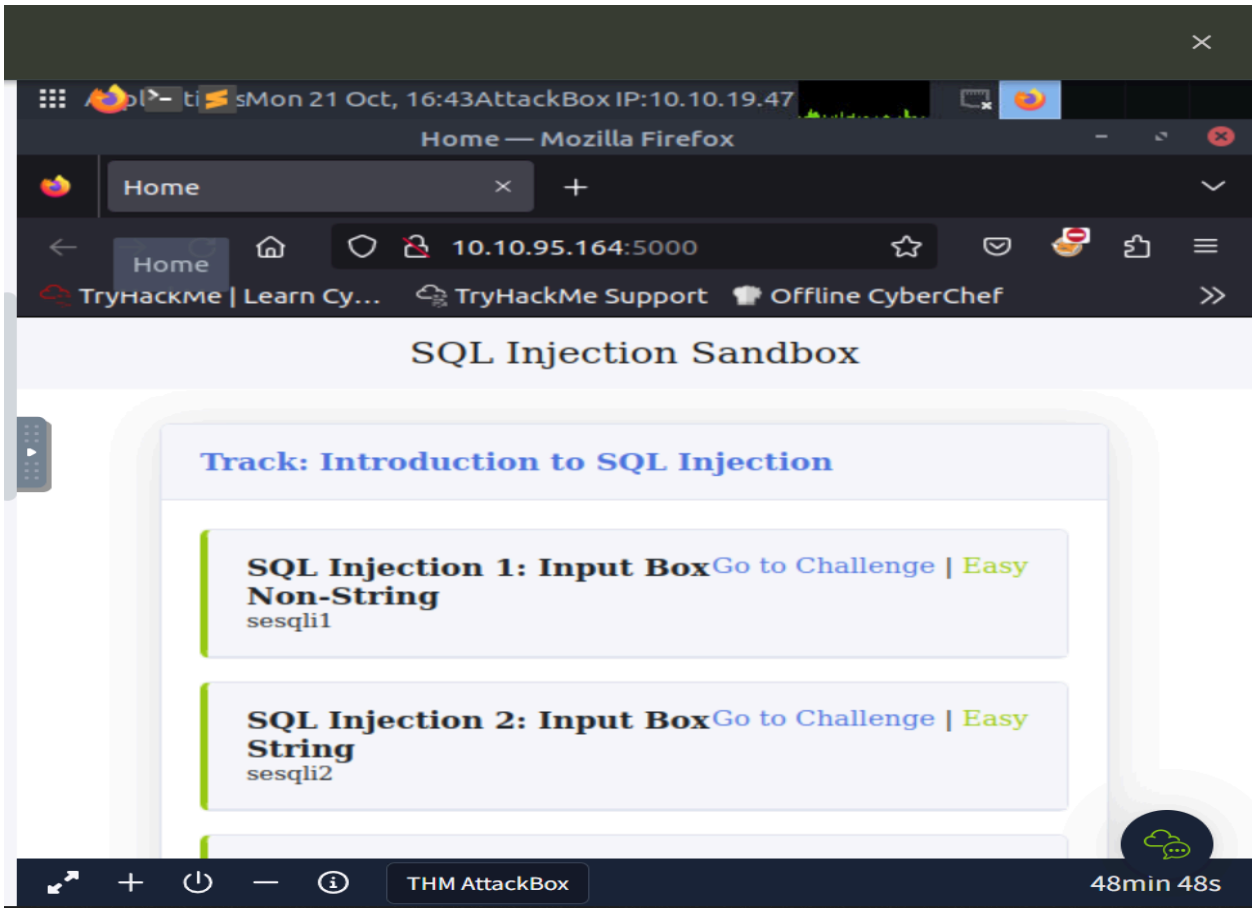
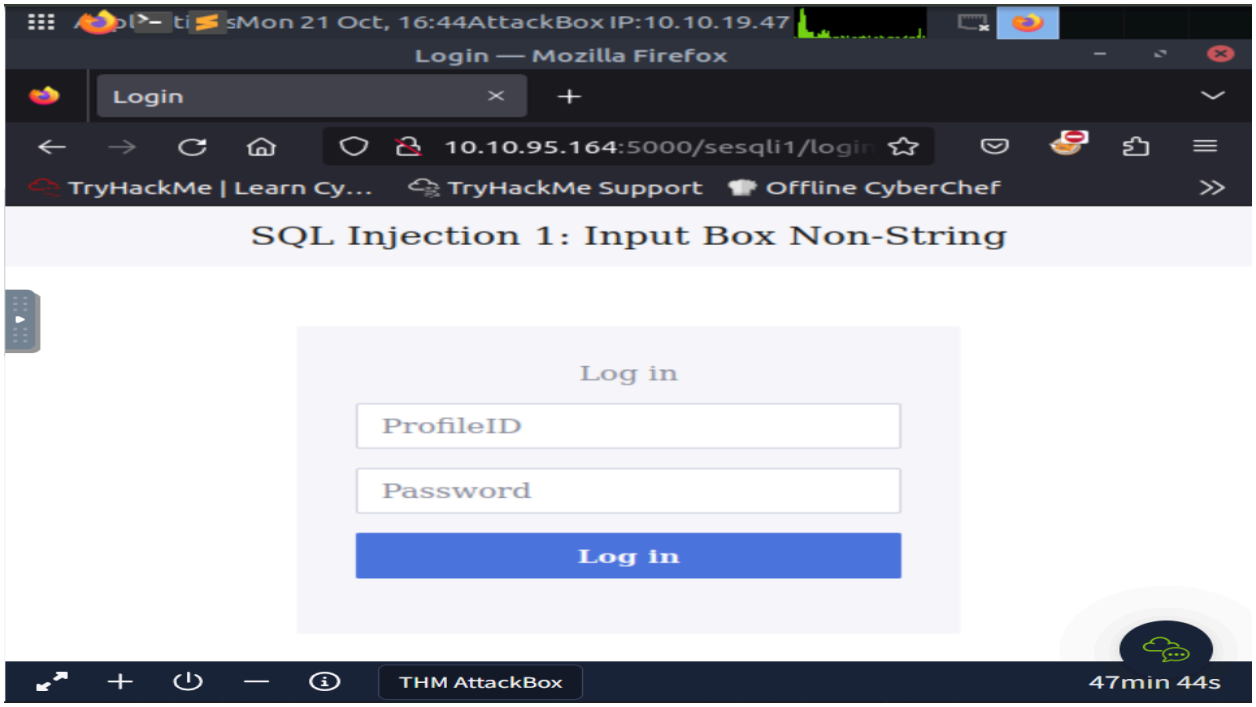
AUTOMATING SQL INJECTION USING SQLMAP.

