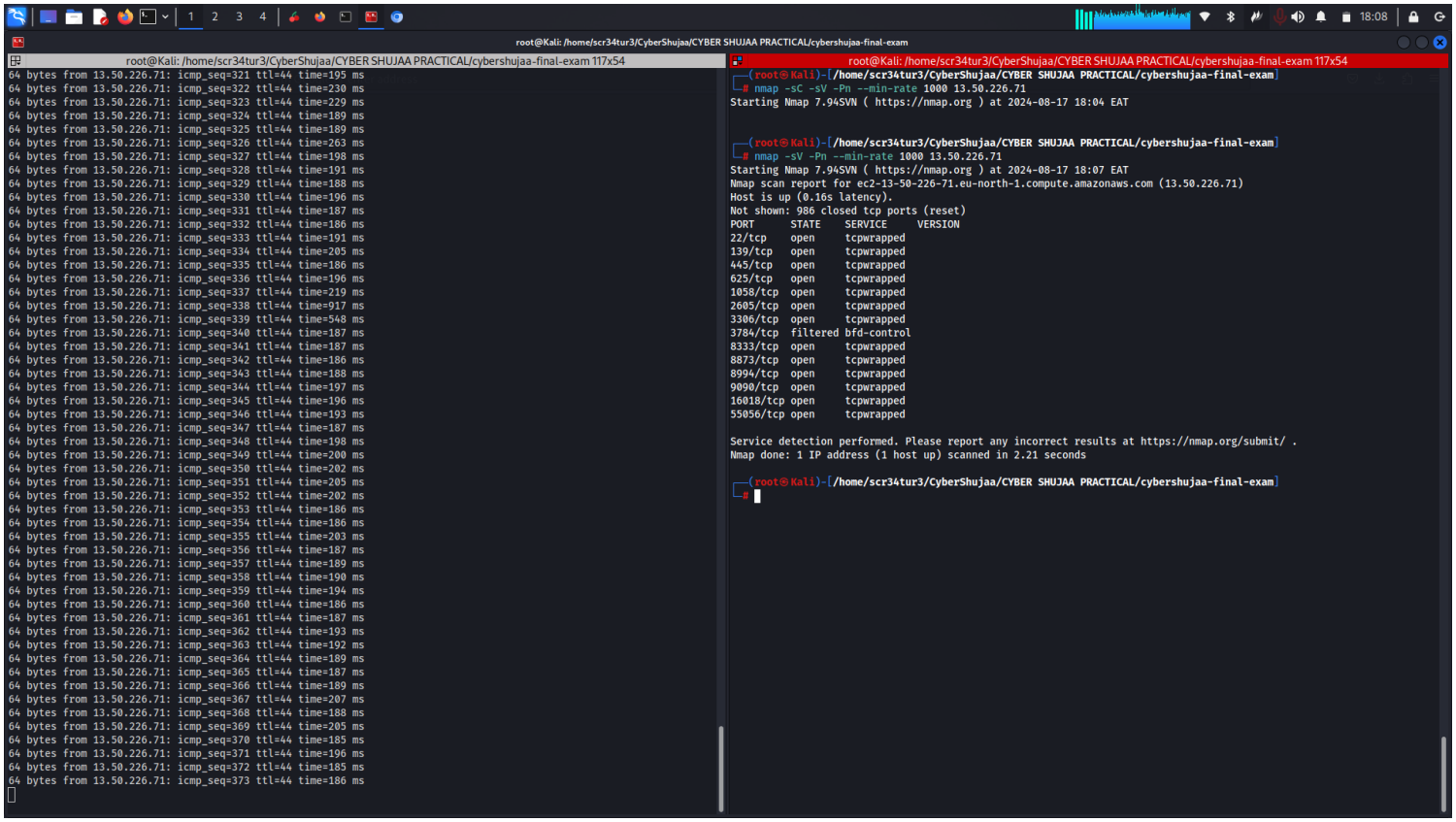
WEB-PENTEST

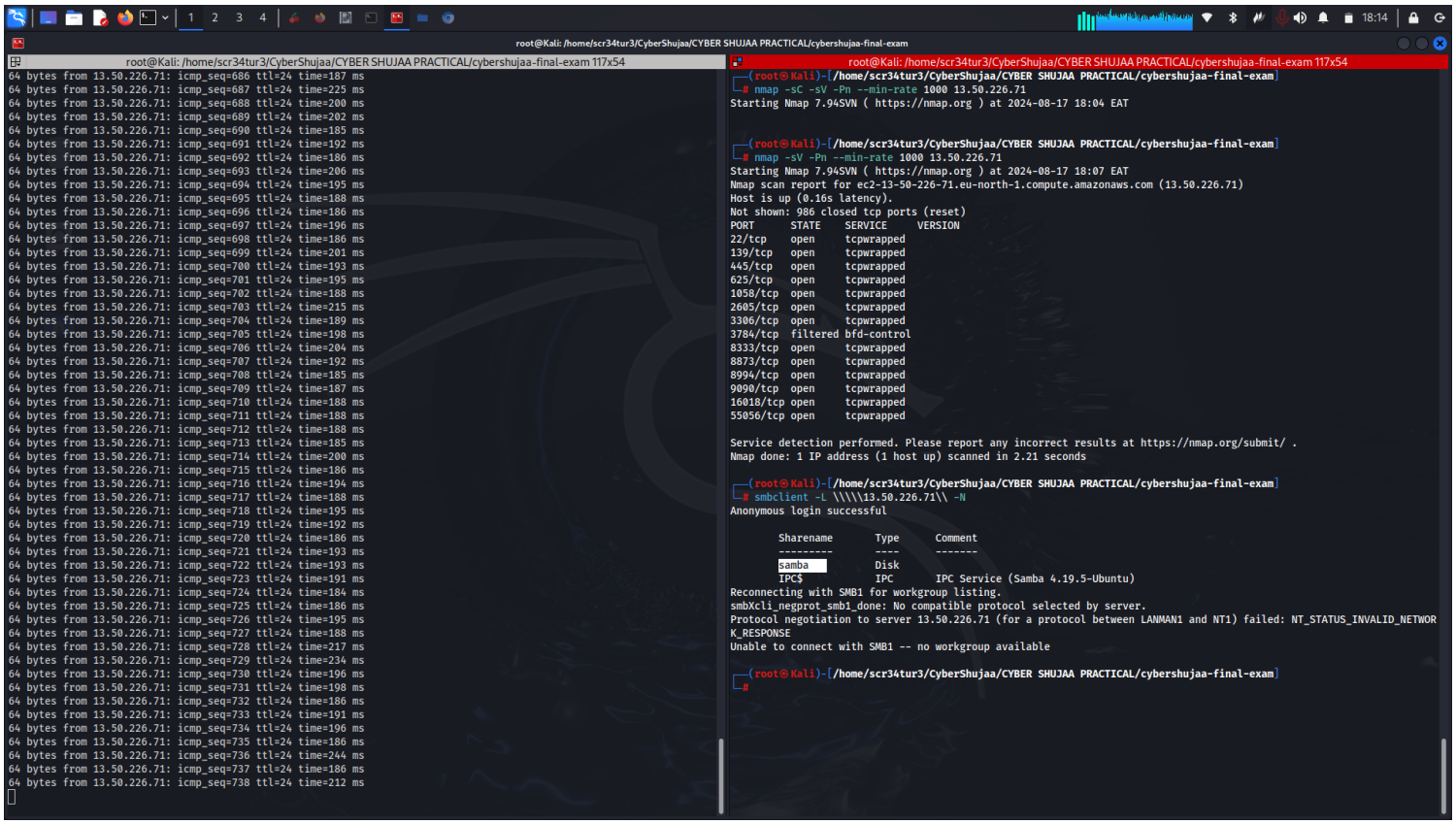
INTRODUCTION

Scenario:
You have been recently hired as a penetration tester by Secure Technologies Limited. They have given you their corporate server for you to check the various ways in which a determined attacker can gain access to it.
Having this information, Lets get started.

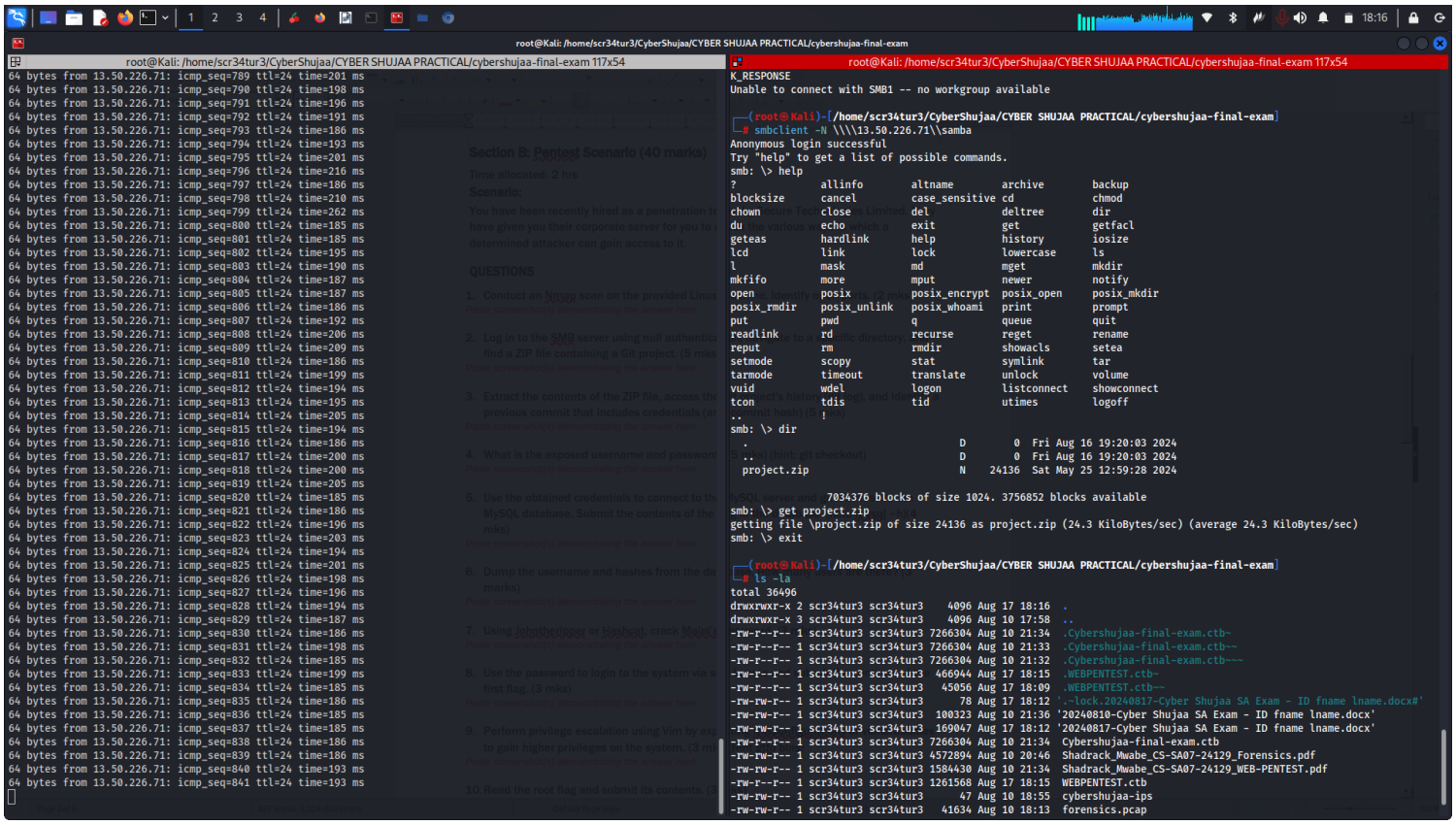
I did an nmap scan against the target as seen below. I noticed some common ports that port 22, port 139 and port 445 on which **ssh, SMB (running over netbios) and SMB (running over tcp/ip)** services were running respectively. This was among other ports as seen below.



