SAIC SysAdmin Test 2023

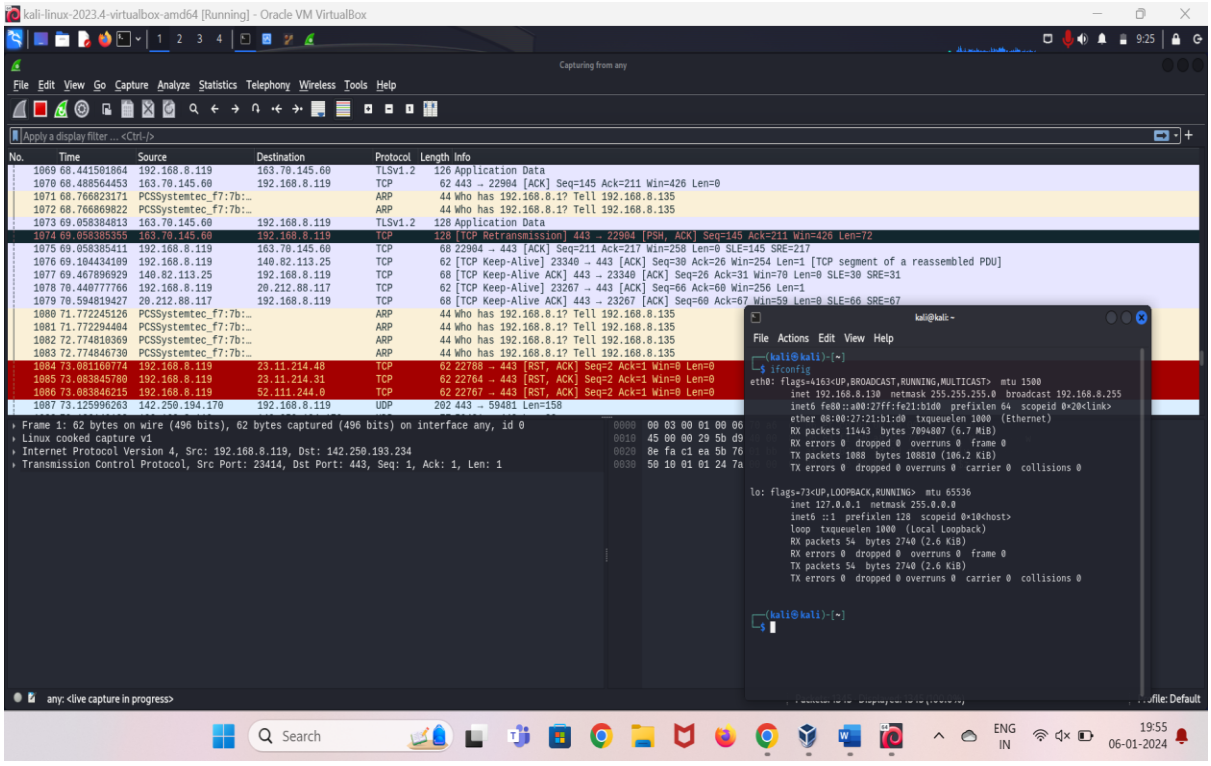
Name: Ayush Gaurav **Roll No:** B21183

Challenge 1: Gain Access to a Remote Server

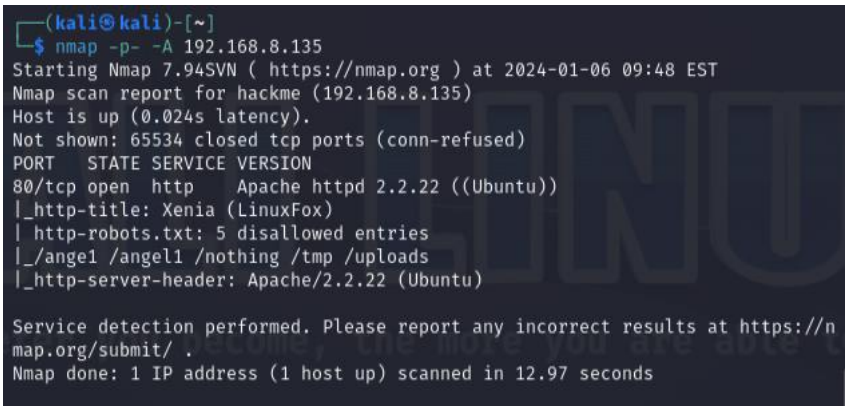
I started with importing the SAIC_fixed into my VirtualBox on Windows but realised that there was some issue with Network Interface. I already had Kali VM installed in my VirtualBox for one of college courses. Hence, I decided to install VirtualBox in my Kali VM. I faced many difficulties while setting up the SAIC_fixed VM inside my Kali VM, especially with VT-x not being available but finally it worked.

