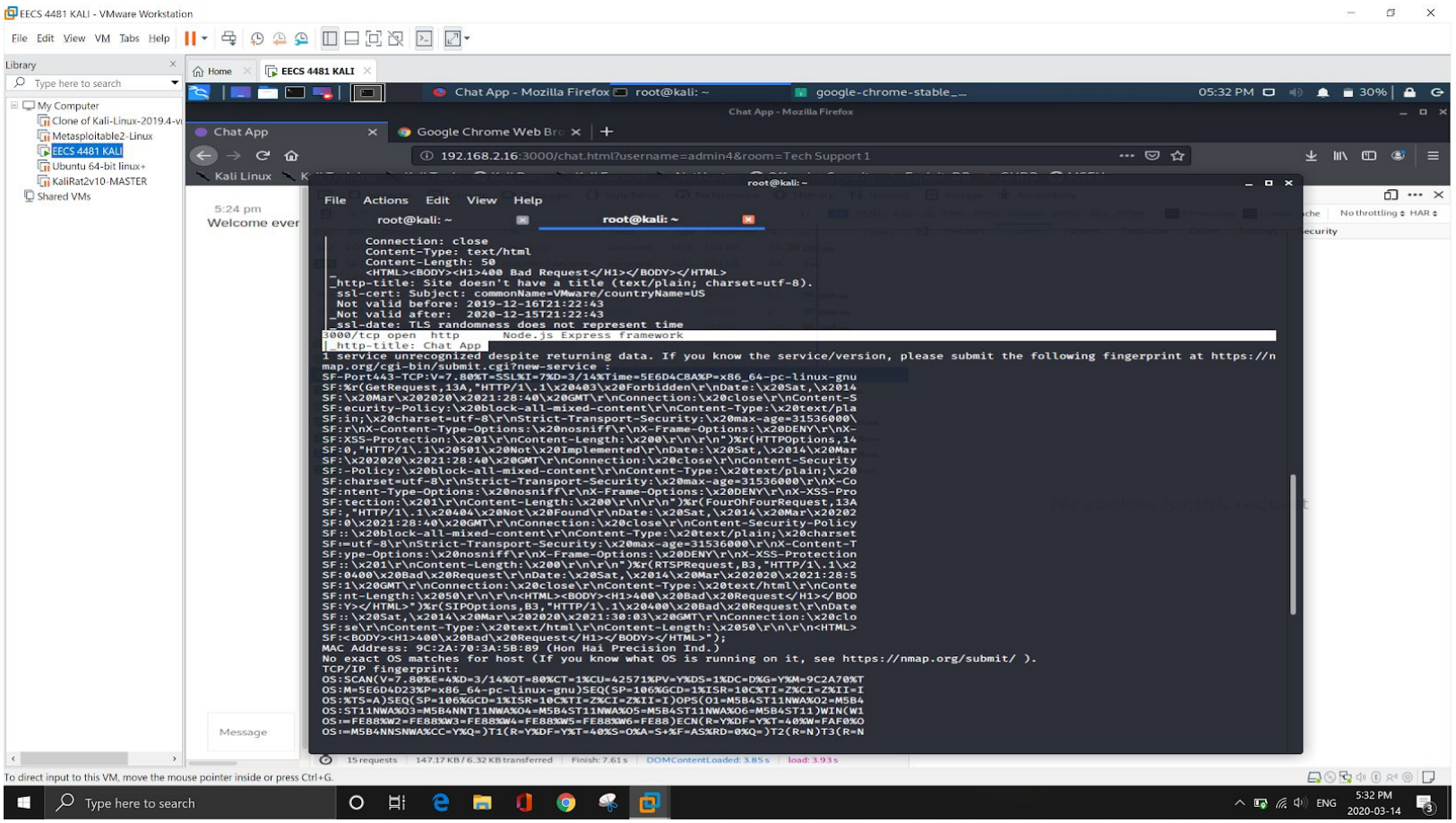
Phase 2 Penetration Testing Report

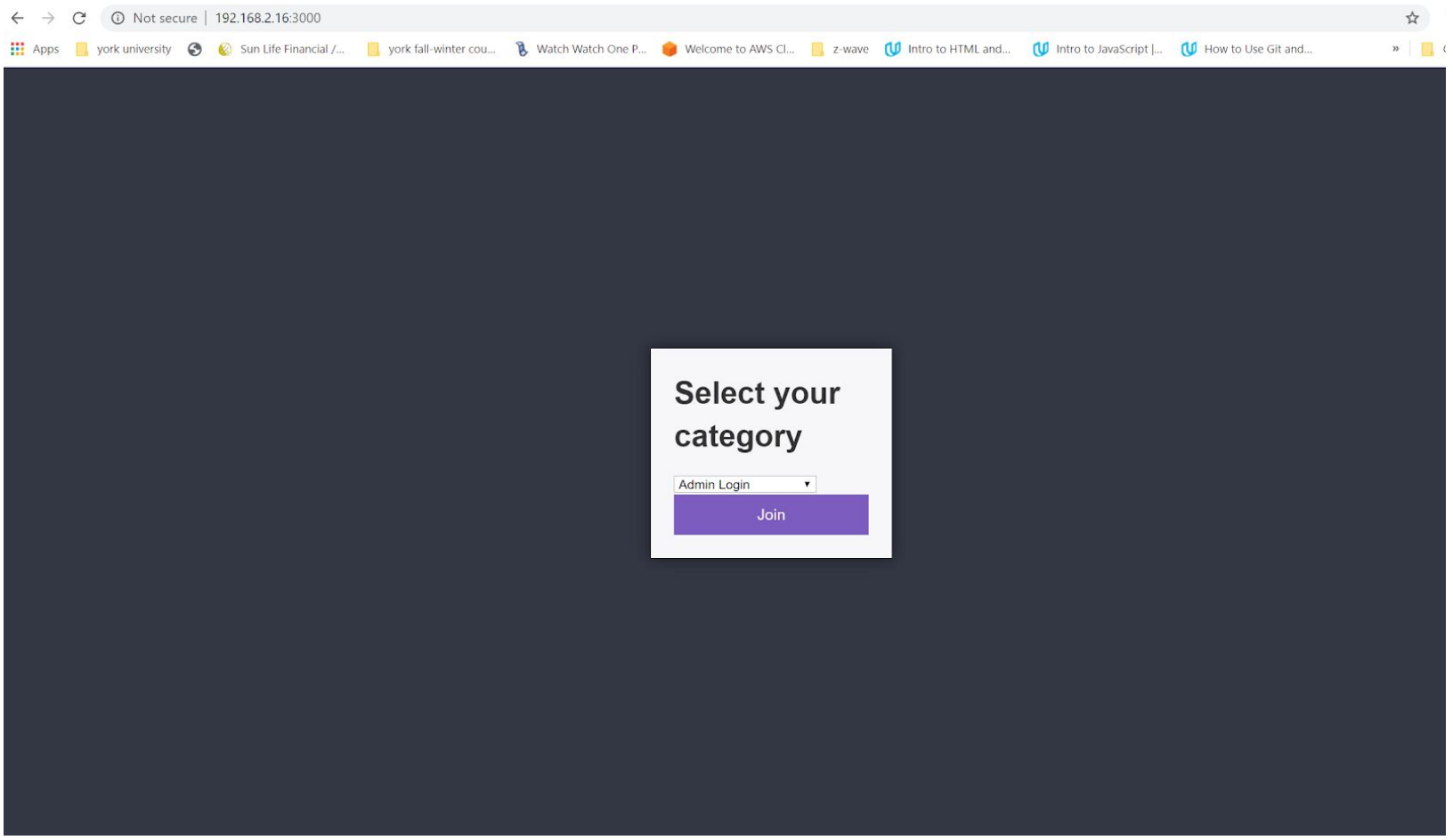
1) The Penetration Test began with nmap to the machine