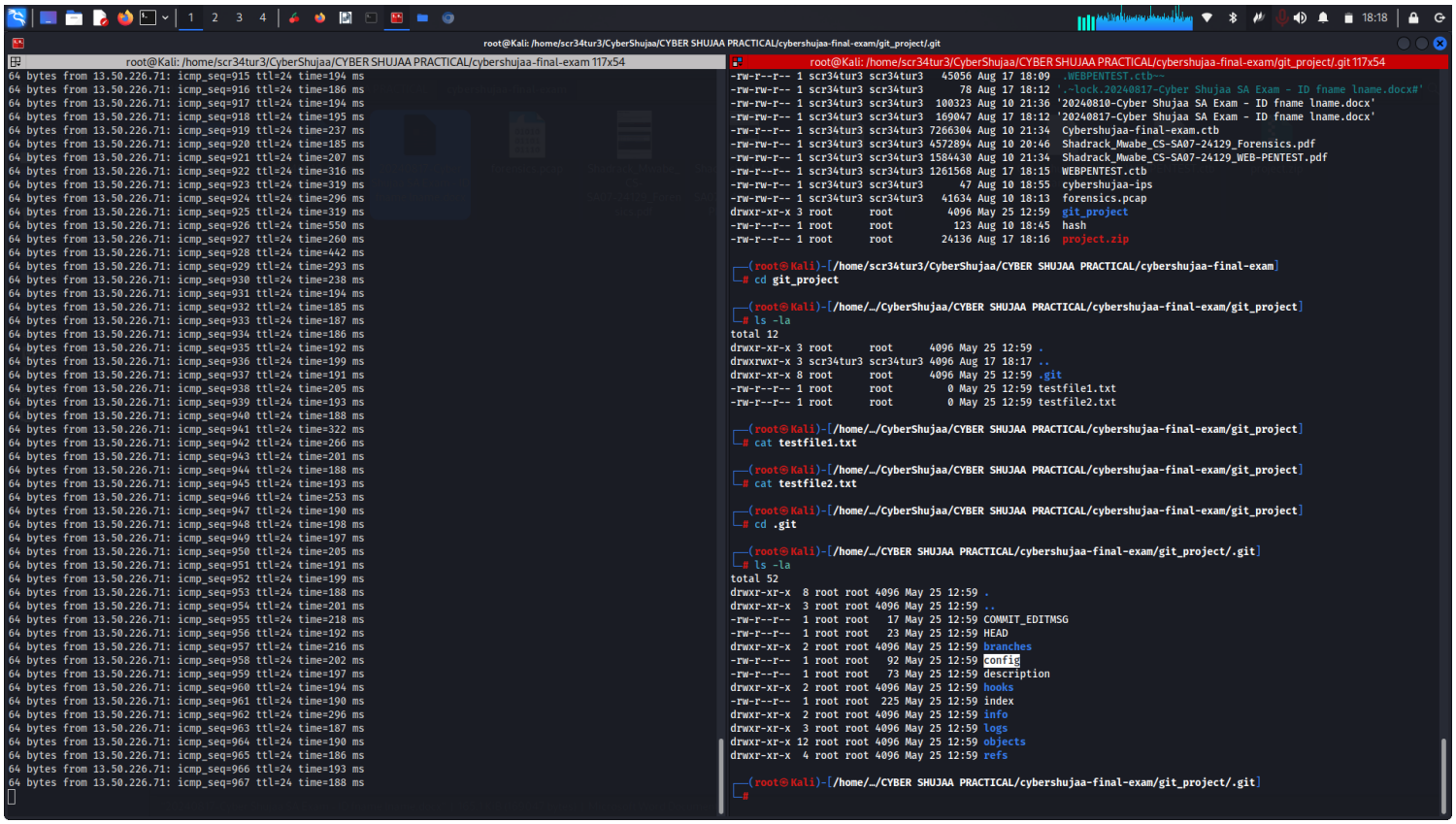
I first used the **smbclient cli tool** with the -L option to list shares under the SMB service as seen in the image below.



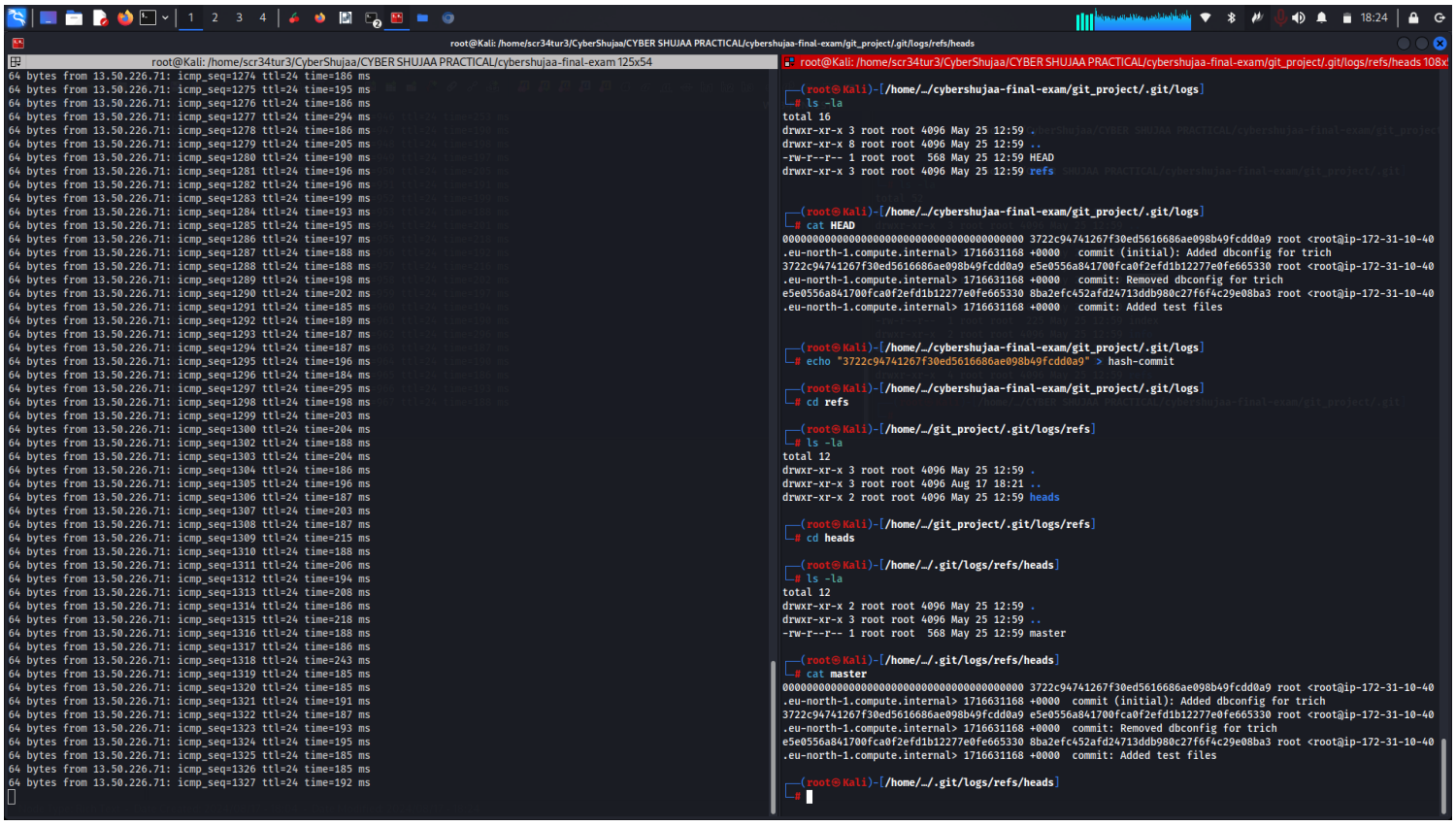
I then used the -N option for null or no-pass authentication and successfully downloaded the project.zip file to my local machine just as shown below.



