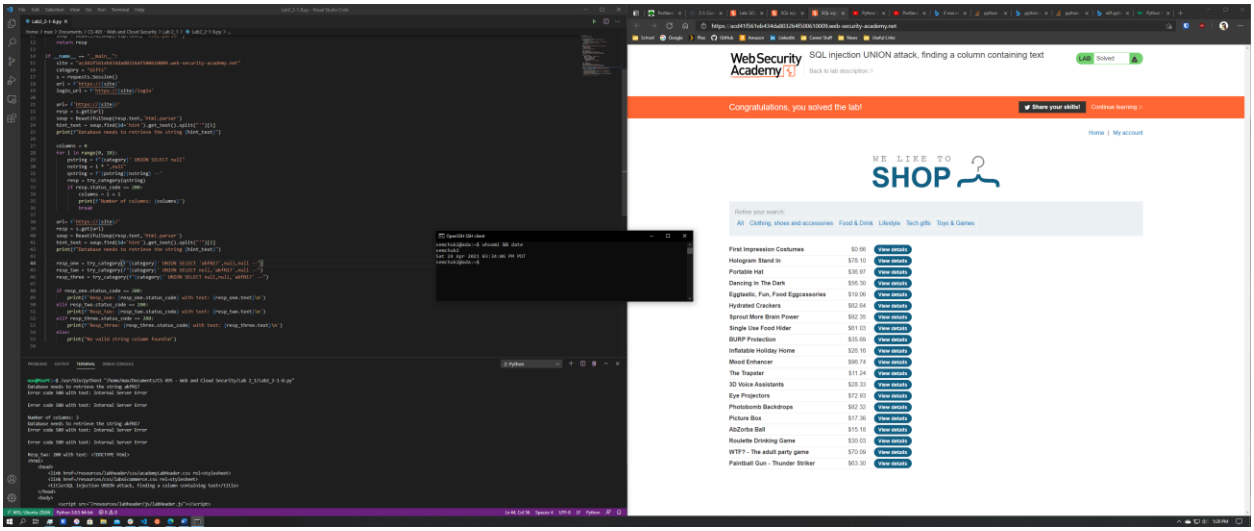
Labs 2.1.7 determine-number-of-columns

- How many columns does the products table contain?
  - 3 columns.



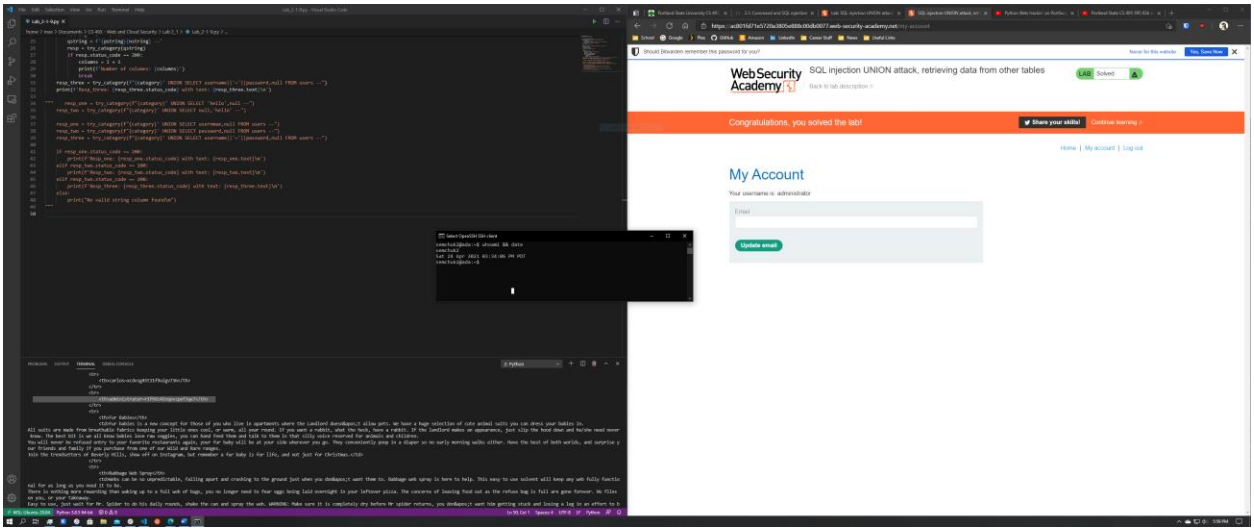
Here, the code looks for number of columns between 1 and 10 columns. The script exists when it finds a response that is a 200, and then prints the number of columns it found.