The command nmap -A 192.168.2.16 provides some major conclusions:

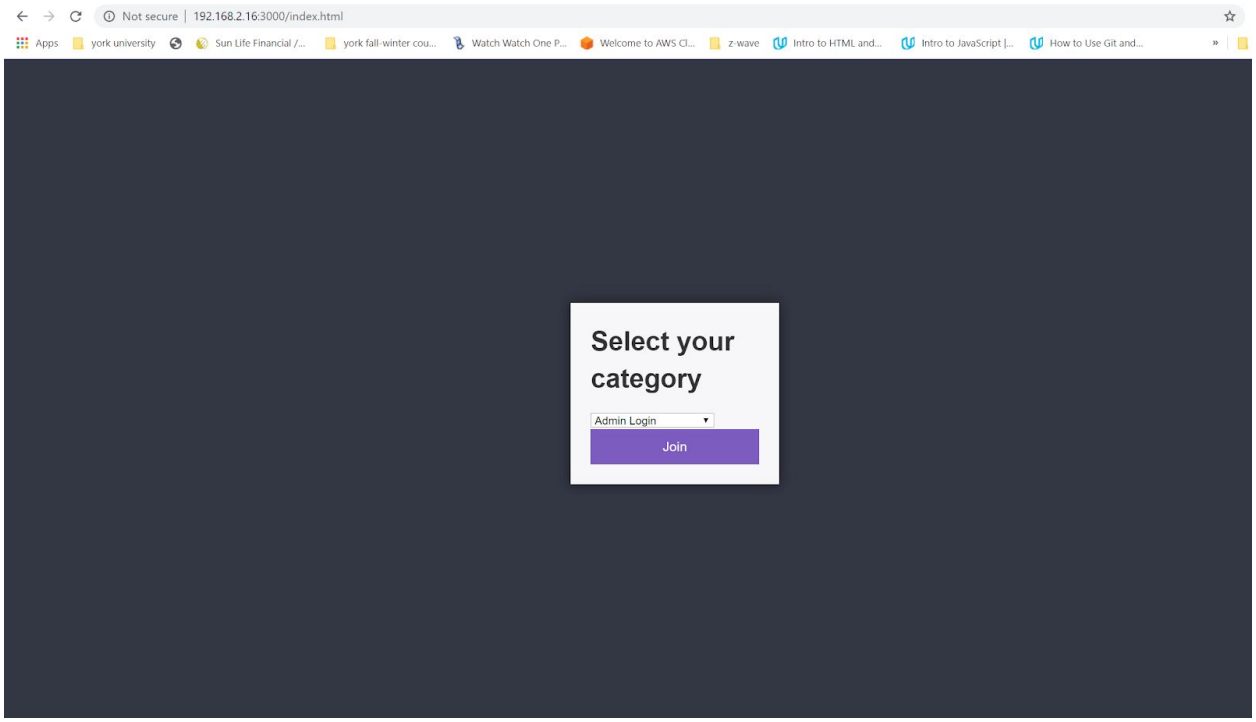
1) The web application is running on port 3000
2) The application has the Node.js Express framework. This step is beneficial for future attacks and exploits with metasploit etc.
3) The OS is ubuntu.

2) As with accessing the webpage, we use the IP and port 3000 and are presented with the following webpage of the application.



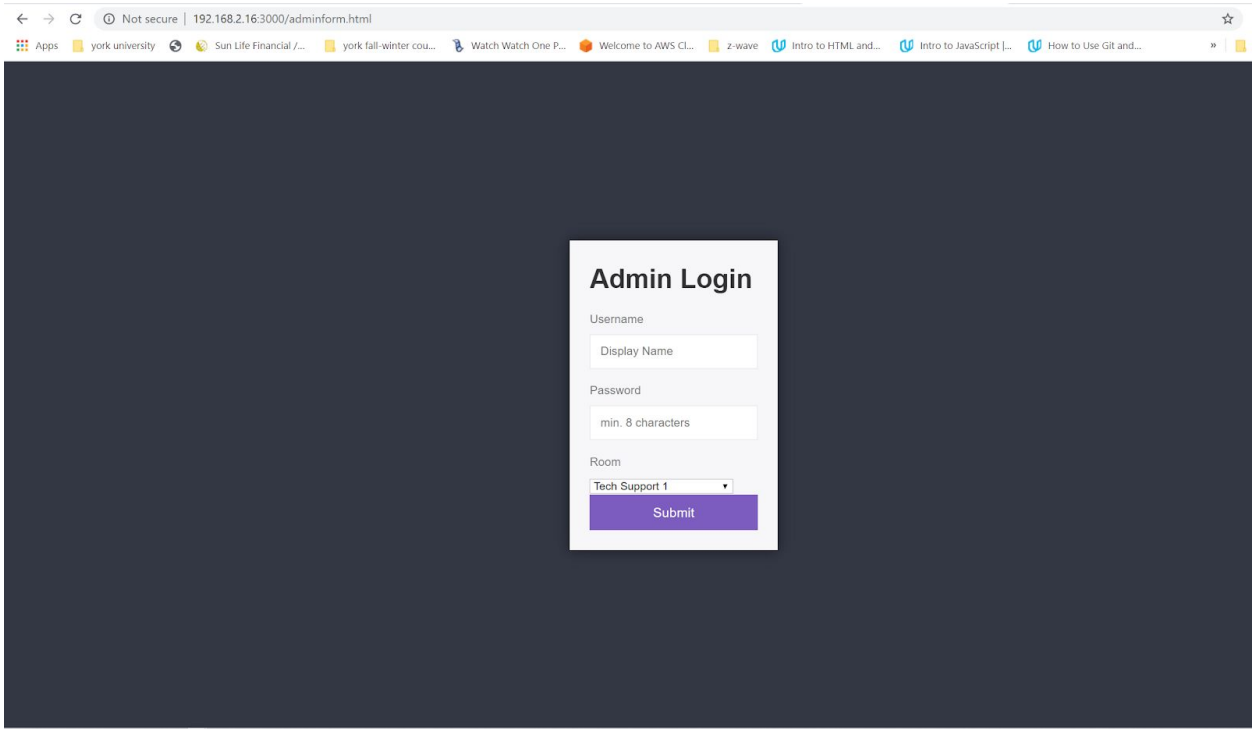
3) **using dirbuster**, several directories as well as files were revealed which could be used further.

A couple of them were index.html and adminform.html .