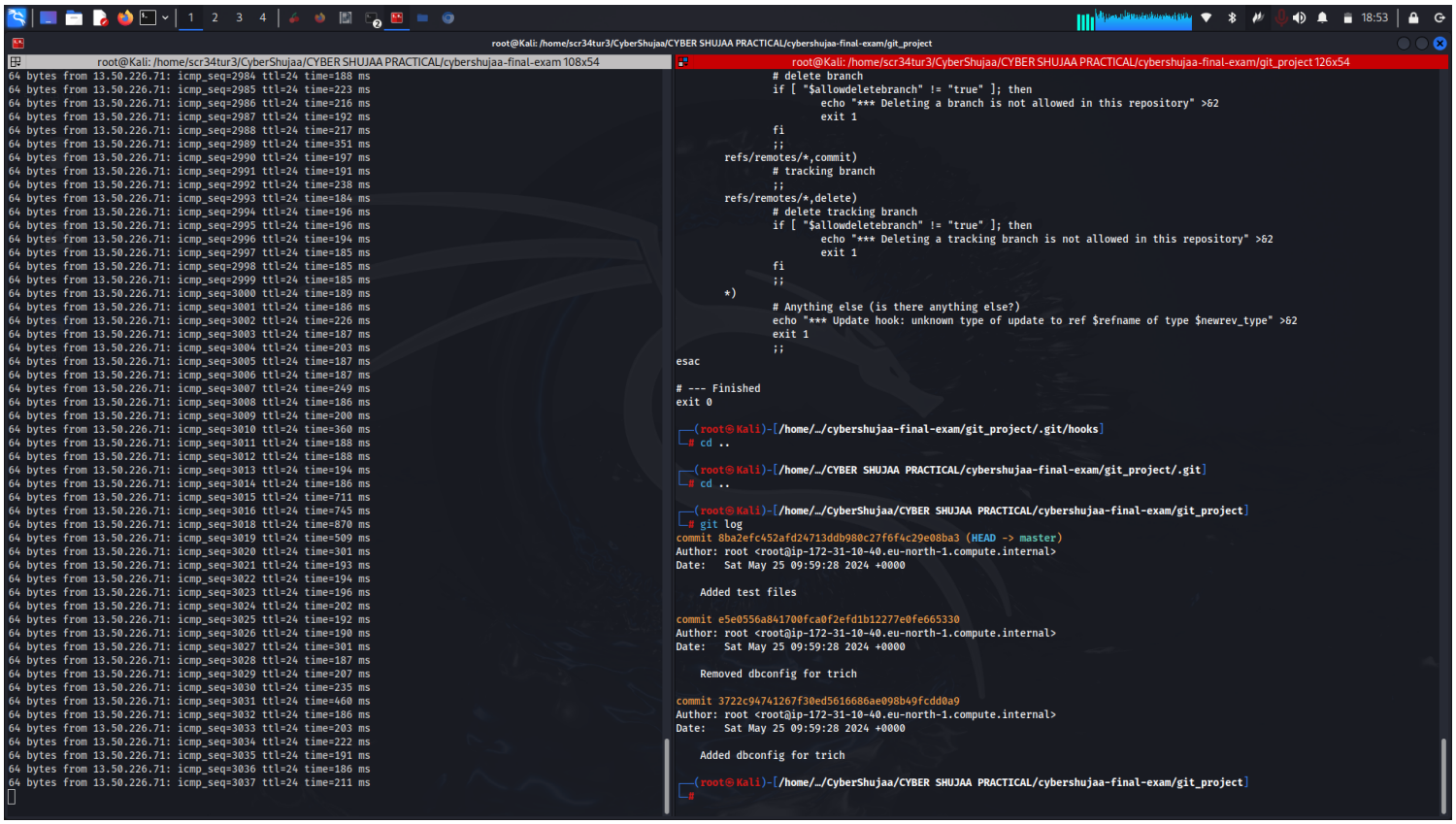
I unzipped the project.zip file, and changed dir to the extracted directory as seen below.



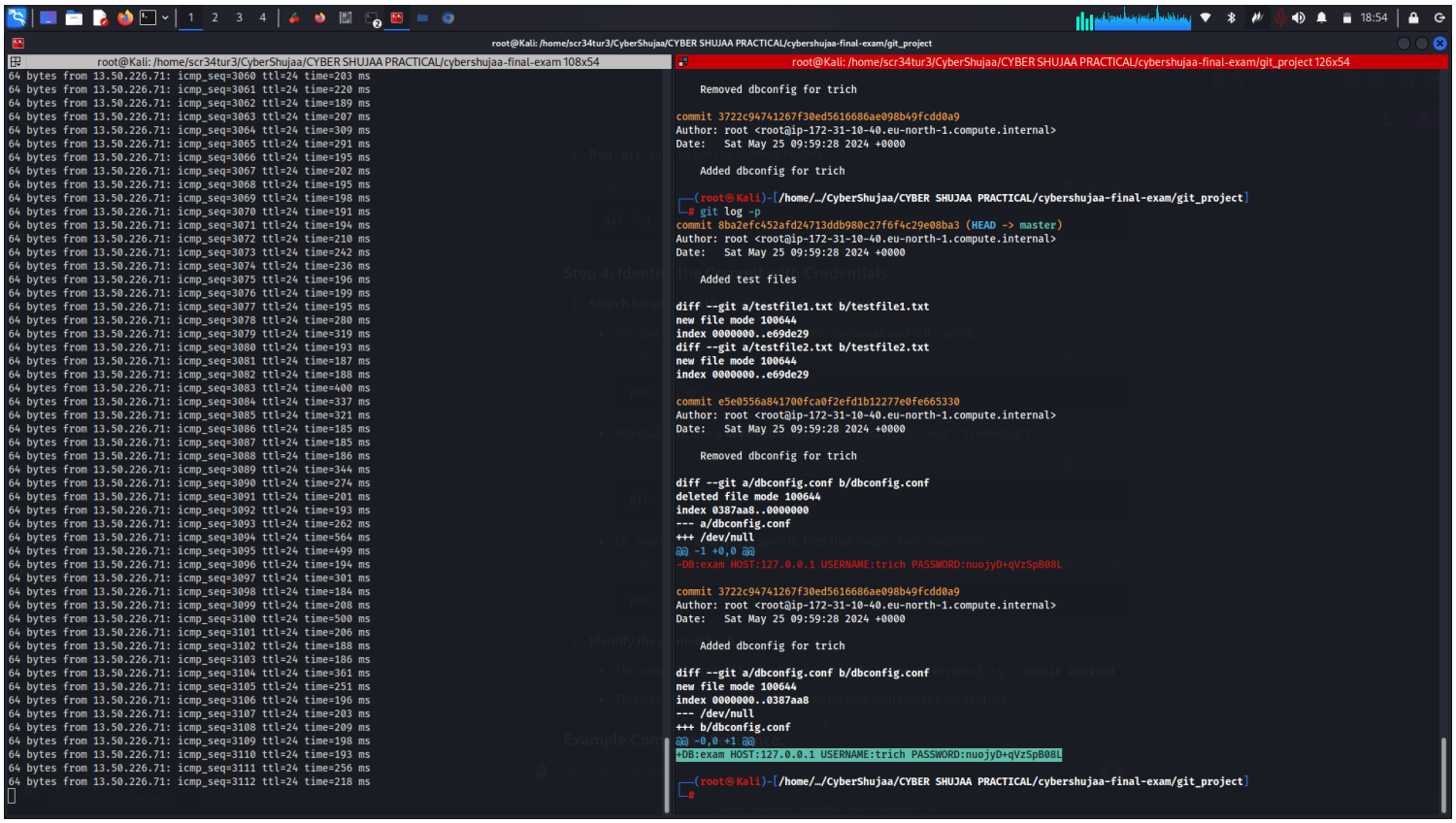
I manually navigated around to see the contents under this directory and found a file that had hashes as seen below.



However another easy and faster way, I was to check on the git history using the command git log. And as seen below, I was able to see hashes from the previous commit.

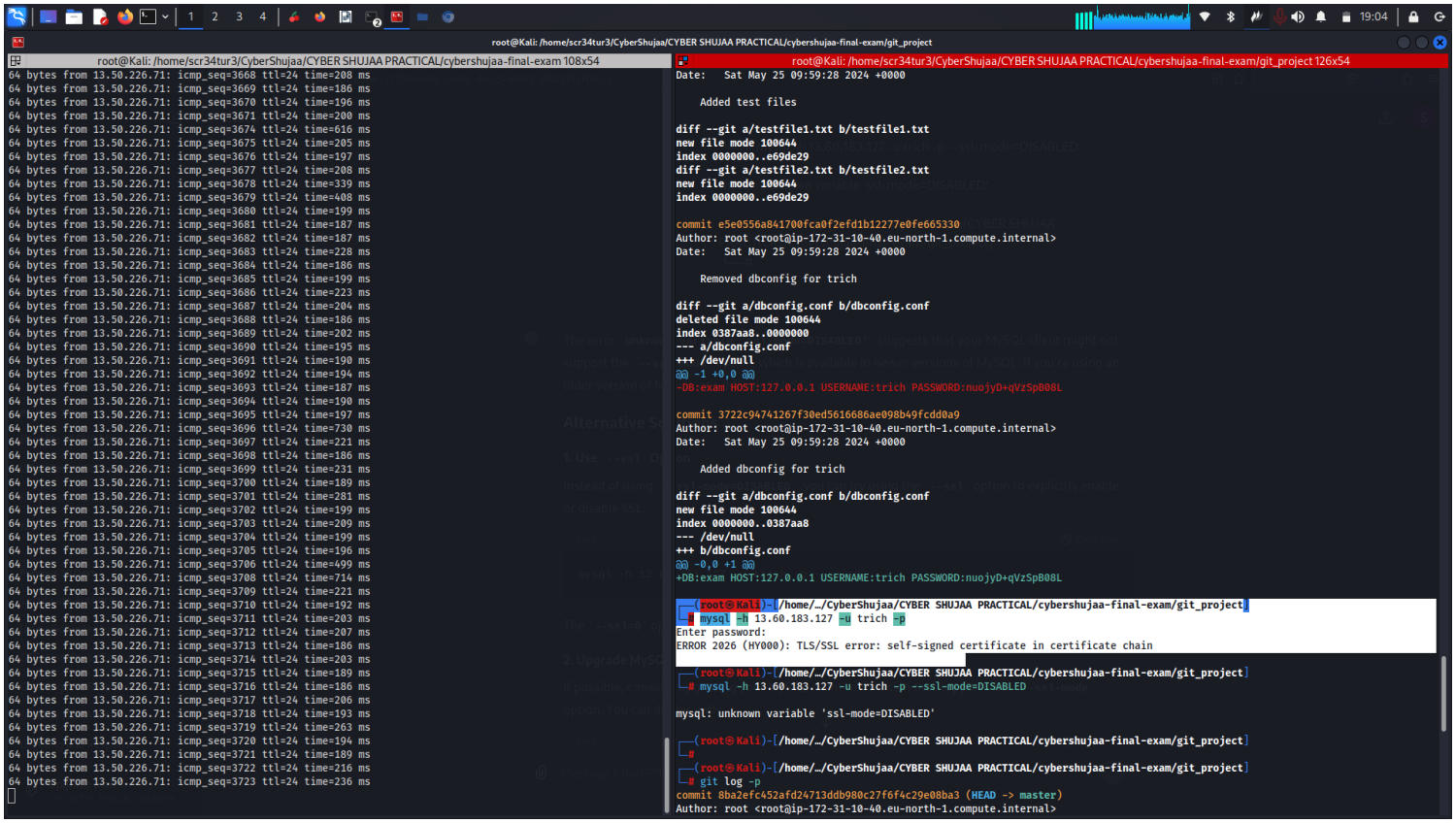


I used the **git log -p** cmd which is used to display the commit history in a Git repository along with the patch (differences) introduced in each commit as seen below. Found plain cred of **USERNAME:trich PASSWORD:nuojyD+qVzSpB08L**