After loading the VM, I tried to guess some passwords for SAIC but was unsuccessful so, I logged into the VM as Guest. A few attempts after I also noticed that the SAIC VM was running Ubuntu 12.04 LTS so, I searched for vulnerabilities in Ubuntu 12.04 on the internet. Since our task is to gain root privileges, I found a vulnerability named sudo Vulnerability at https://ubuntu.com/security/notices/USN-4705-2 but couldn't discover how to exploit it.

As a guest, I was not allowed to use **ifconfig** on the SAIC VM but I wanted to know its IP address so, I turned on **Prominiscuous mode** in my virtualbox for my Kali VM to able to capture packets which were not addressed to my Kali VM using **Wireshark**. Since both VMs Kali as well as SAIC were running Bridged adapted mode, their IP address were assigned by the DNS server in my wifi. From settings of my wifi, I could see only one device connected with IP address 192.168.8.119 which was my Windows host. Using ifconfig in Kali VM, I came to know that it has IP address 192.168.8.130 so, I used wireshark in my Kali VM to analyze packets and found an IP address 192.168.8.135 which was the IP address of the SAIC VM.

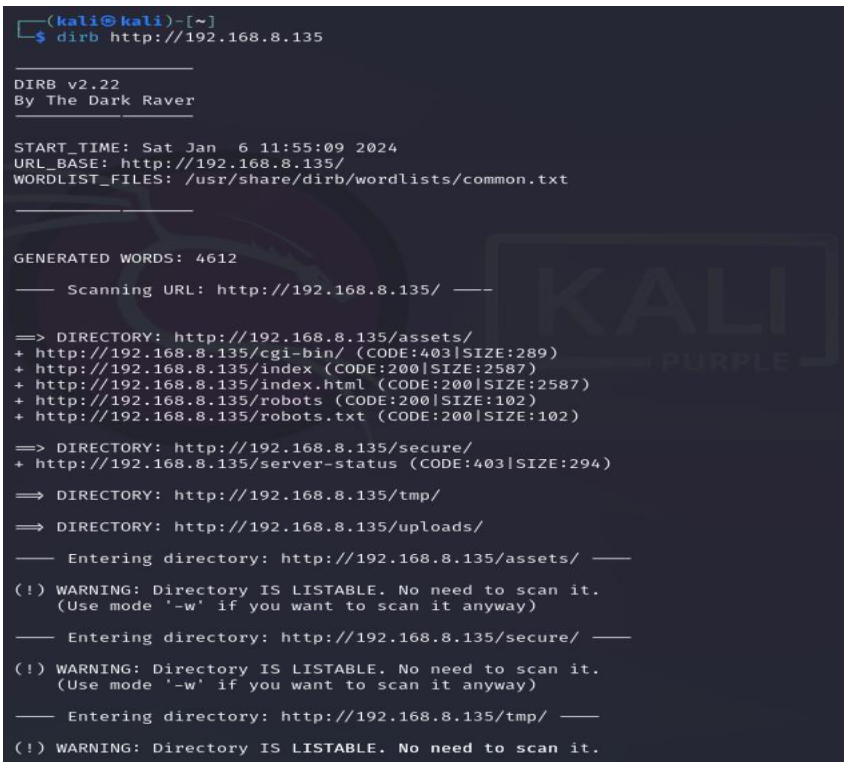


After discovering the IP address of SAIC VM, I used **nmap** to run an extensive scan of all the ports on the device. I discovered only one open port which was Port 80 running **http** service using **Apache** server with http-title as **Xenia (LinuxFox)** using **TCP** protocol as can be seen in the image below.