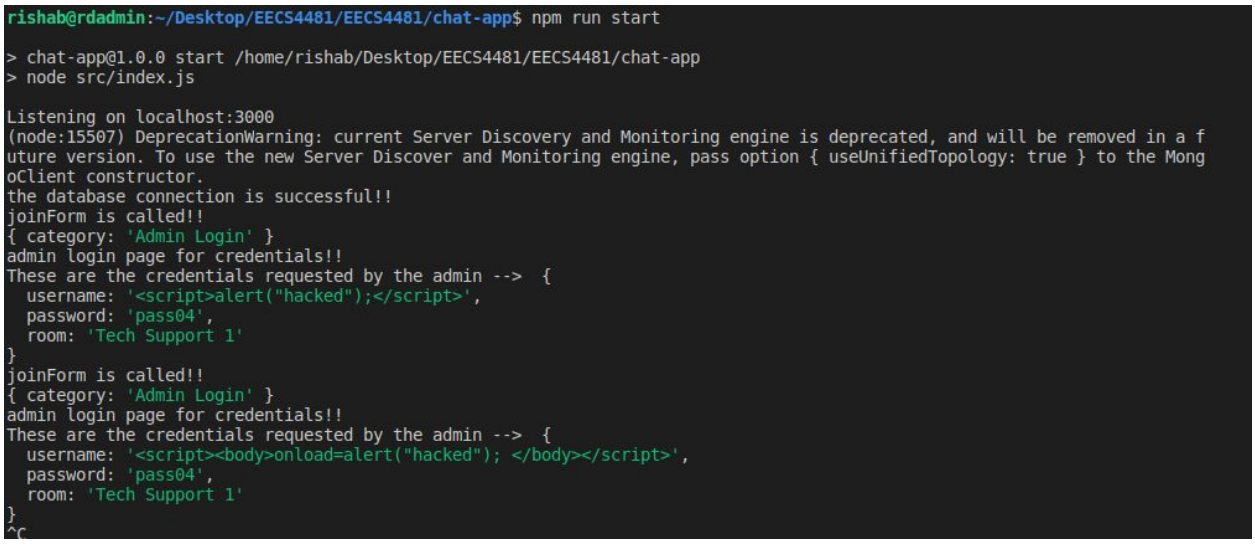
Check the URL, it will state index.html.

On changing the URI to adminform.html, we get the following webpage



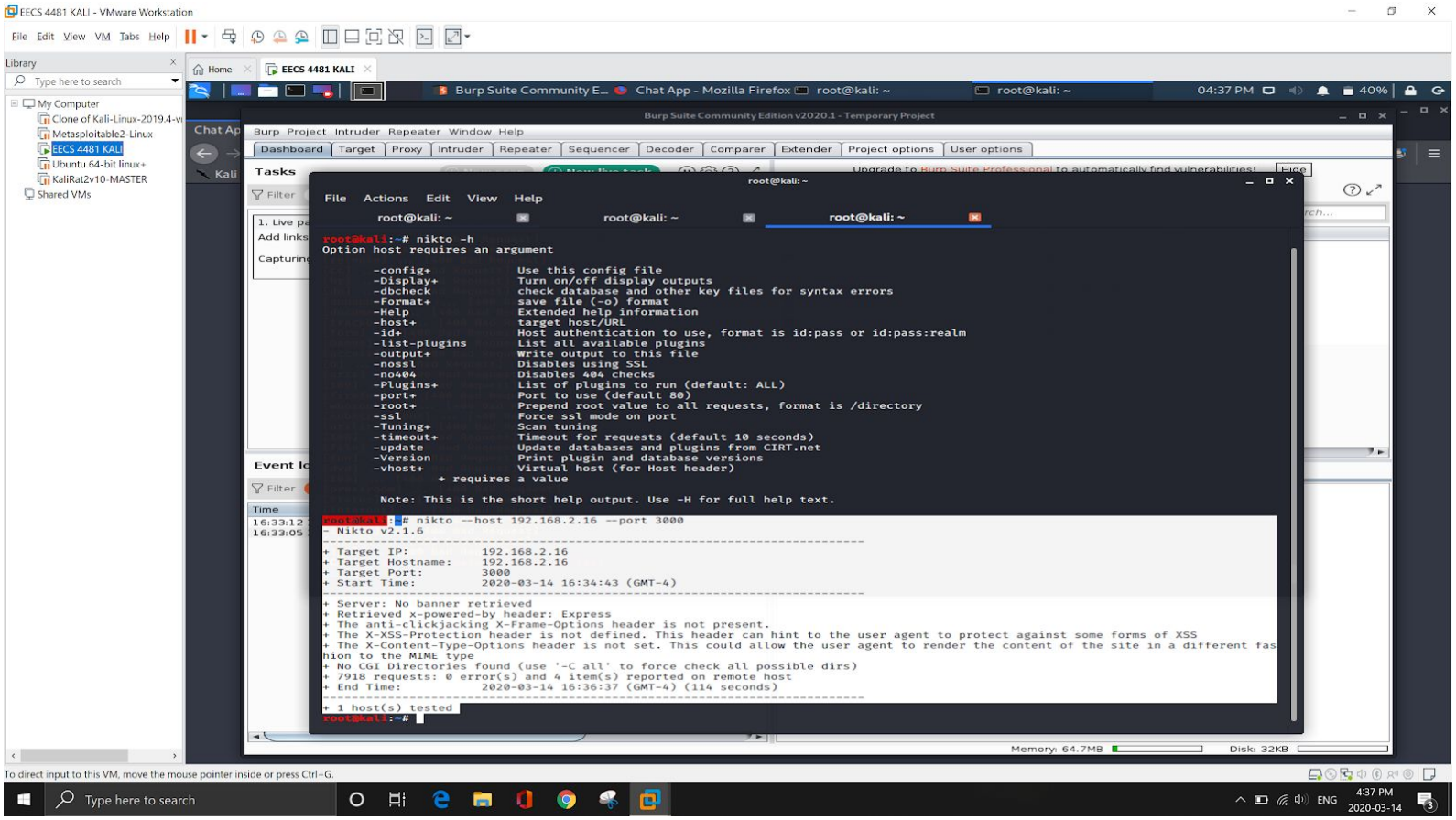
: This page prompts for credentials, therefore it will prove a vital attack point.

4) Performing **XSS attacks** on this page with basic script tags for testing purposes didn't work as the response wasn't returned back from the server for invalid username or invlaid password. On the other side, validations on the server as well as authentication phase of server checking proper credentials worked. Every Single <script> tag attacker tried to inject into the form was logged into the backend server.



This is a log from the back-end node js server. The green colored statements present the <script> commands tried by the attacker.