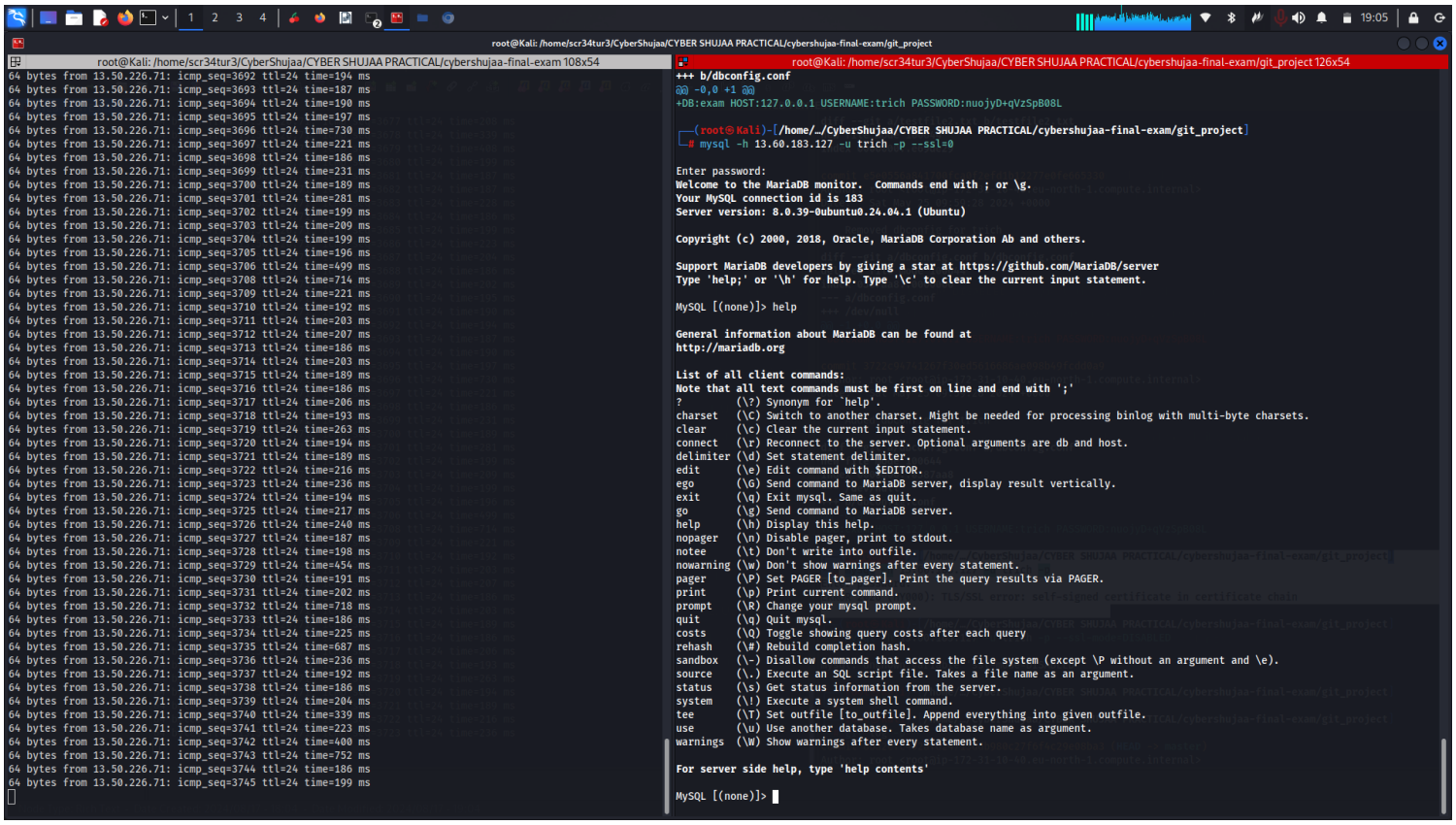


Using the obtained creds, I tried to login to the mysql server, however I kept on receiving an error as indicated in the image below. This was because MySQL was trying to establish a secure (SSL/TLS) connection, but the certificate it is

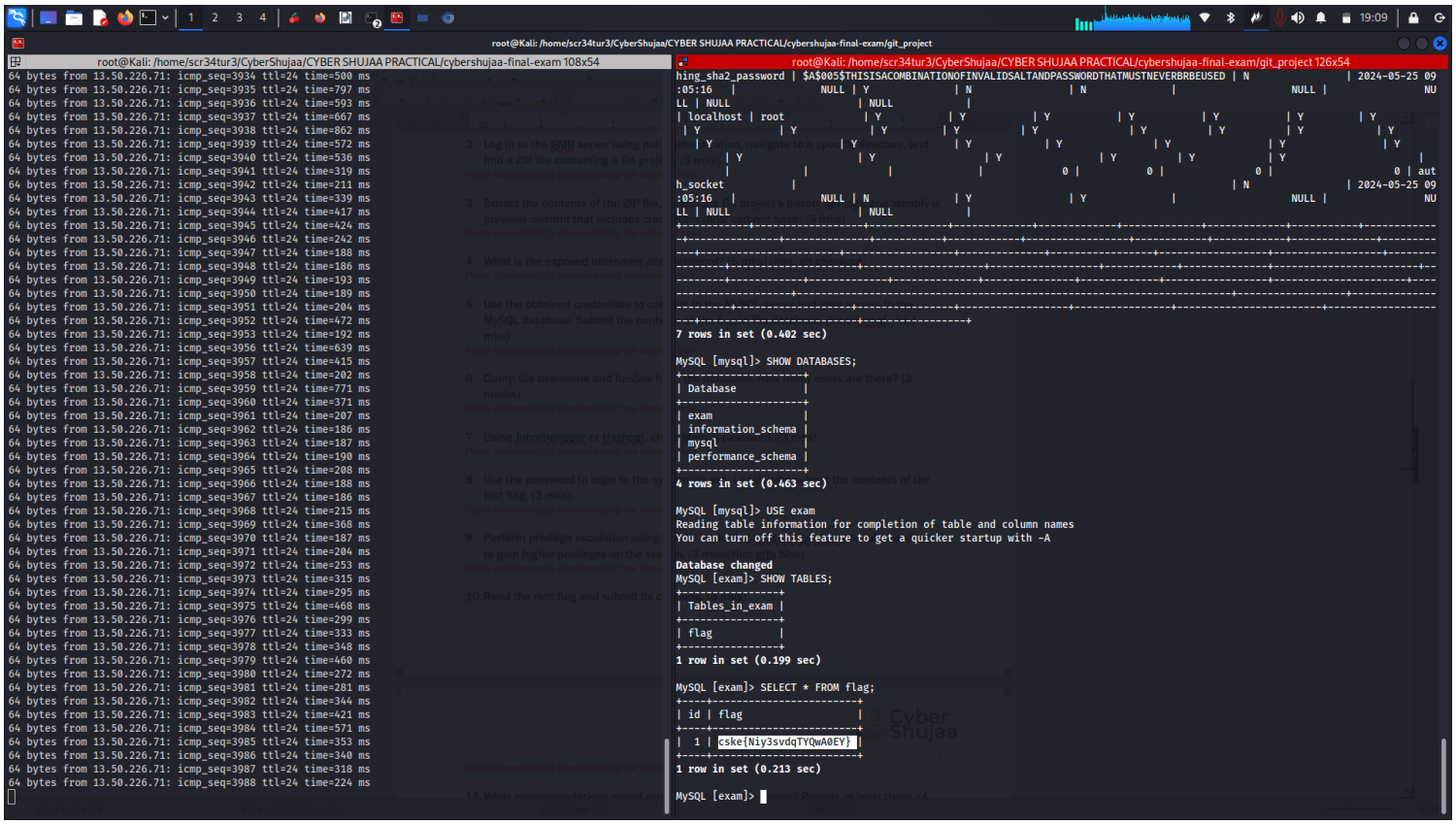
using is self-signed and not trusted by your system.