The nmap showed 5 disallowed entries. Using **curl**, I opened each of these 5 entries. They basically contained HTML code and some image but the source code of one of those files contained some **passwords in the form of a comment**, I kept a note of those as shown in the image below.

After this I performed **directory busting** using common wordlist packaged with “dirb” to find vulnerable directories and files as shown in image below but it didn’t help me much apart from finding a few additional directories as the info in them didn’t help me much.



After this I was only left to try with option of attacking the Apache server so, I searched the internet for common vulnerabilities in **apache httpd 2.2 22**. I found a useful resource at https://httpd.apache.org/security/vulnerabilities_22.html.

I tried many of the vulnerabilities listed there but was unable to find a way out there as the server had no login kind of functionality as per my exploration of the contents.

Then, I looked at the webpage the server was hosting. I saw that it was having many images, I wondered if these could have a vulnerability which could be exploited but was unable to find anything much relevant.

Challenge 2: Docker Monitoring & Scripting

I began with learning about basic questions such as Virtual Environment vs Container vs Virtual Machine and what is Docker and a Dockerfile. After installing **Docker Desktop**, I learnt how to write dockerfile by writing a very basic one and then, also wrote a very basic script and executed it in the Bash shell.

After that, I initially started with the idea of writing three web applications, one each in NodeJS, Python and Java but due to some issues in running Spring Boot and others, I decided to build two web applications, similar in nature both in NodeJS.

I wrote the two very simple web applications **node_app1.js** and **node_app2.js** as well as their corresponding Dockerfiles **Dockerfile_node1** and **Dockerfile_node2**.

After this I went on to write the script for detecting changes and for Email automation. I have combined both scripts into a single **powershell** script name **monitor_script.ps1** (I originally wrote a **Bash** but due to errors in the email part especially while using **mailx** in **WSL**, I finally decided to write a powershell script). The script has a **Send-Email** function which basically performs the Email automation.

Before running the script, the user need to fill in below details:

i. **Sender's Email** at Line No. 13
ii. **Receiver's Email** at Line No. 14
iii. **App Password** at Line No. 32

Now, to perform the desired task we need to first build Dockerimages from the Dockerfiles and finally containers, run the script for Email automation and detecting changes and run the containers as shown in the images below:

I. Building the Containers

```
PS C:\Users\Dell\Desktop> docker build -t node_app1 -f Dockerfile_node1 .
[+] Building 1.4s (7/7) FINISHED                                                                                    docker:default
 => [internal] load .dockerignore                                                                                           0.0s
 => => transferring context: 2B                                                                                             0.0s
 => [internal] load build definition from Dockerfile_node1                                                                  0.0s
 => => transferring dockerfile: 125B                                                                                        0.0s
 => [internal] load metadata for docker.io/library/node:14                                                                  1.2s
 => [internal] load build context                                                                                           0.0s
 => => transferring context: 34B                                                                                            0.0s
 => [1/2] FROM docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa            0.0s
 => CACHED [2/2] COPY node_app1.js /node_app1.js                                                                            0.0s
 => exporting to image                                                                                                      0.0s
 => => exporting layers                                                                                                     0.0s
 => => writing image sha256:b9d450db881c4302d9f6ba1f869ed8001ddf8d2cd6ade17d10d4de9fe74dc718                                0.0s
 => => naming to docker.io/library/node_app1                                                                                0.0s

What's Next?
  1. Sign in to your Docker account → docker login
  2. View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\Dell\Desktop> docker save -o node_app1.tar node_app1
PS C:\Users\Dell\Desktop> docker load -i node_app1.tar
Loaded image: node_app1:latest
```

Steps for building container for the second file are similar (replace node_app1 with node_app2 and Dockerfile_node1 with Dockerfile_node2 everywhere in commands).