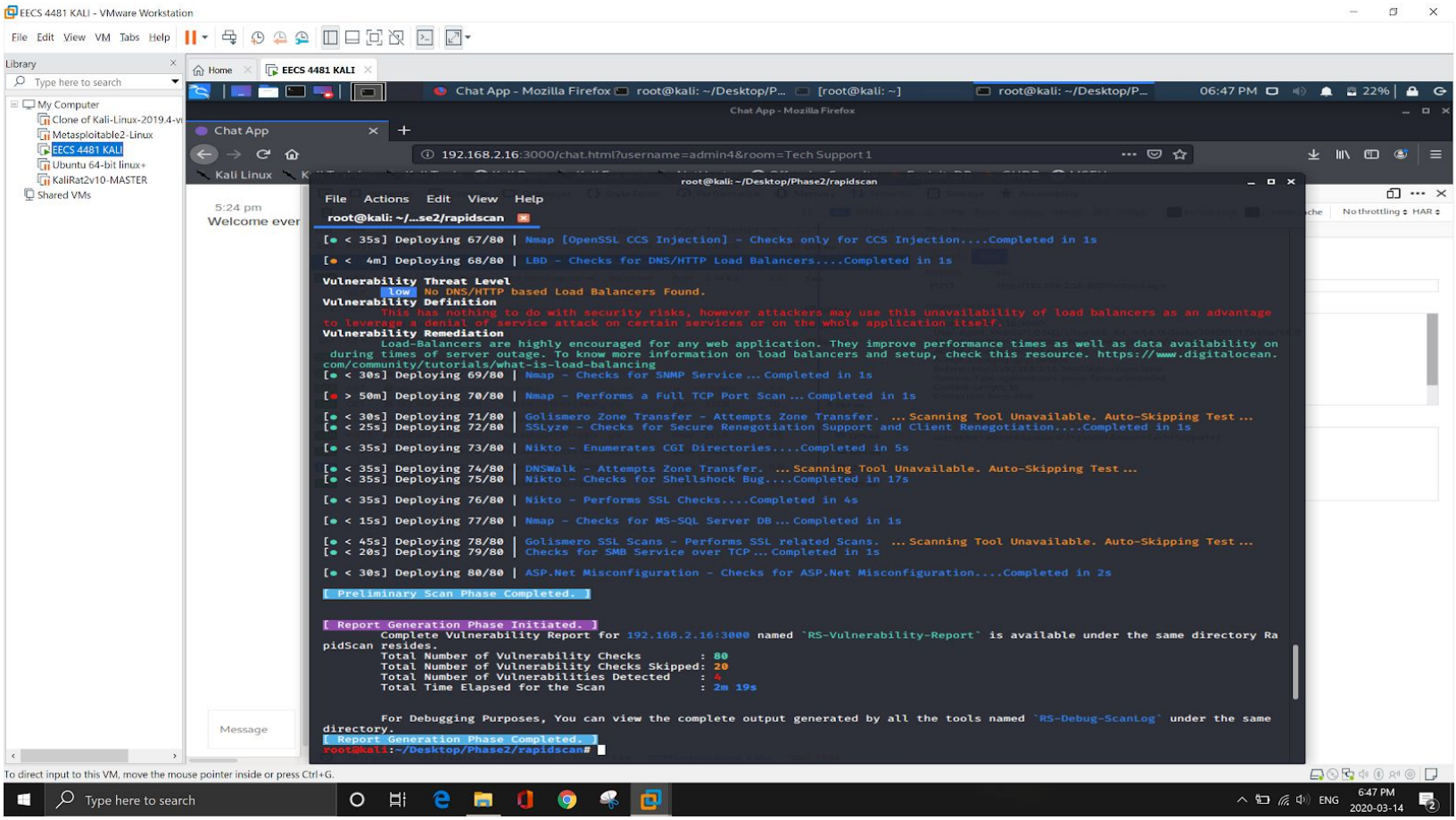
5) **Nikto**



These are the scanned results from the nikto on the website. They provide warnings for XSS header as well as anti-clickjacking X-Frame Option. To solidify the XSS vulnerability, I moved to open vulnerability scanners next.

6) **First open vulnerability scanner is Arachni**. Below are major results drawn from the scans of the website through this scanner:

6.1)



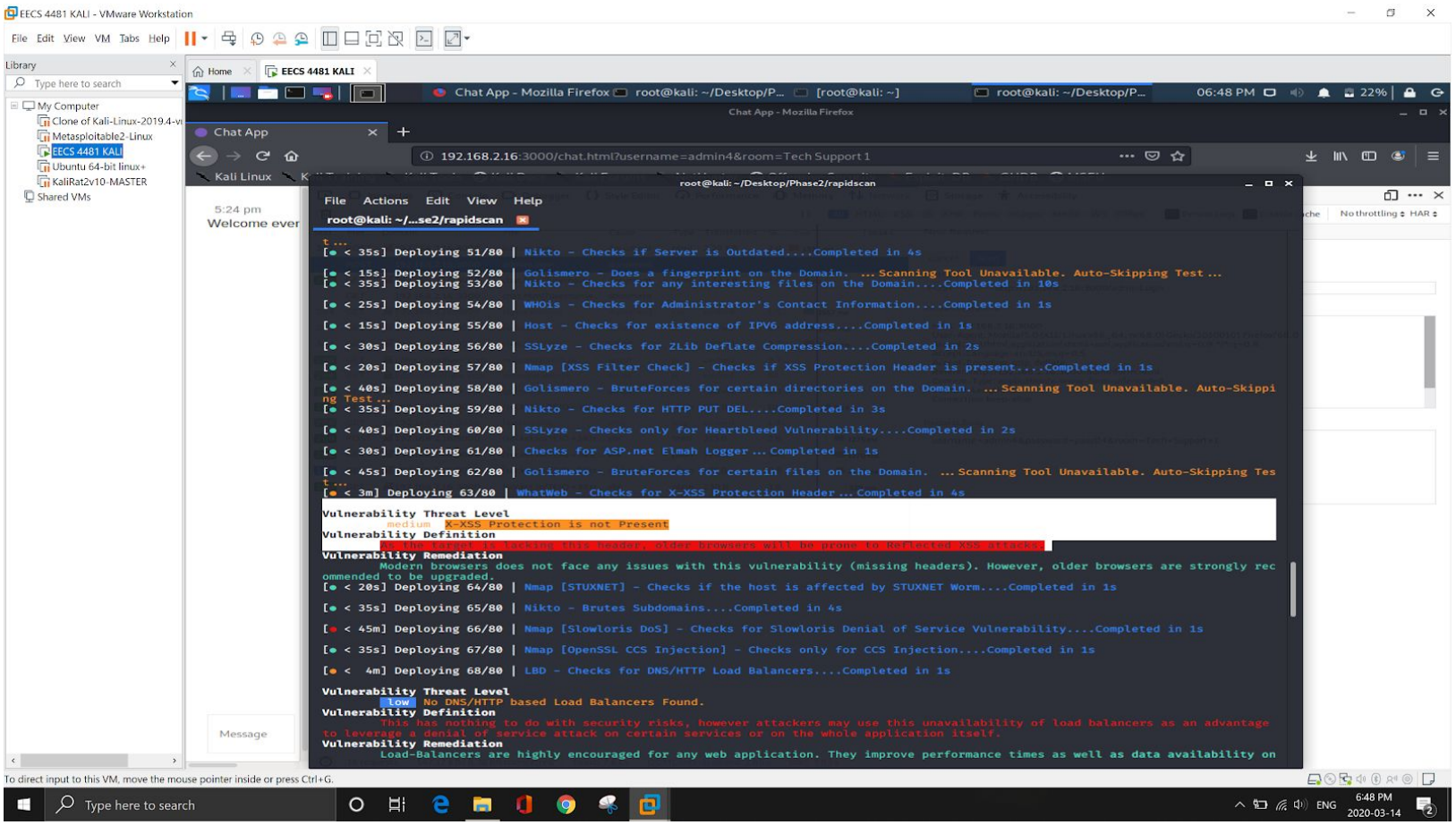
This shows the scanned report as well as log created by the scanner in 2 separate files:

RS-Vulnerability-Report

RS-Debug-Scanlog

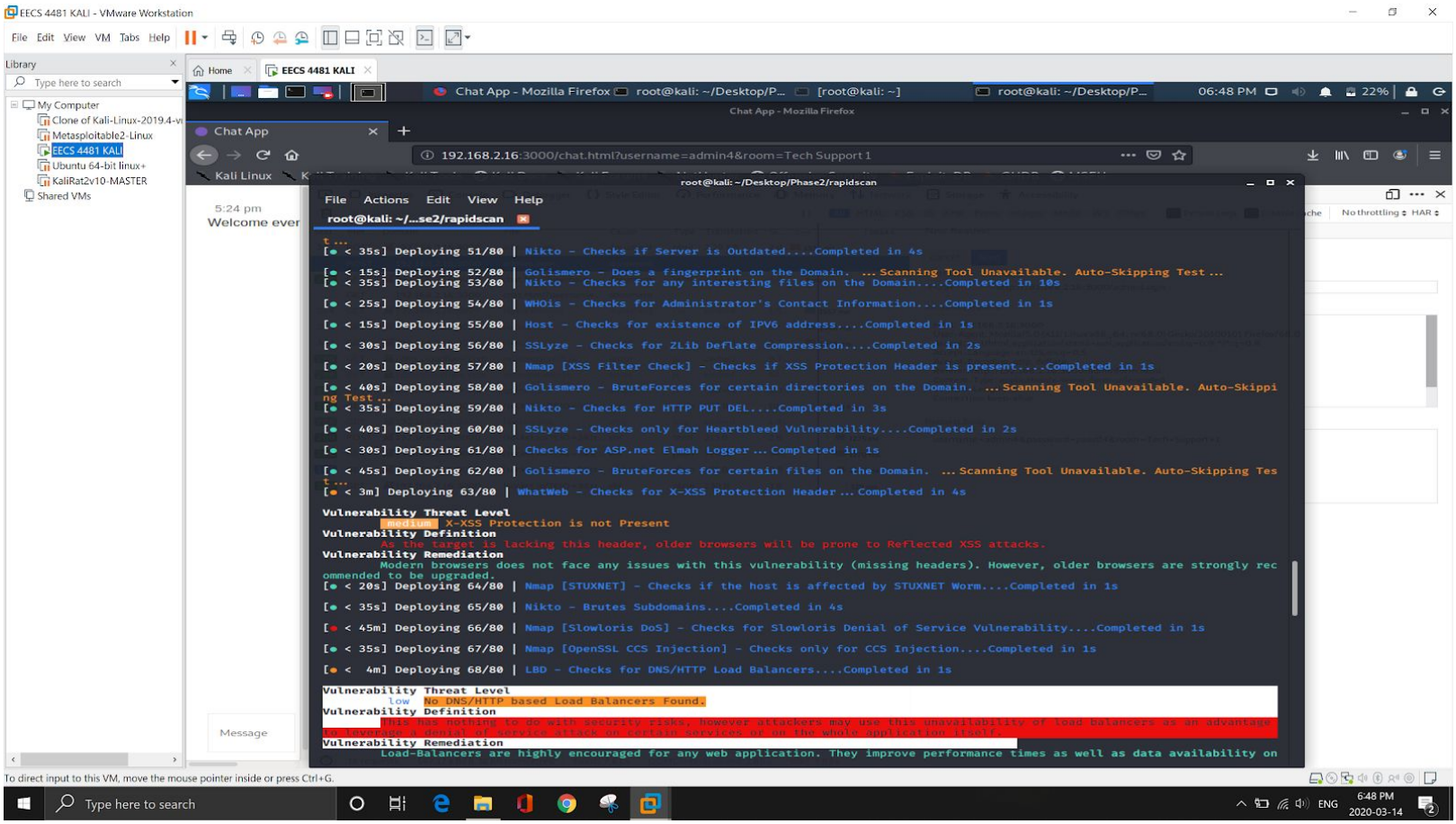
→ **Both of these files are attached in the dropbox for further references:**

6.2)



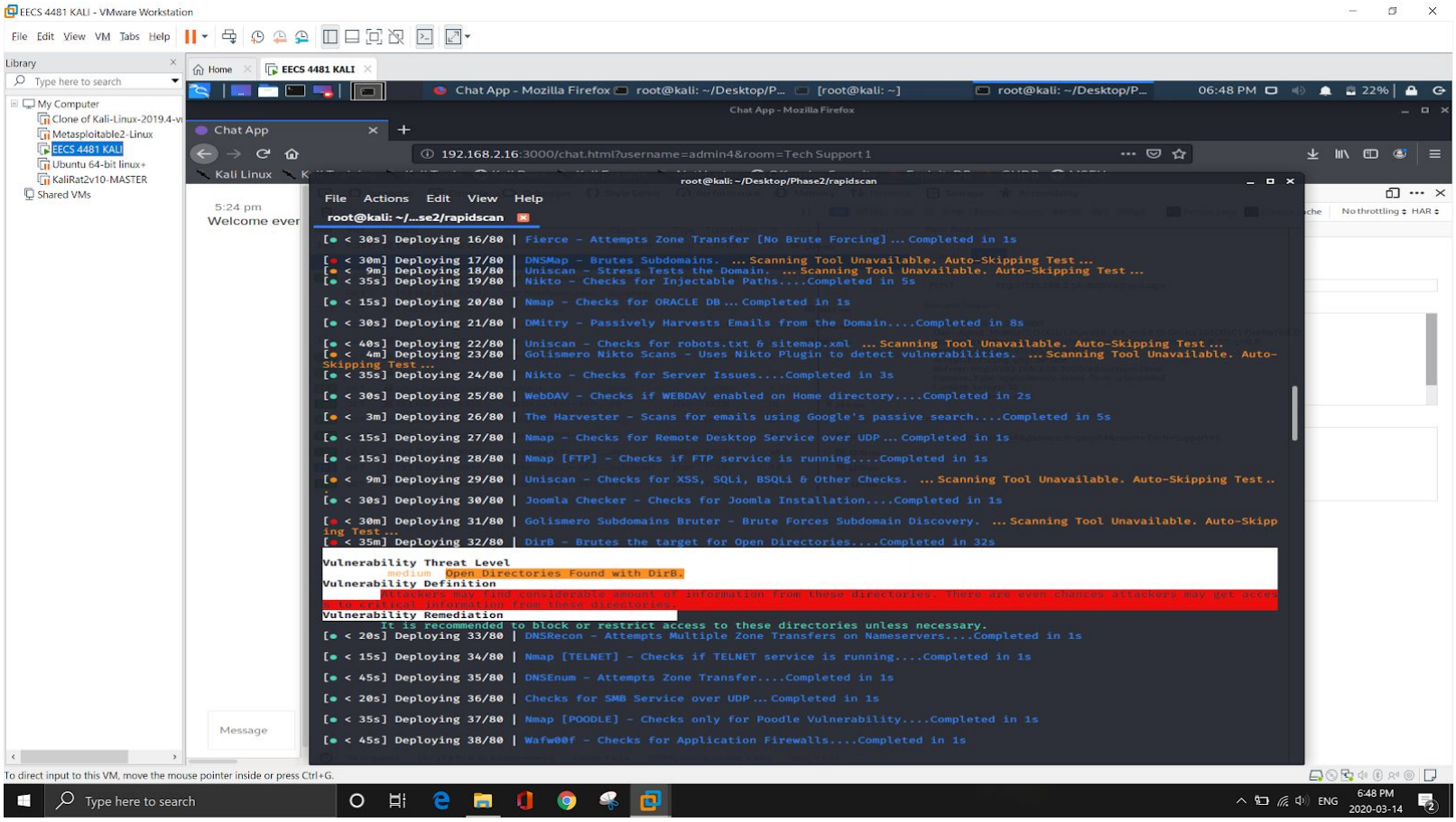
This scan confirms the XSS vulnerability implied by nikto previously. XSS-protection not set as well as target website lacking XSS header - **Making it prone to Reflected XSS attacks.**