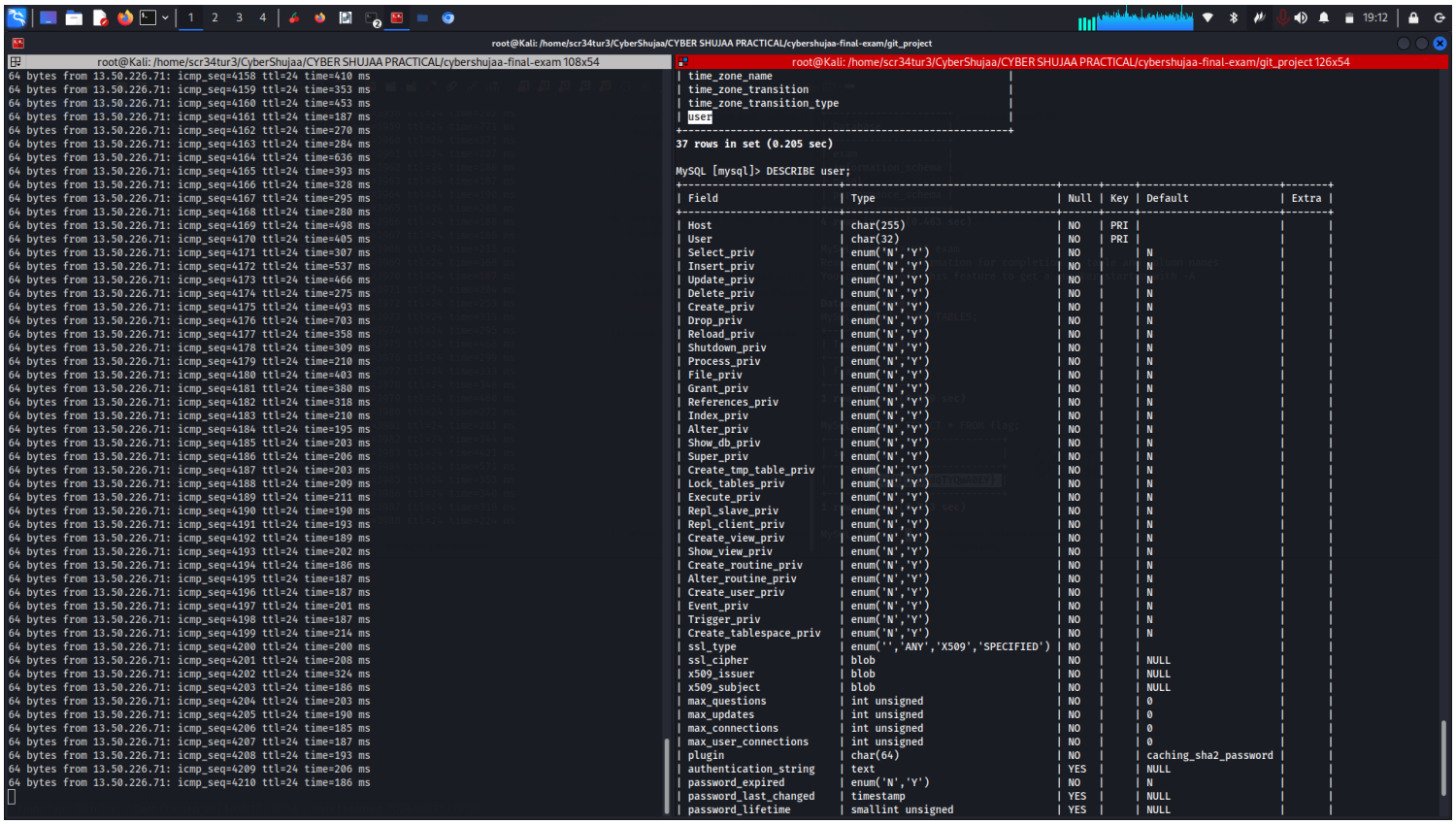
So I used the **--ssl=0** to disable SSL. By this I successfully logged in to the mysql server as user trich as seen below.



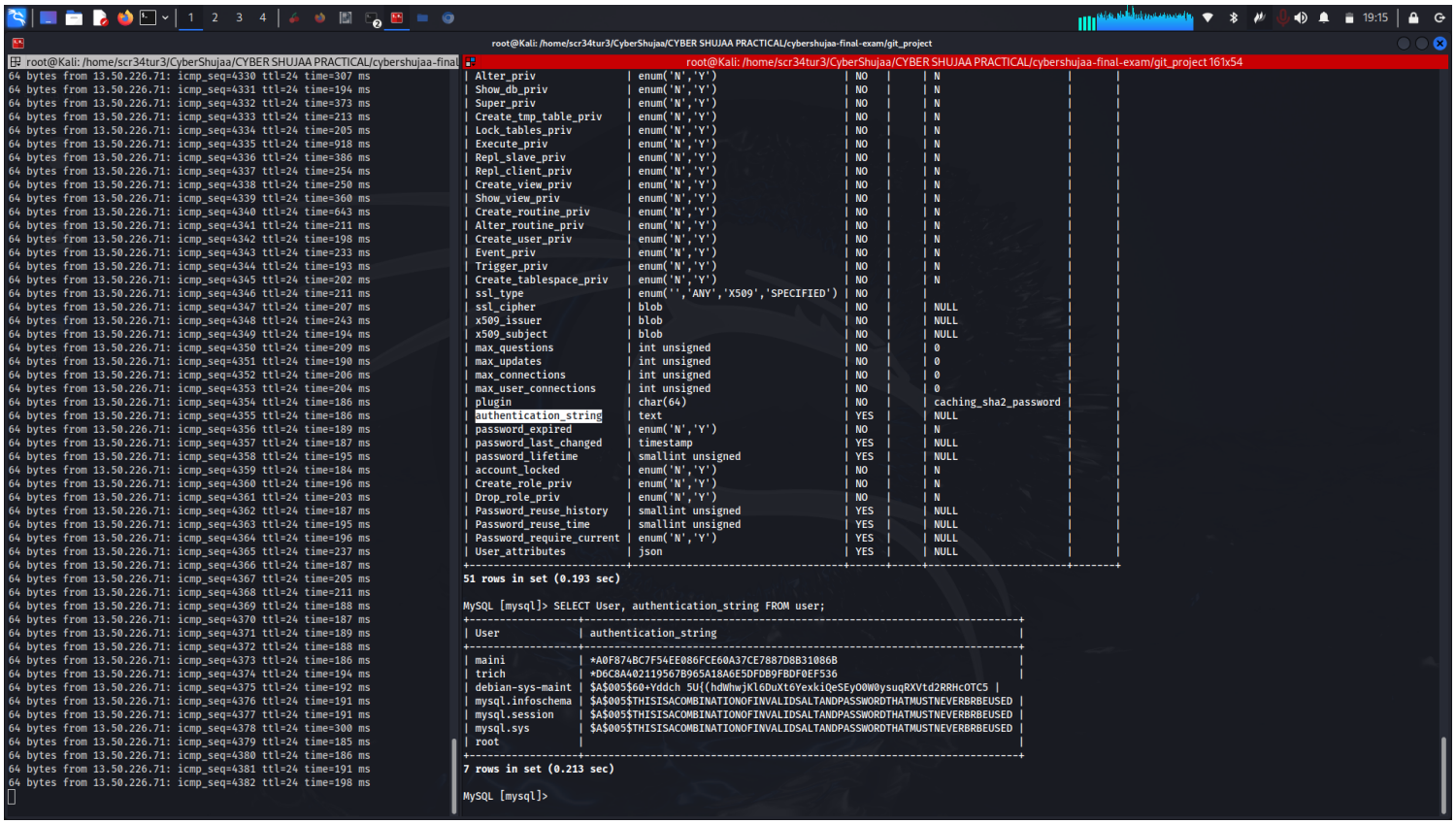
Under the exam DATABASE, there was a table called flag, that most probably contained the flag. Accessing this table as seen below, I was able to retrieve the first flag.

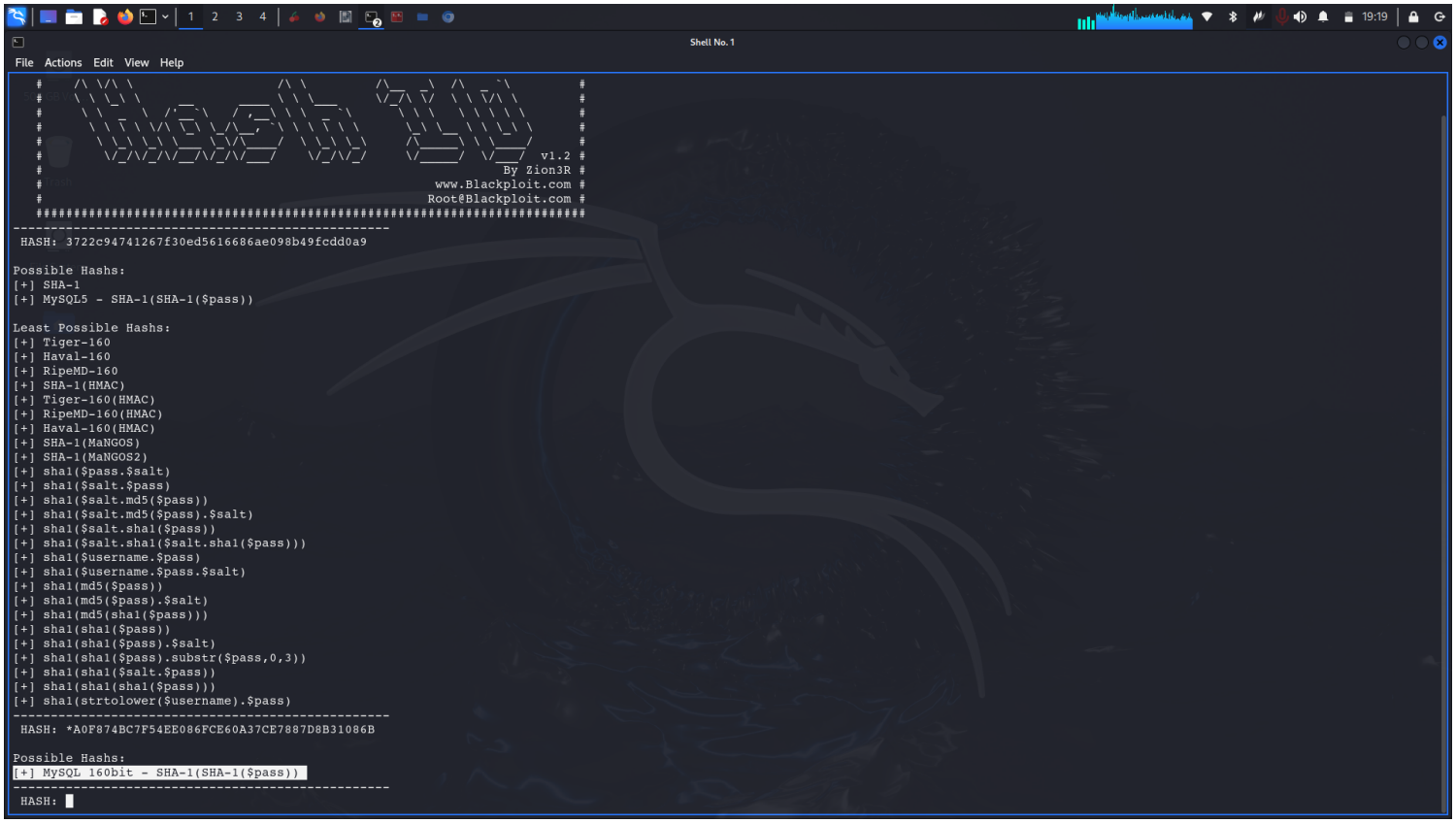


I first used the **DESCRIBE** cmd to see the available columns under table user.