Labs 2.1.8 find-column-containing-text



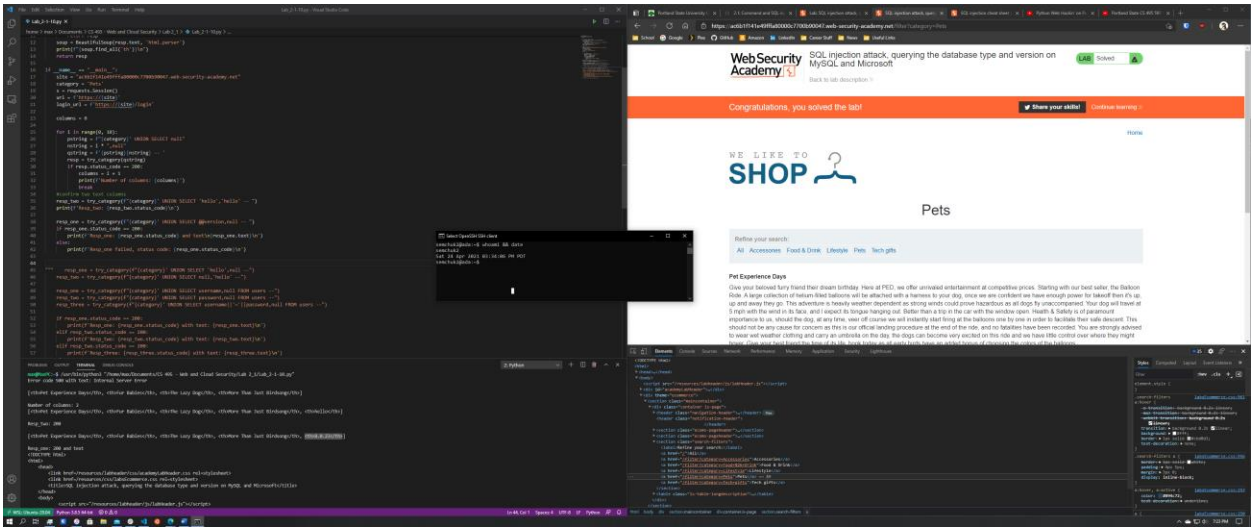
The second column is the one with the hint text.

Labs 2.1.9 retrieve-data-from-other-tables



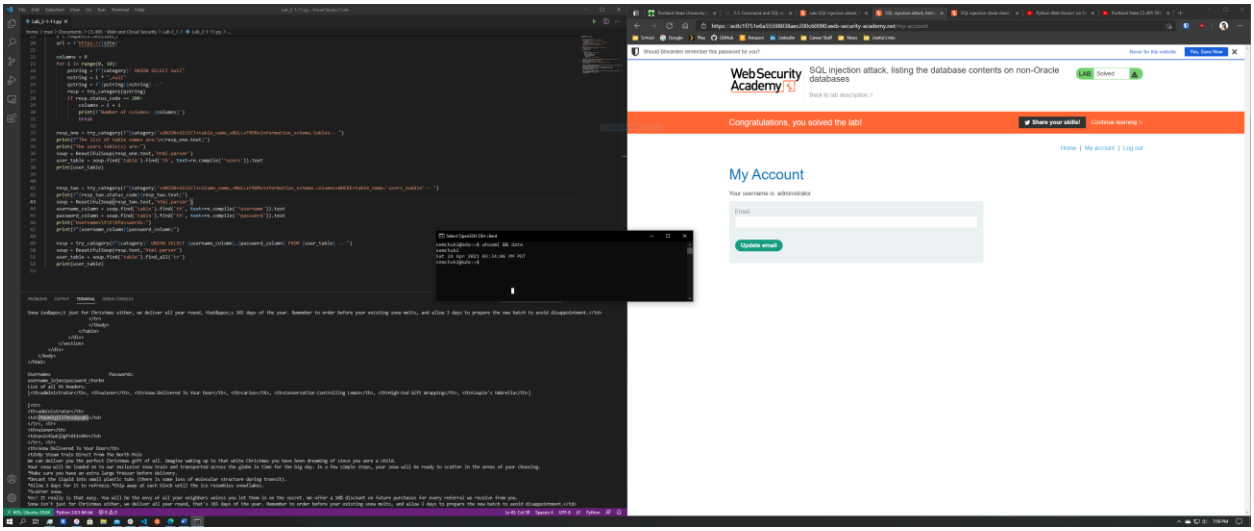
I was able to get the information using a UNION query which got the administrator password and showed it with the username.

Labs 2.1.10 querying-database-version-mysql-microsoft



I was able to get the version of the database.

Labs 2.1.11 listing-database-contents-non-oracle



Here I was able to get the database information for the username and password columns of the database. This basically shows that using the SQL/PSQL the attacker can get a list of usernames and passwords without knowledge of the database. This should never be allowed, the returning of database names should not be allowed by restricting how commands are parsed (that are passed into the command line).