6.3)



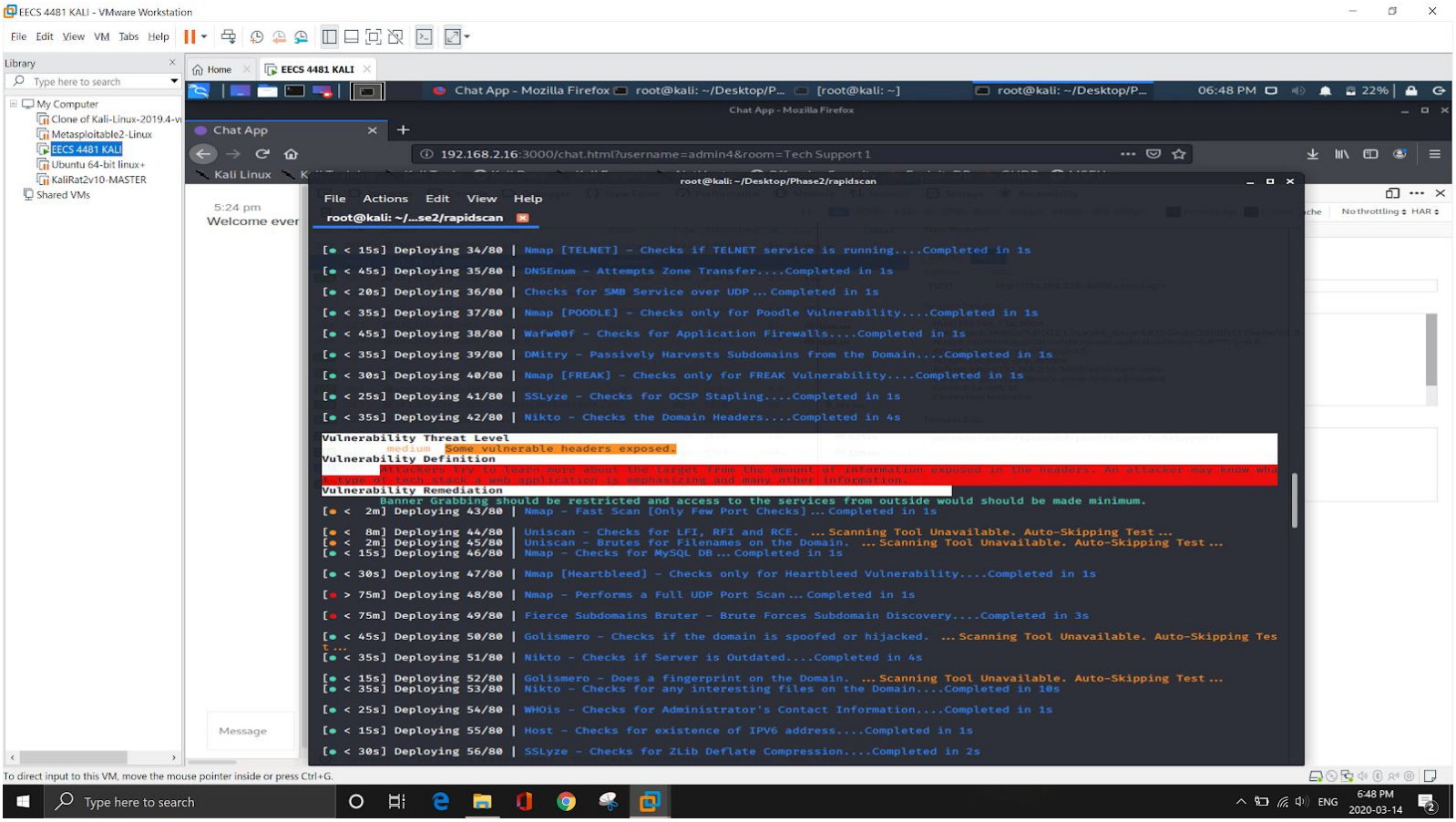
Another major vulnerability found is that there is no DNS/HTTP based Load Balancers Found by the scanner on the website machine. This provides an attacker a serious advantage as he can do a Denial of Service Attack on the web server without sending a high volume of packets, in contrast to a load balancer as nginx (the one in lab 5)

6.4)



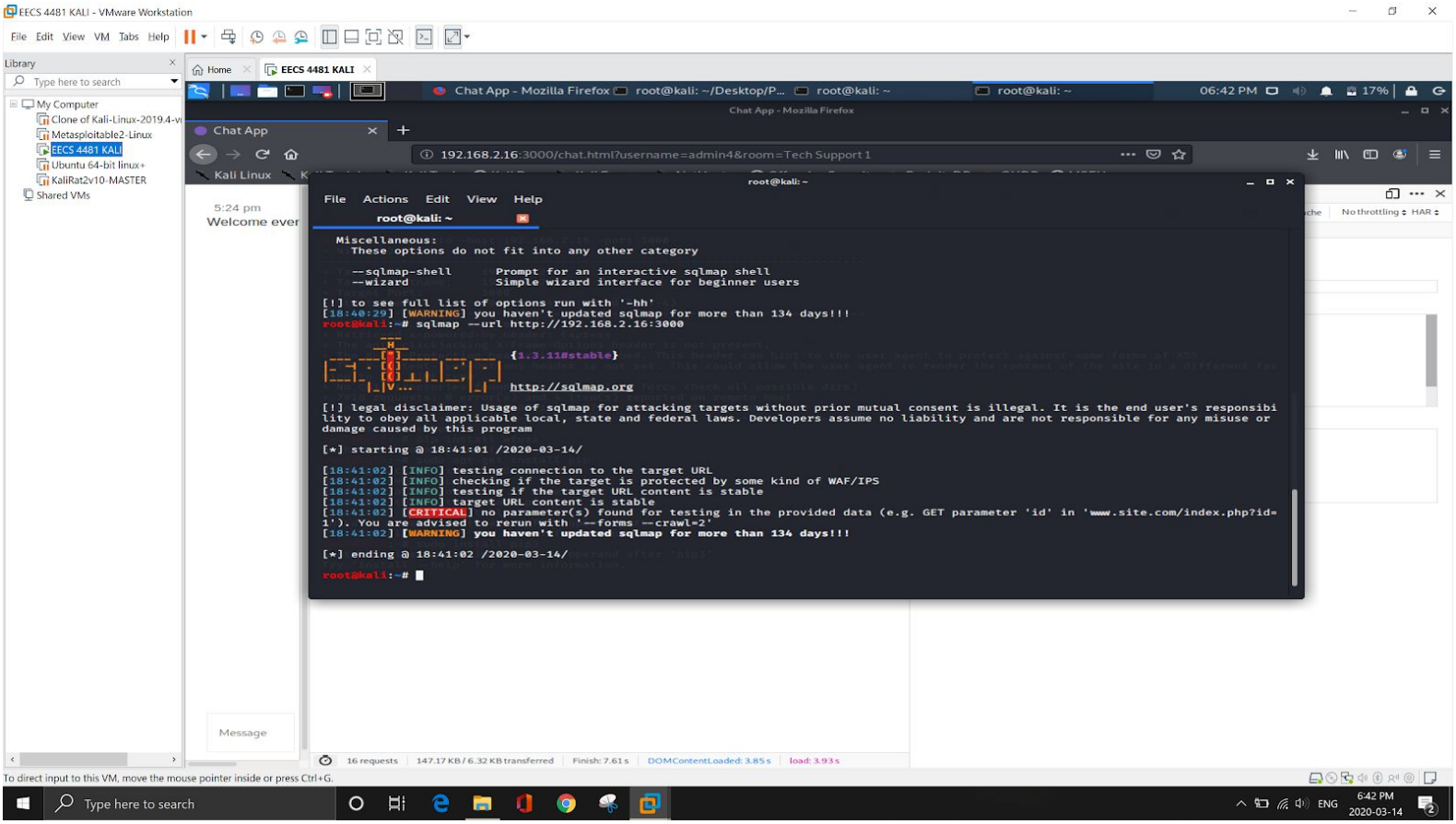
This result confirms with the results from dirbuster with open directories. The vulnerability addressing the concern about attackers finding valuable information from these folders which is enough to get access to critical information in these directories is a very important point here.