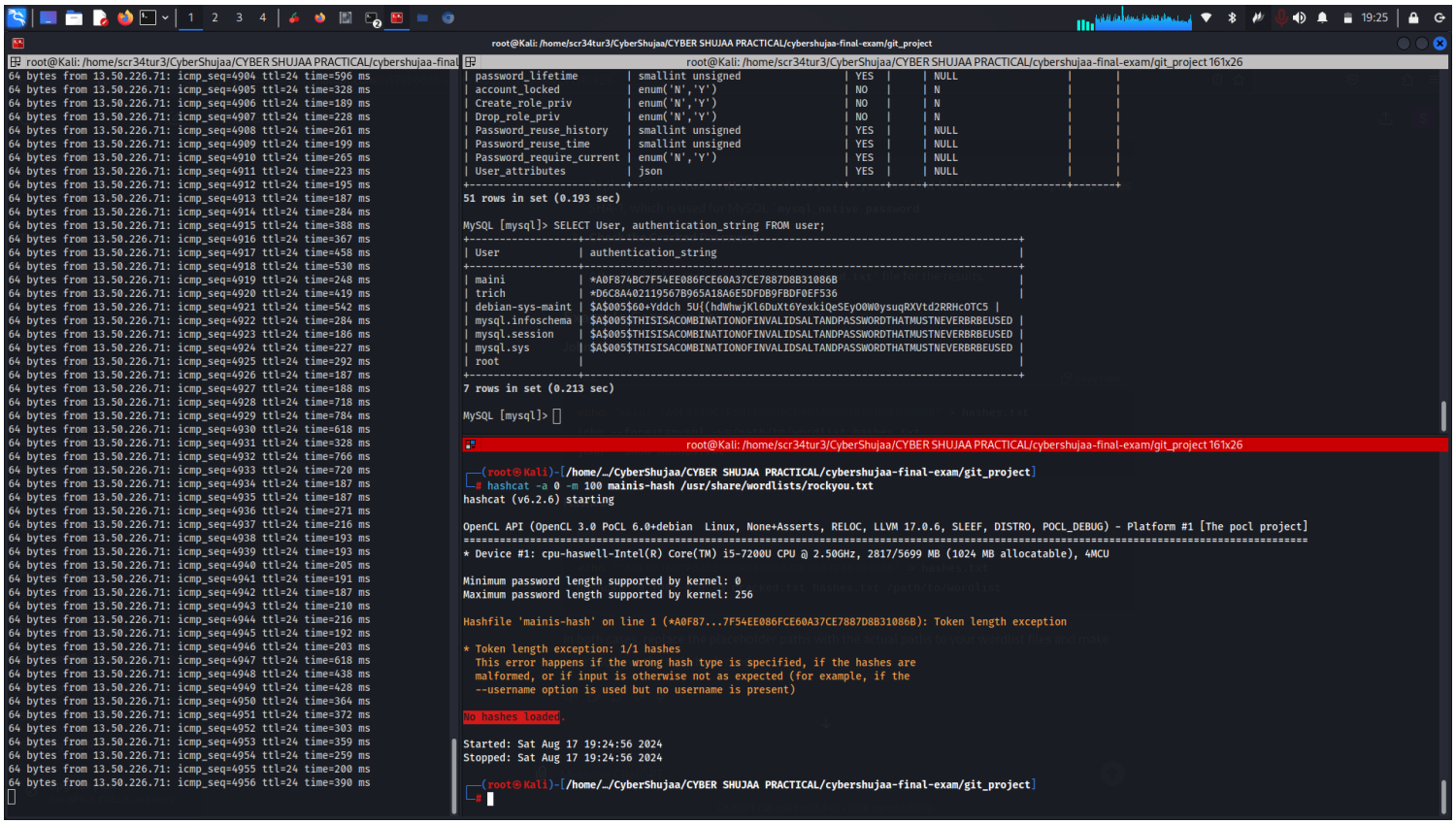


Looking closely, I noticed this column **authentication_string** that contained the password hashes as seen below. With this knowledge, I successfully dumped users and password hashes.

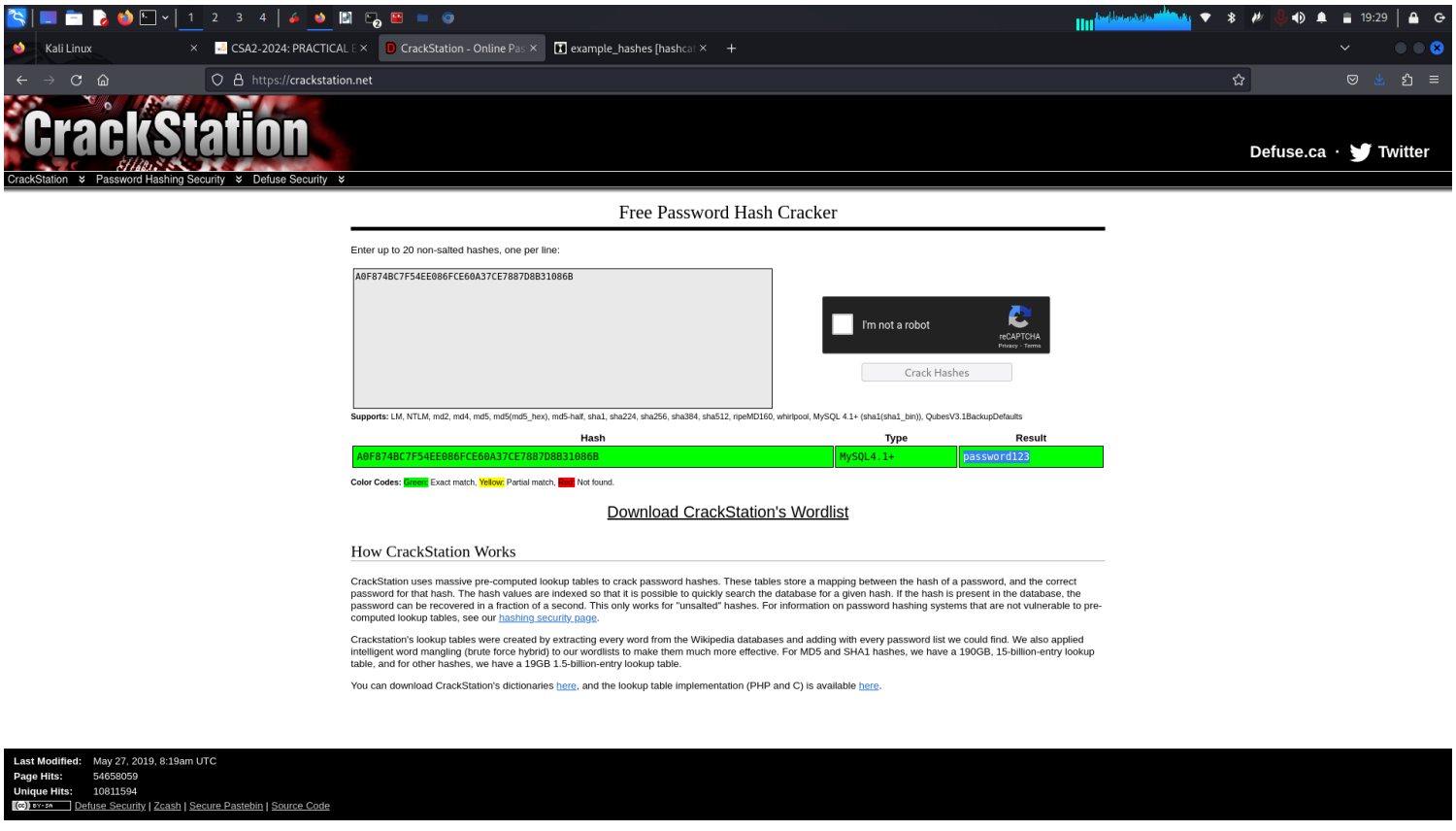




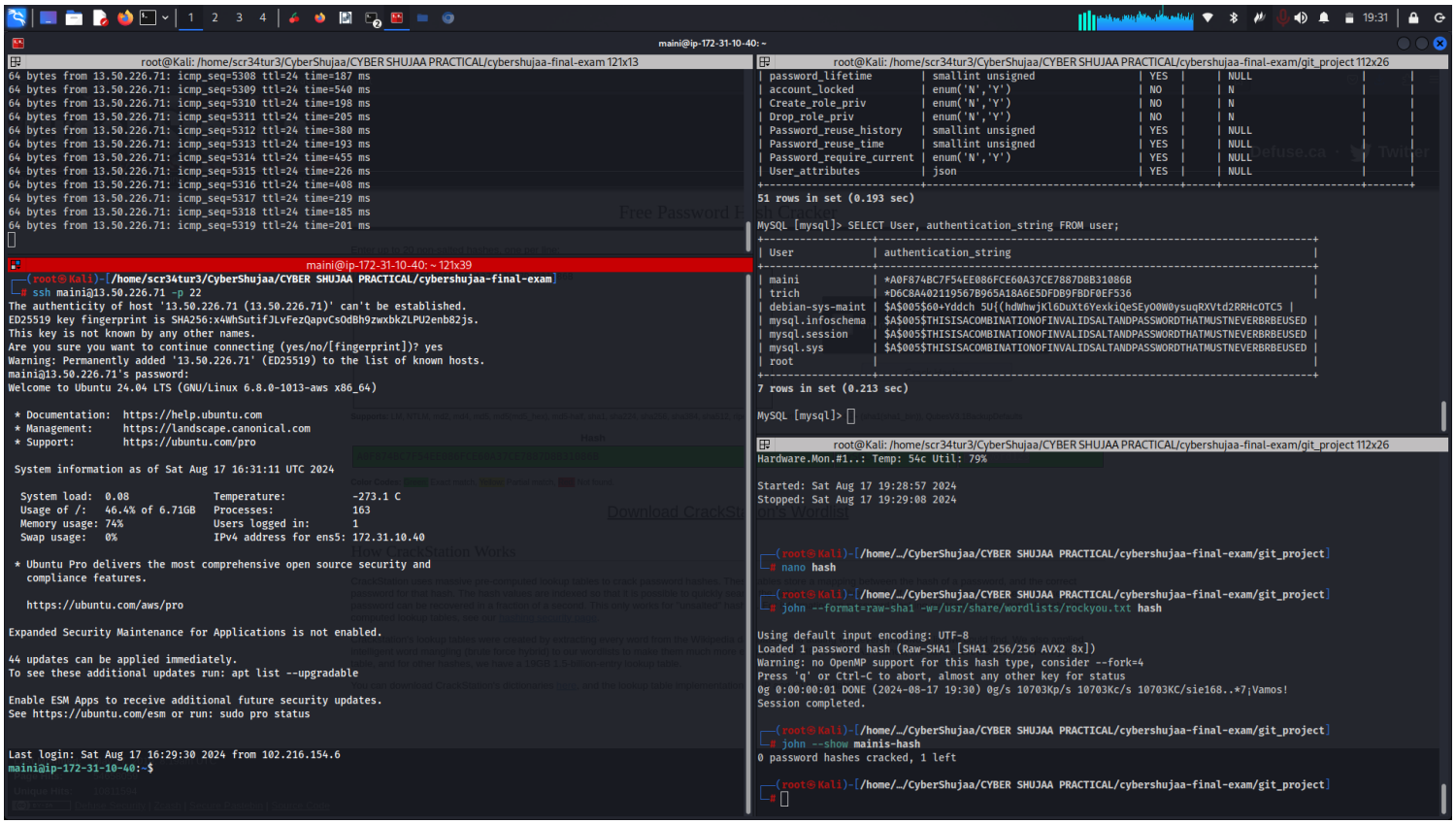
Using this tools from my local machine to crack Maini's pass, I encountered a couple of errors that after several attemt to resolve the problem, I was not able. This can be seen below.