II. Running the Email Automation and Main Script to detect changes:

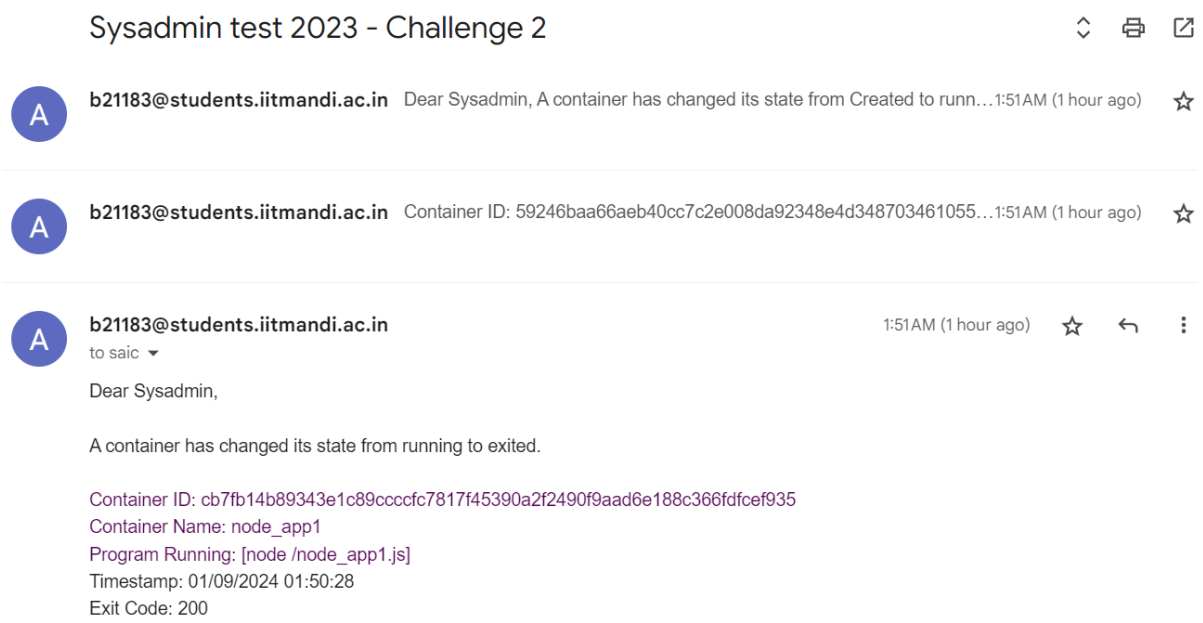
```
PS C:\Users\Dell\Desktop> .\monitor_script.ps1
```

III. Running the Docker Containers:

```
PS C:\Users\Dell\Desktop> docker run -d -p 3500:3500 --name node_app1 node_app1
42ba648a447c1ec370de5ecc04f1043a95864848828f9cf2af388b544a52a849
PS C:\Users\Dell\Desktop> docker run -d -p 3000:3000 --name node_app2 node_app2
d55bc5cc5996b9e26d953306810aec83450410c3a2d9566bc2bede4517c88d73
```

The running containers for node_app1 and node_app2 will exit after 45 sec and 60 sec respectively with error codes of 200 and 100 respectively.

Result:



Challenge 3: Docker Deployments

I started with **cloning** the GitHub repositories of SAIC IIT Mandi and GitHub languages. After this, I created **Dockerfile** in the root of both the repositories. Next, I wrote the Docker compose script **docker-compose.yml**.

The mappings used are:

- SAIC IIT Mandi: Host port 8081 to container port 80
- GitHub Languages: Host port 8082 to container port 3000

Based on the knowledge which I gained about Network types for containers, the suitable network types for each container are:

- **Bridge Network for SAIC IIT Mandi:** Since as far as I could see the website, it does not need to communicate with services outside the docker environment i.e. it does not seem to have external dependencies. Hence, I suggest to use **Bridge Network** type for it.
- **Custom Bridge Network for GitHub Languages:** Since using a Custom Bridge Network provides isolation and more control over network configuration, especially for a container running a web application with multiple services such as HTML-CSS-JS and Ruby on Rails and also considering the fact that Ruby on Rails application brings in additional complexity associated with it and we may want the containers to communicate with each other on a custom network. This approach may provide a dedicated network for the interconnected services within the GitHub Languages container, although I was unable to use this network type due to various errors.