6.5)



This final vulnerability addresses vulnerable headers exposed which type of tech stack the application uses as well as apply banner grabbing which would further reveal critical information about the website as well as organization that owns the website.

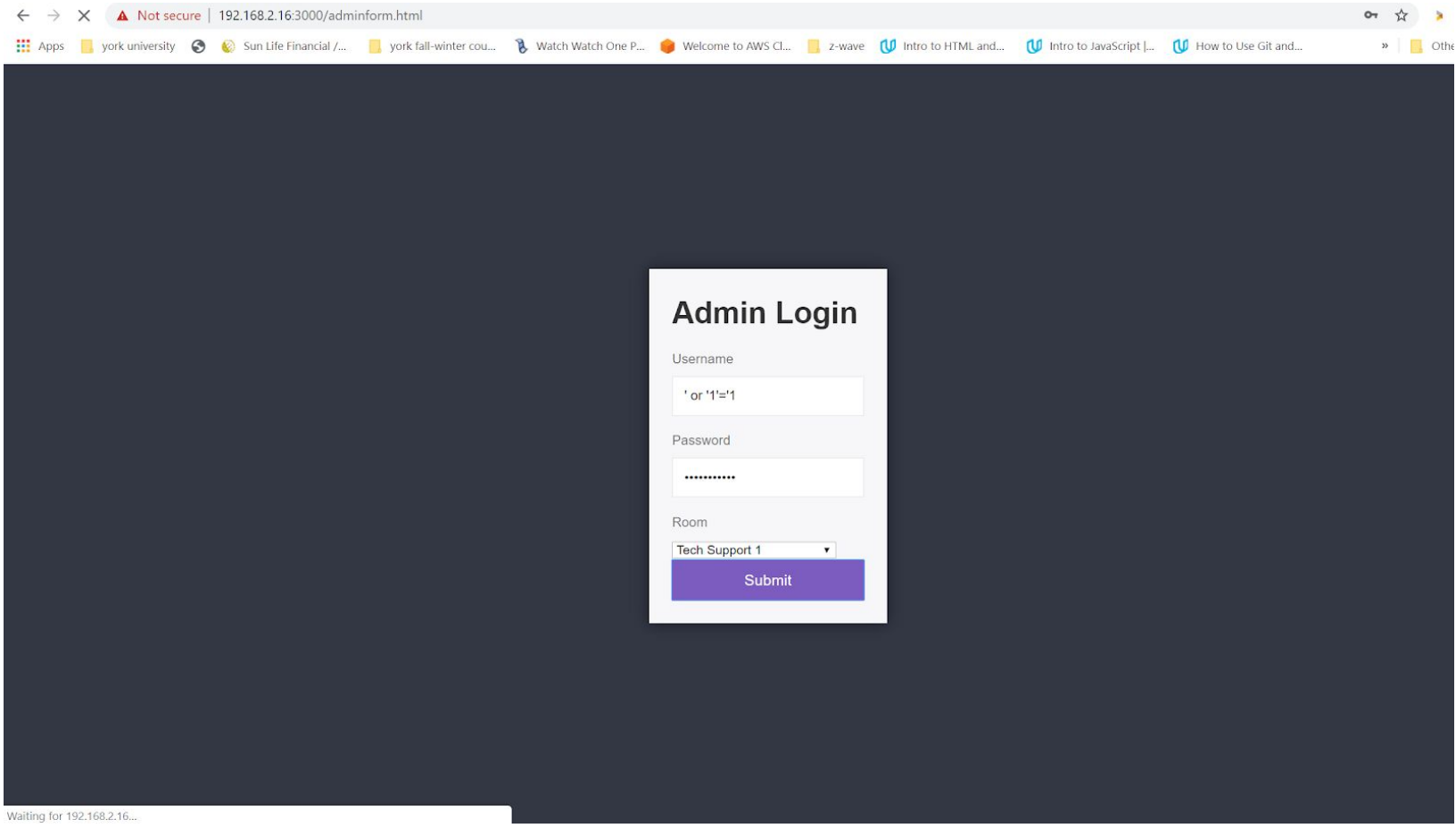
7)**SQLMap**



Using SqlMap didn't provide any worthwhile results, to confirm it further I moved to SQL Injection phase.

8) **Regarding SQL injection** , I injected a couple of statements to check the response from the server.

The server logged these in the backend and didn't give any response.