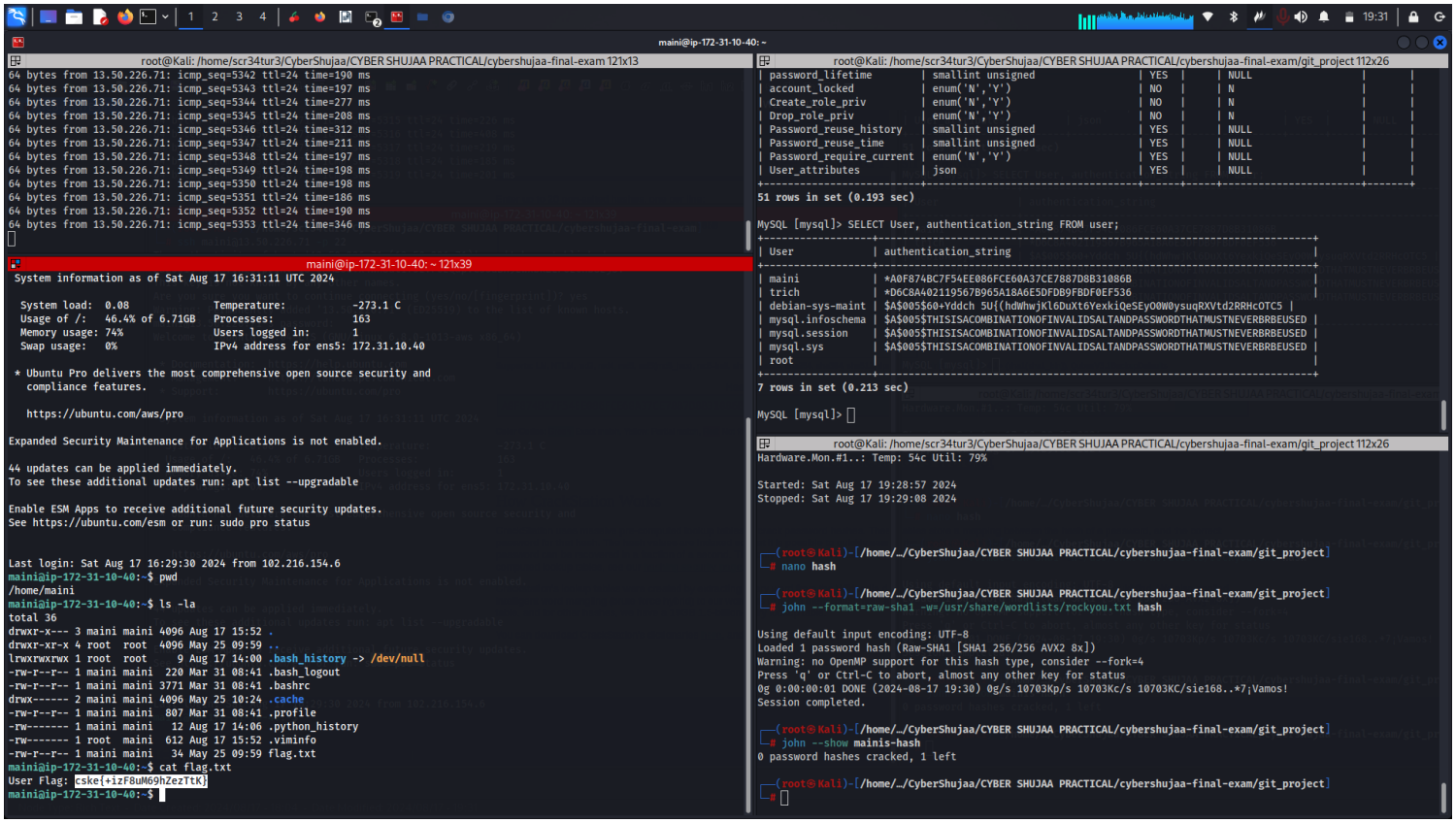
So I shifted to an online website that successfully cracked the password as seen below. The password was **"password123"**



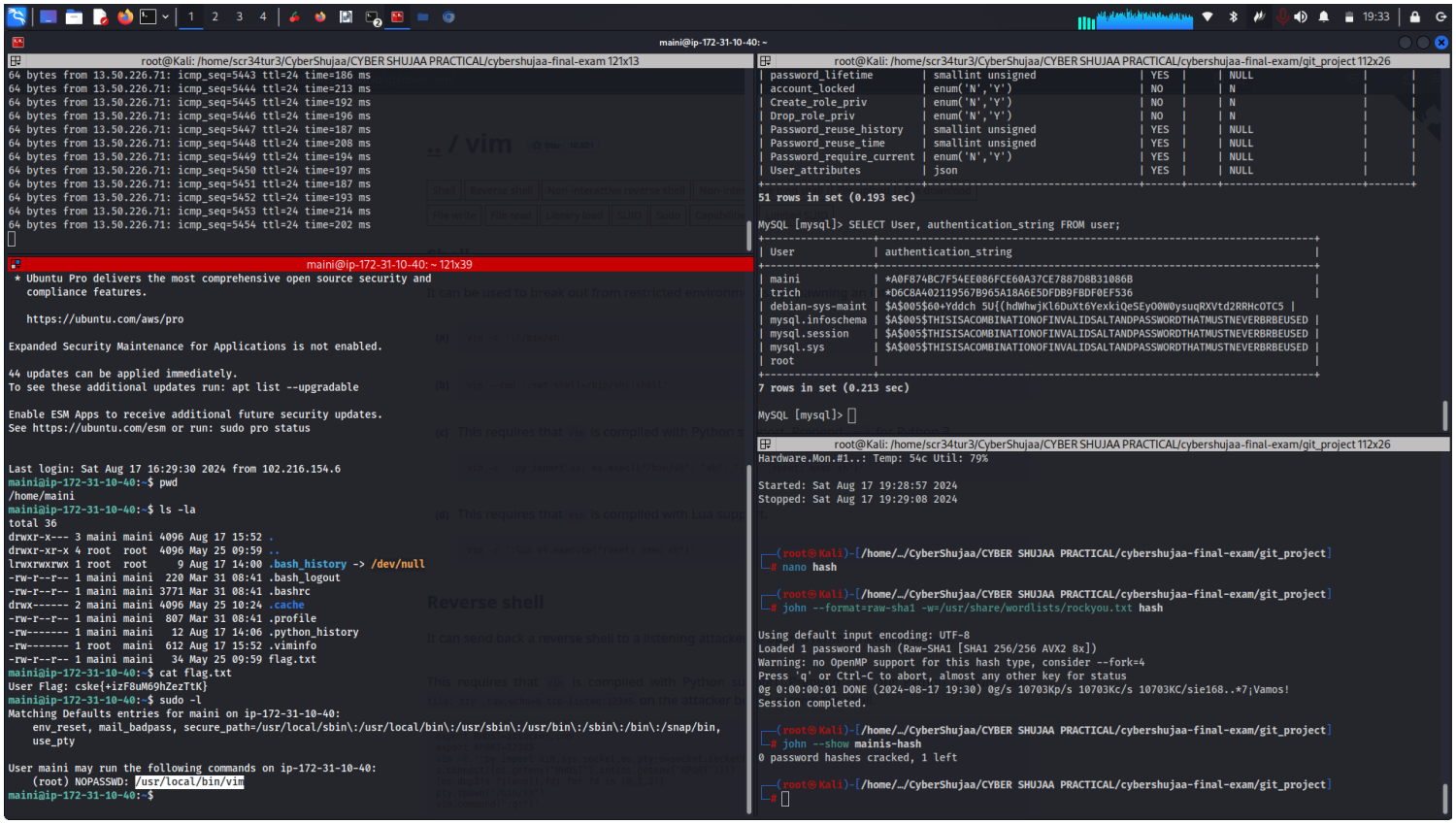
Using the cracked password belonging to user Maini, I successfully logged in to the target system via ssh as seen below.