After completing docker-compose.yml, I wrote **backup.ps1** powershell script for backup and also wrote **schedule.ps1** powershell script to schedule the backup script.

After this, I ran commands:

docker-compose build to build images and

docker-compose up -d to create and start the containers from the images built.

Result:

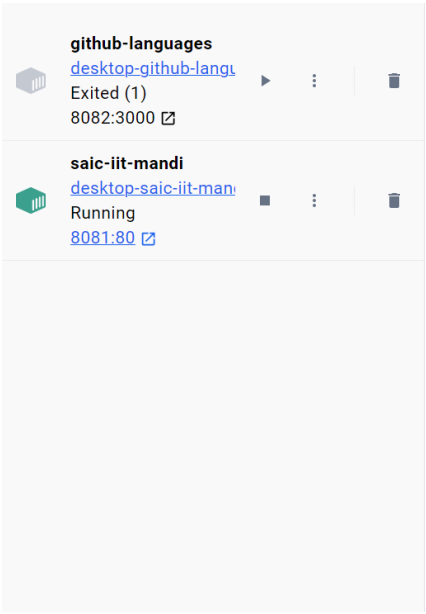
The **docker-compose build** command failed many times, after some time I manages to build image for SAIC IIT Mandi website but it was challenging to do for the GitHub languages website as it used Ruby on Rails which required many dependencies. Many times the commands failed but after making changes in the **Gemfile** and **Gemfile.lock**, especially with sometimes missing dependencies and sometimes incompatible versions, I was finally able to build a docker image for both the websites. On running the containers, I was able to **successfully host SAIC IIT Mandi** on my system but there was still some issue with the container of GitHub Languages and it exited after being started with some errors. I fixed many of such issues mentioned therein such as incompatible versions and required dependencies using Gemfile and

Gemfile.lock by making changes in them and rebuilt the images and ran the container many times but once an issue was fixed another came up and it was difficult to deal with.

Running the scripts for backup and scheduling:

```
PS C:\Users\Dell\Desktop> .\backup.ps1
Backup completed for SAIC IIT Mandi: C:\Users\Dell\Desktop\Backup\saic-iit-mandi-20240111.zip
Backup completed for GitHub Languages: C:\Users\Dell\Desktop\Backup\github-languages-20240111.zip
PS C:\Users\Dell\Desktop> .\schedule.ps1
Scheduled task created to run the backup script at 23:55 daily.
```

SAIC IIT Mandi successfully running and issues with GitHub Languages:




```
2024-01-11 18:44:29 saic-iit-mandi    | 2024/01/11 13:14:29 [notice] 1#1: start work
er process 33
2024-01-11 18:44:29 saic-iit-mandi    | 2024/01/11 13:14:29 [notice] 1#1: start work
er process 34
2024-01-11 18:44:29 saic-iit-mandi    | 2024/01/11 13:14:29 [notice] 1#1: start work
er process 35
2024-01-11 18:44:29 saic-iit-mandi    | 2024/01/11 13:14:29 [notice] 1#1: start work
er process 36
2024-01-11 18:44:29 saic-iit-mandi    | 2024/01/11 13:14:29 [notice] 1#1: start work
er process 37
2024-01-11 18:44:29 github-languages  | /usr/local/bin/ruby: warning: shebang line e
nding with \r may cause problems
2024-01-11 18:44:29 github-languages  | /usr/local/lib/ruby/2.6.0/bundler/resolver.r
b:287:in `block in verify_gemfile_dependencies_are_found!': You have requested: (Bun
dler::GemNotFound)
2024-01-11 18:44:29 github-languages  |   mini_portile2 >= 2.4.0
2024-01-11 18:44:29 github-languages  |
2024-01-11 18:44:29 github-languages  | The bundle currently has mini_portile2 locke
d at 2.1.0.
2024-01-11 18:44:29 github-languages  | Try running `bundle update mini_portile2`
2024-01-11 18:44:29 github-languages  |
2024-01-11 18:44:29 github-languages  | If you are updating multiple gems in your Ge
mfile at once,
github-languages exited with code 1
github-languages  | 2024-01-11T13:14:29.898635053Z try passing them all to `bundle u
pdate`
```