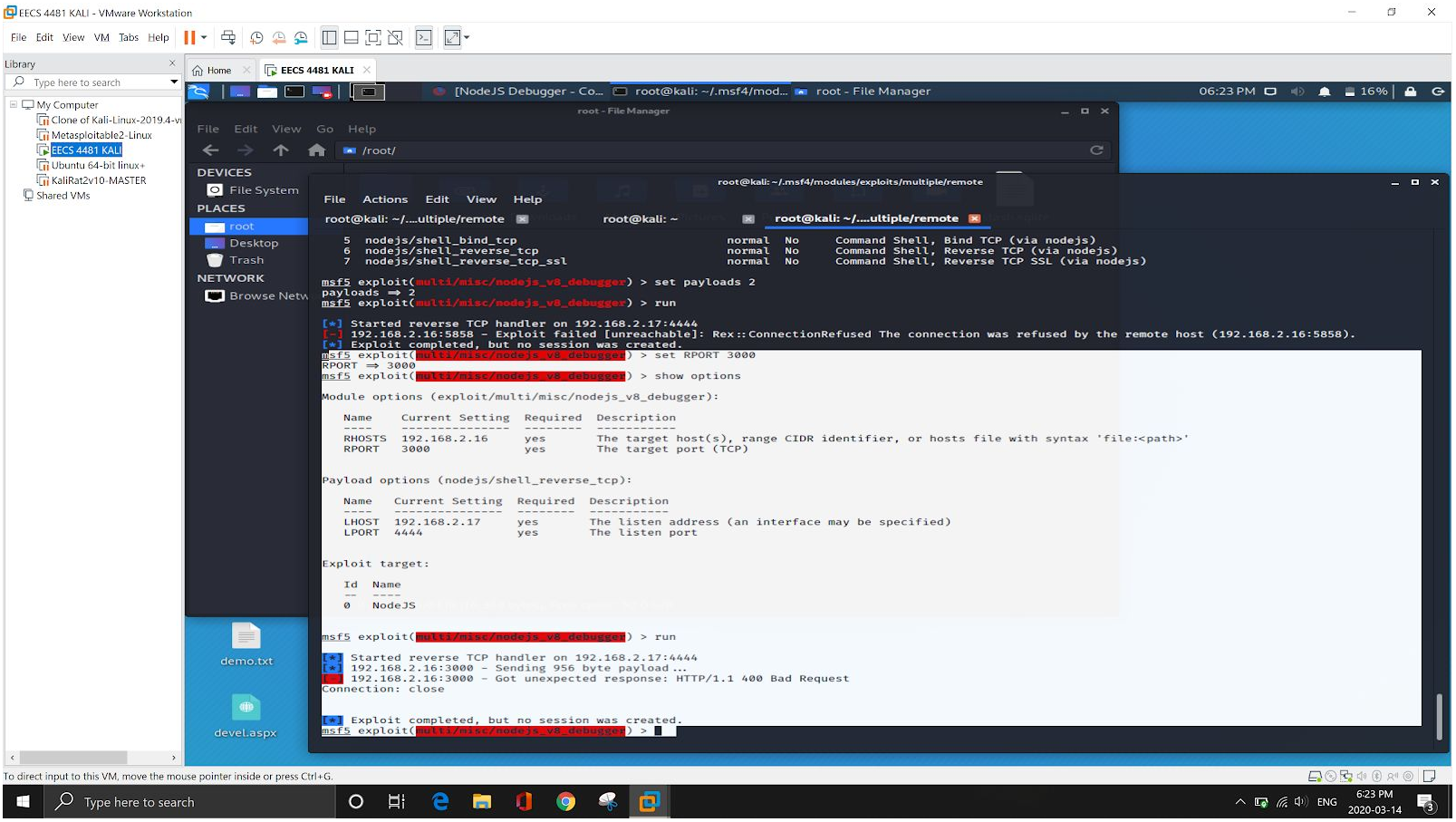

```
These are the credentials requested by the admin -->  { username: 'admin', password: `' or 1=1"`, room: 'Tech Support 1'
}
admin login page for credentials!!
These are the credentials requested by the admin -->  {
  username: "' or '1'='1",
  password: "' or '1'='1",
  room: 'Tech Support 1'
}
```

The logs above show that the attacker tried to pass SQL statements in the password field in the first

attempt as well as the username & password pair in the second attempt .

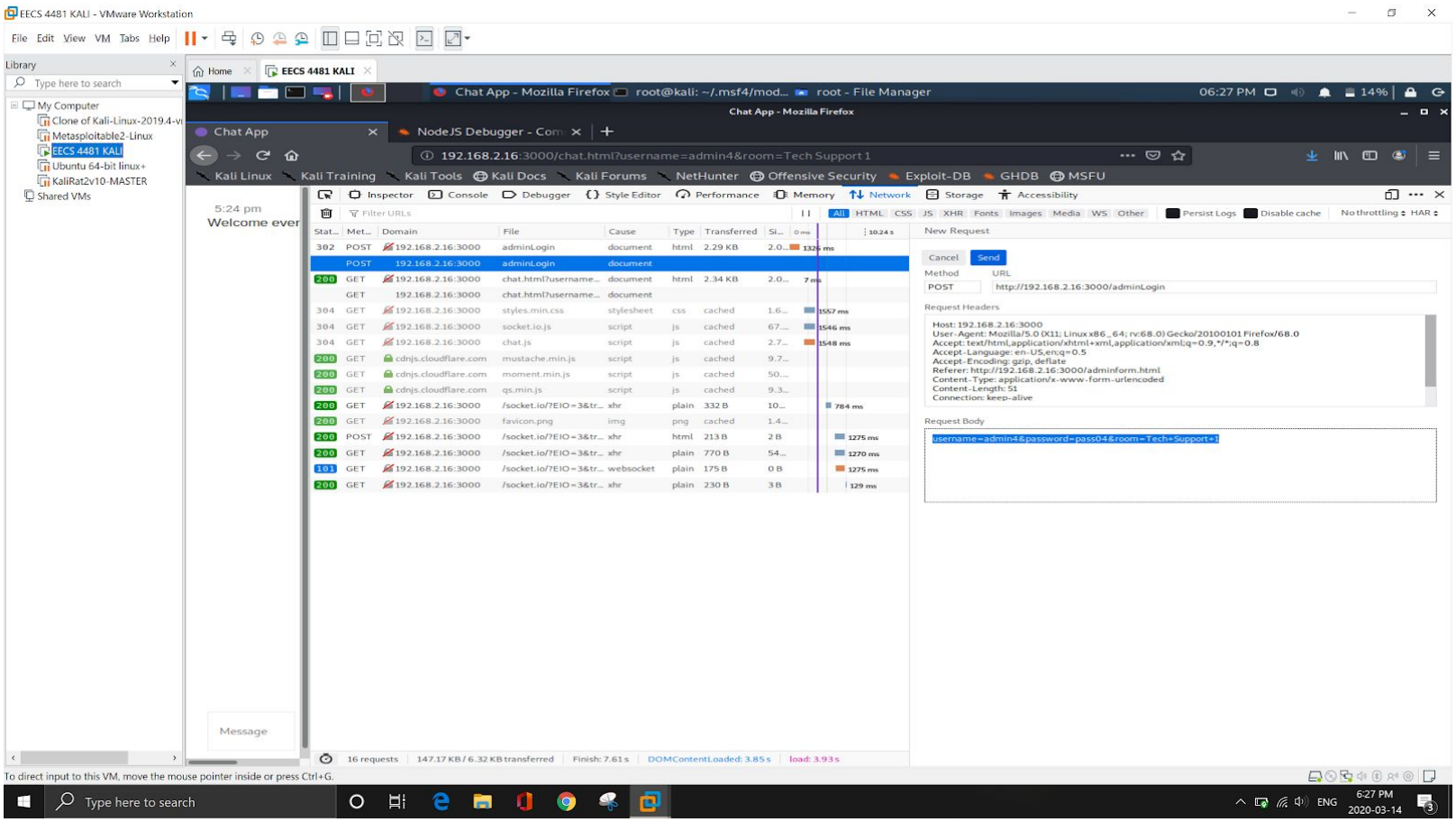
NOTE : Since the database is noSQL using MongoDB, the database is safe from SQLinjection attacks.

9) Finally, based on the nmap result of specifying that the application has the framework of Express Node.js , I tried several **exploits on node js using metasploit** and tried to get tcp reverse shells through payloads.



The image above shows that the exploit was completed , but it was unable to create a session. This confirms that further analysis with writing custom scripts and injecting them with msfvenom while the session running might work for later phases.

10) **the password cracking for Hydra** based on website query strings as well as other information wasn't successful.



A more extensive scan with burp suite and using the results from that to Hydra might work for sure.