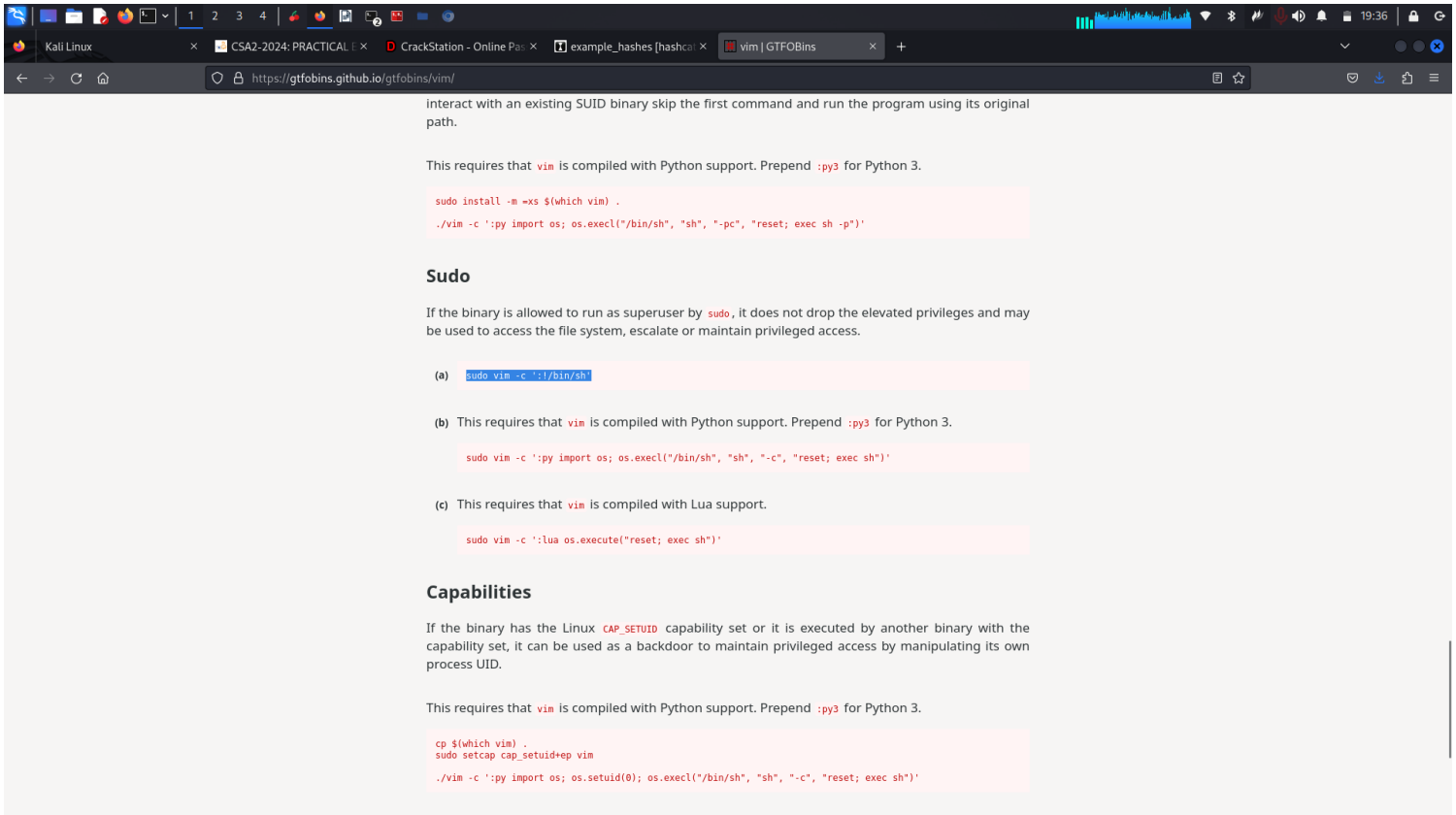
I successfully retrieve the flag under the maini's home directory as seen below.



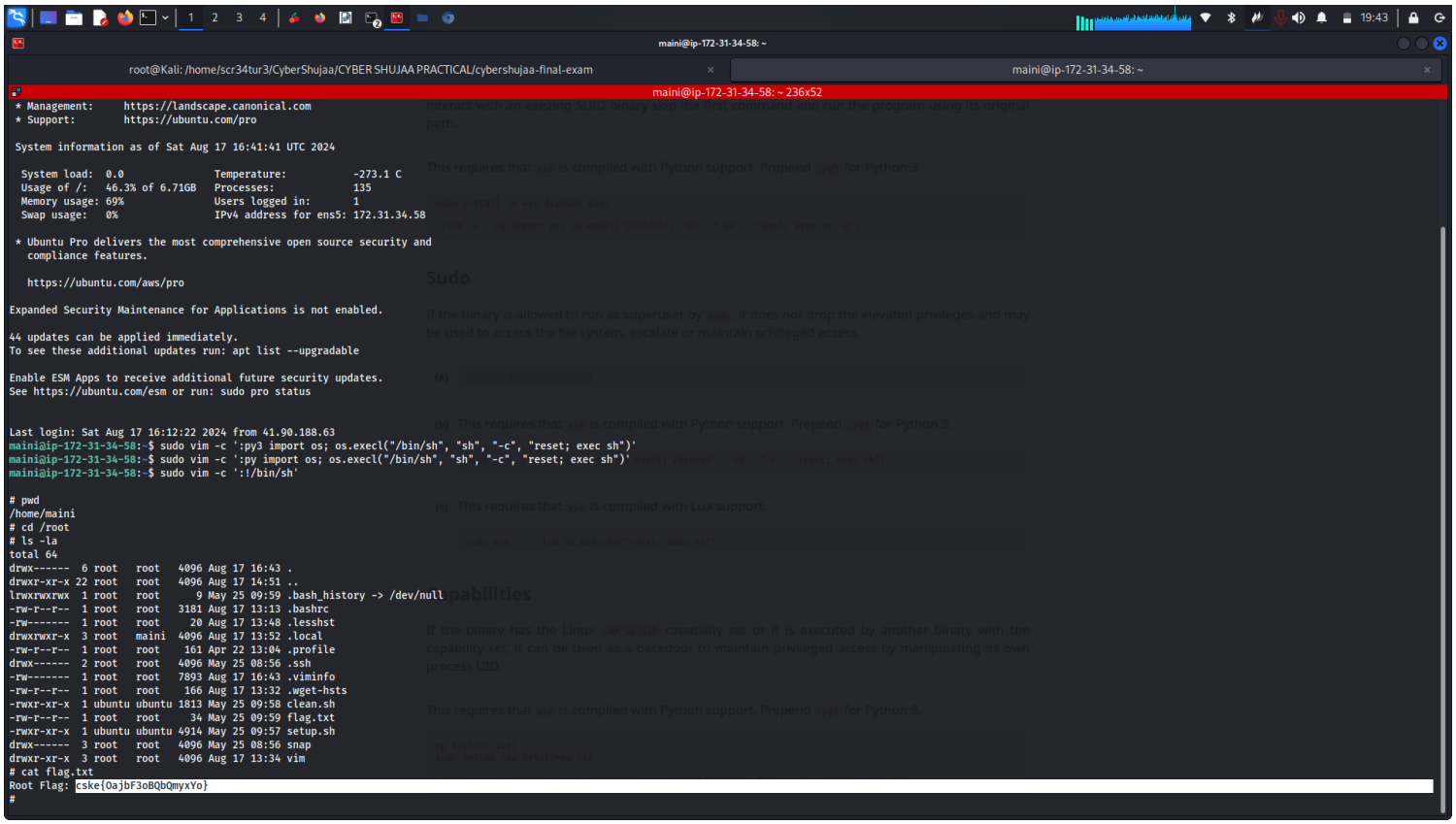
Escalating my privileges to root was not that hard. The `sudo -l` command is used to list the allowed (and forbidden) commands for the invoking user on the current host. I used this command to check which commands I can run with `sudo` as user maini and whether he needed a password to run them as shown below.



I found that he can run vim with sudo without password. Having this in mind, I was able to abuse it and gain a root shell.



I used the first payload under SUDO from the image above, and executed it. Boom! it spawned a root shell as seen below.



I was able to successfully read the content of flag.txt file under the root directory.

Penetration Test Report Conclusion

Summary of Findings:
During the penetration test against the target server, several critical vulnerabilities were identified that allowed unauthorized access to sensitive information and the elevation of privileges. Below is a summary of the findings:
1. **SMB Service: Null or Anonymous Login Allowed**
• **Impact:** Unauthorized users could connect to the SMB service without authentication, which could potentially expose sensitive files or provide an entry point for further attacks.
• **Recommendation:** Disable null or anonymous logins on the SMB service. Ensure that all connections require strong authentication mechanisms.

• **Git Repository: Plaintext Credentials Found**
◇ **Impact:** Plaintext credentials were discovered in the Git history, leading to the potential compromise of the MySQL server.
◇ **Recommendation:** Implement proper secrets management practices, such as using environment variables or secure vaults for storing sensitive information. Regularly audit the Git repository to remove sensitive data and enforce a strict code review process to catch such issues before they are committed.

• **MySQL Server: Weak and Crackable Password Hashes**
◇ **Impact:** The MySQL server was vulnerable due to weak password hashes, which were easily cracked. This allowed access to sensitive user data.
◇ **Recommendation:** Enforce a strong password policy that requires the use of complex passwords. Implement account lockout policies to mitigate brute-force attacks. Additionally, use strong hashing algorithms (e.g., bcrypt, scrypt, or Argon2) to store passwords securely.

• **Privilege Escalation: Abuse of Vim to Gain Root Access**
◇ **Impact:** It was possible to escalate privileges to root by exploiting Vim, which could be run as root without a password. This allowed complete control over the target system.
◇ **Recommendation:** Review and restrict sudoers file entries to minimize unnecessary privilege escalation paths. Regularly audit systems for potentially dangerous configurations that could lead to privilege escalation. Additionally, enforce the principle of least privilege to ensure users have only the access they need to perform their tasks.

General Recommendations:

1. **Security Audits and Hardening:** Regularly conduct security audits to identify and mitigate vulnerabilities in the system. Implement hardening procedures across all servers and services.

2. **Monitoring and Incident Response:** Establish robust monitoring to detect unauthorized access attempts and unusual activity. Ensure that incident response procedures are in place to quickly respond to security breaches.

3. **Employee Training:** Provide ongoing security awareness training for developers and system administrators to minimize the risk of introducing vulnerabilities such as hardcoded credentials and improper configurations.

4. **Patch Management:** Ensure all software, especially those exposed to the internet, is kept up to date with the latest security patches.

By addressing the vulnerabilities highlighted in this report and implementing the recommended security measures, the security posture of the target server can be significantly improved, reducing the risk of unauthorized access and potential breaches.