Challenge 4: Docker & Networking

I started by cloning the **metavinayak/matrix-custom** from Docker Hub.

```
PS C:\Users\Dell\Desktop\4> docker pull metavinayak/matrix-custom
Using default tag: latest
latest: Pulling from metavinayak/matrix-custom
```

After this, I wrote a very simple **Dockerfile**. Then, I wrote **Docker-compose.yml** with three instances. After this, I wrote **nginx.conf** which is the main configuration file of the Nginx web server which performs load balancing.

After this, I started the containers using:

```
PS C:\Users\Dell\Desktop\4> docker-compose up -d
[+] Running 5/5
 ✓ Network 4_default                  Created
 ✓ Container matrix-custom-container3  Started
 ✓ Container matrix-custom-container1  Started
 ✓ Container matrix-custom-container2  Started
 ✓ Container load-balancer             Started
PS C:\Users\Dell\Desktop\4> Invoke-WebRequest -Uri http://localhost


StatusCode        : 200
```

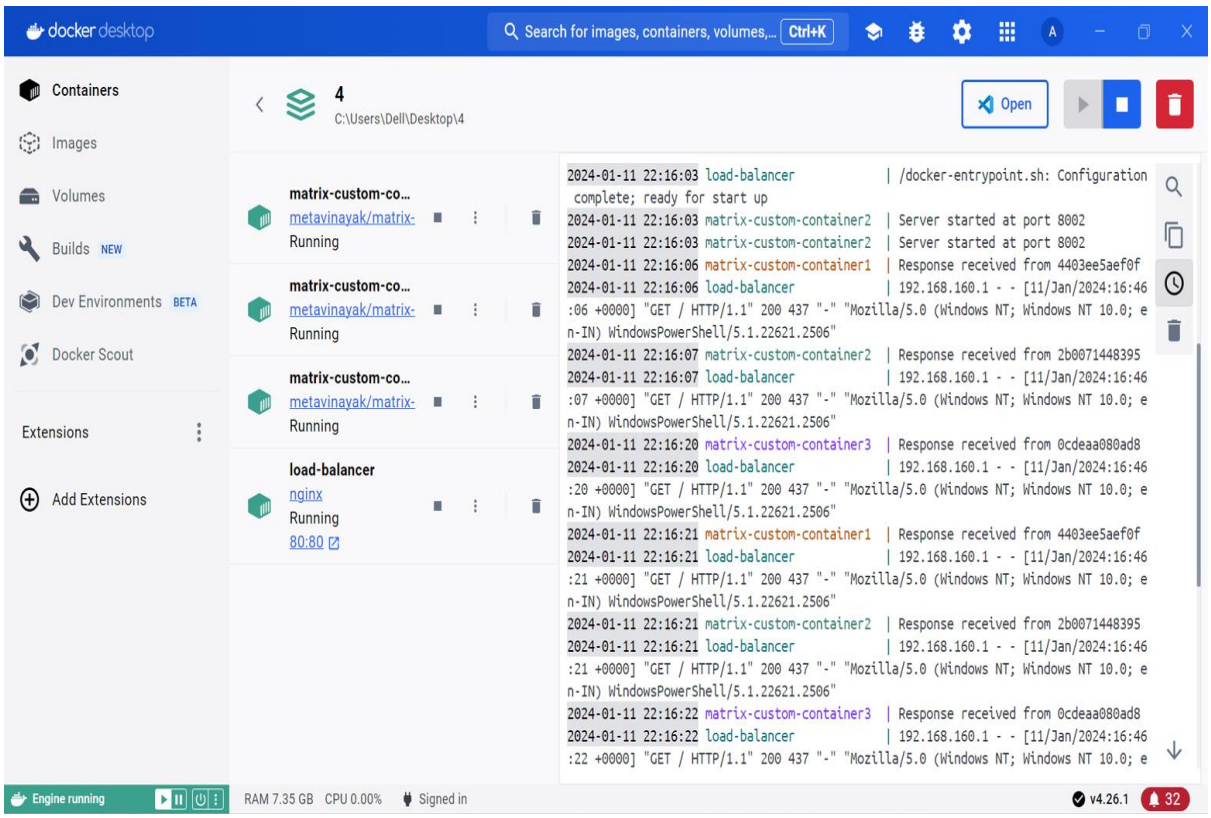
After which I sent multiple requests using various tools such as **Invoke-WebRequest** in Powershell and **Curl** in WSL as shown below:

```
ubuntuwsl@DESKTOP-PTCM1UP:/mnt/c/Users/Dell/Desktop/4$ curl http://localhost
<body style=margin:0><canvas id=q /><script>var q=document.getElementById('q'),s=window.screen,w=q.width=s.width,h=q.height
=s.height,p=Array(256).join(1).split(''),c=q.getContext("2d"),m=Math;setInterval(function(){c.fillStyle="rgba(0,0,0,0.05)";
c.fillRect(0,0,w,h);c.fillStyle="rgba(0,255,0,1)";p=p.map(function(v,i){r=m.random();c.fillText(String.fromCharCode(m.floor
(2720+r*33)),i*10,v);v+=10; return v>768+r*1e4?0:v})},33)</script>
```

Then, I ran **docker logs -f load-balancer** but It didn't seemed to give me any information about Load Balancing as shown below:

```
PS C:\Users\Dell\Desktop\4> docker logs -f load-balancer
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
192.168.96.1 - - [11/Jan/2024:16:29:04 +0000] "GET / HTTP/1.1" 200 437 "-" "Mozilla/5.0 (Windows NT; Windows NT 10.0; en-IN
) WindowsPowerShell/5.1.22621.2506"
192.168.96.1 - - [11/Jan/2024:16:29:05 +0000] "GET / HTTP/1.1" 200 437 "-" "Mozilla/5.0 (Windows NT; Windows NT 10.0; en-IN
) WindowsPowerShell/5.1.22621.2506"
192.168.96.1 - - [11/Jan/2024:16:29:06 +0000] "GET / HTTP/1.1" 200 437 "-" "Mozilla/5.0 (Windows NT; Windows NT 10.0; en-IN
) WindowsPowerShell/5.1.22621.2506"
192.168.96.1 - - [11/Jan/2024:16:29:07 +0000] "GET / HTTP/1.1" 200 437 "-" "Mozilla/5.0 (Windows NT; Windows NT 10.0; en-IN
) WindowsPowerShell/5.1.22621.2506"
192.168.96.1 - - [11/Jan/2024:16:29:07 +0000] "GET / HTTP/1.1" 200 437 "-" "Mozilla/5.0 (Windows NT; Windows NT 10.0; en-IN
) WindowsPowerShell/5.1.22621.2506"
192.168.96.1 - - [11/Jan/2024:16:29:12 +0000] "GET / HTTP/1.1" 200 437 "-" "curl/7.81.0"
192.168.96.1 - - [11/Jan/2024:16:29:13 +0000] "GET / HTTP/1.1" 200 437 "-" "curl/7.81.0"
192.168.96.1 - - [11/Jan/2024:16:29:14 +0000] "GET / HTTP/1.1" 200 437 "-" "curl/7.81.0"
192.168.96.1 - - [11/Jan/2024:16:29:14 +0000] "GET / HTTP/1.1" 200 437 "-" "curl/7.81.0"
192.168.96.1 - - [11/Jan/2024:16:29:15 +0000] "GET / HTTP/1.1" 200 437 "-" "curl/7.81.0"
```

Thus, I went to logs in Docker Desktop application and there I was able to see Load Balancing in the logs as shown below:



After this, I also tried to generalize it for n-instances. I wrote nginx.conf.template and made changes in my Docker-compose.yml but for that I needed to specify the path to powershell.exe in my system which I specified but gave me errors while executing. I tried to fix it but it did not work and I realized that time is up (see the timing in the logs 😊) so, I have put those two files outside the directory of Challenge 4 on GitHub.

Challenge 5: Scripting

As I had already written an Email automation script in **Powershell** in Challenge 2, I initially decided to write a Powershell script but due too many errors while configuring **Chrome Driver, Selenium and Chromiun**, after spending six hours resolving it, I realized that writing a **Python** script would be an easy option.

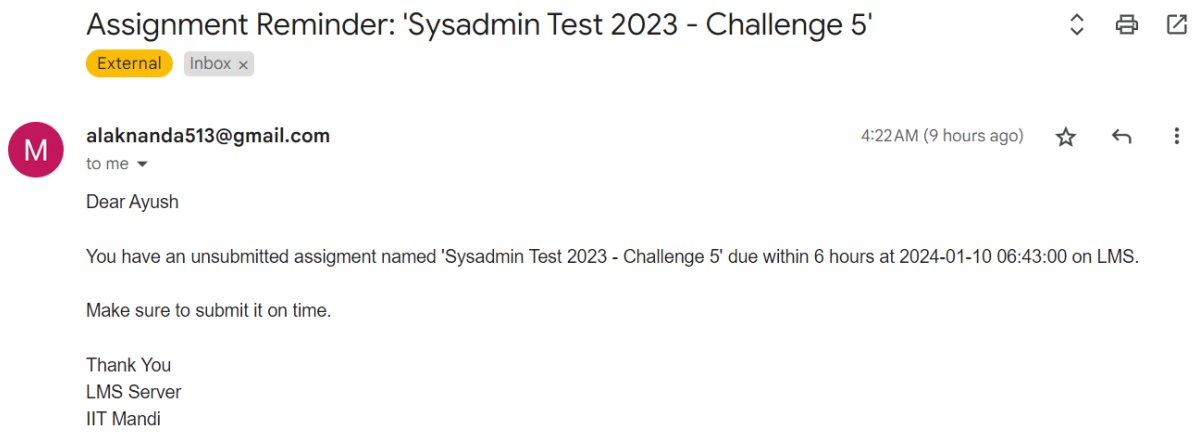
I wondered how the process for checking unsubmitted assignments may be automated, for this explored LMS many times with different perspectives. During this, I came across a functionality to **Export Calendar** as a **.ics** file which contains info all the info about the events for a user.

Thus, in my Python script, I used **Selenium** to automate the process of browsing through LMS and exporting the calendar as a .ics file and downloading it. After this, in my script I read the **most recent** .ics file, parsed it and retrieved information about all events. I stored the info about all events in the form of a **List of Objects** and then further used to find events due within the next six hours. Based on this, if an even was due within the next six hours, I used an App password to send an Email to the recipient or printed the details of the event if it was already due.

Steps to use the Script (assignment_check.py):

1. **Customize the script according to the user at following places:**
    i. **Email Credentials:**
        a. Sender's Email at Line No. 44
        b. Recipient's Email at Line No. 45
        c. Recipient's Name at Line No. 46
        d. App Password for Sender's Gmail Account at Line No. 57
    ii. **LMS Credentials:**
        a. Username at Line No. 61
        b. Password at Line No. 62
    iii. **Chrome Driver and Chrome executable paths:**
        a. Chrome Driver Path at Line No. 65
        b. Chrome Binary Path at Line No. 68
2. **Run the script:** Use the command below.
    python -u "Path/to/the/script"

Result:



Comments:

Apart from exporting the calendar, another approach could have been to click on the links of all courses available on the page, navigate through them in a similar way as done in the script and check whether the assignment is unsubmitted and has deadline either passed or due within the next six hours and add it to the list of events (which in the above case is obtained after parsing through the .ics file). This approach would also have checked whether the assignment has already been submitted (which the above script does not). I decide to go with previous approach due to lack of an assignment actually pending within the next six hours on LMS and as we can't create such as assignment ourselves, it would have been difficult to test the correctness of the script. Hence, for the purpose of this challenge I decided to go with the approach of downloading the .ics file and parsing it to obtain the event.

Also, the language of the challenge seemed to be a bit changed from the time I solved it to the final submission and as I had already solved the challenge much earlier (it being the third challenge which I tried) it was difficult for me to make changes at the last moment.

THANK YOU!!