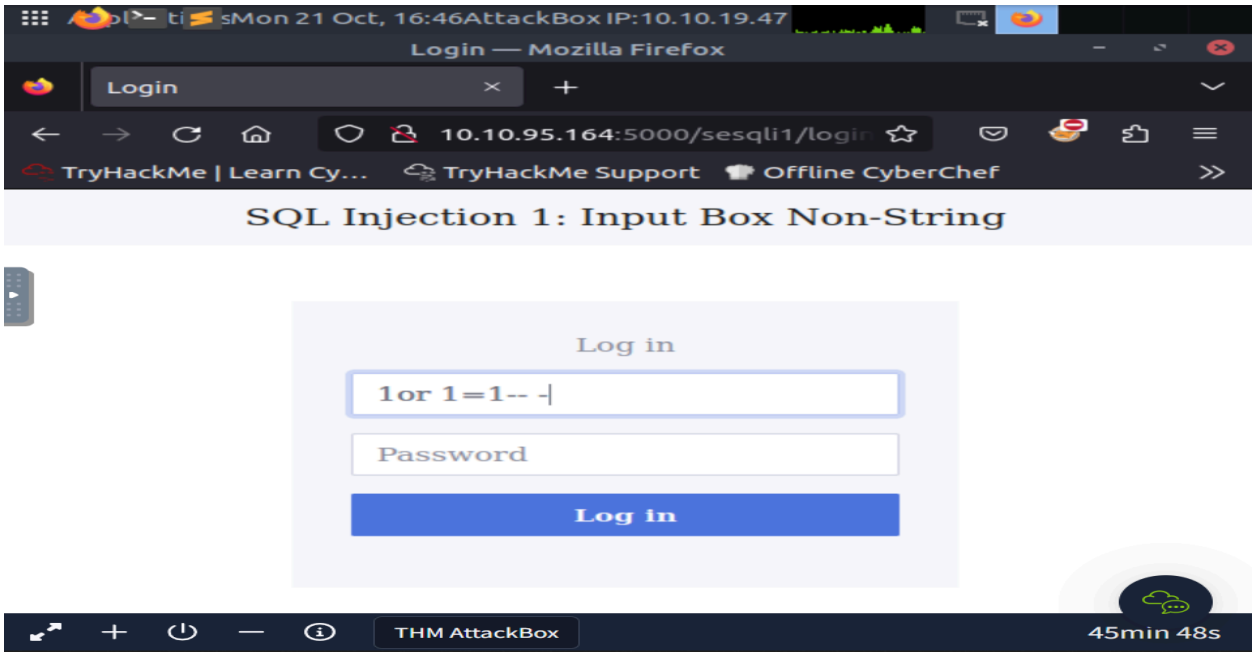
TOOLS : WEB BROWSER [TRYHACKME- ATTACK BOX]
Project-Site : sql injection sandbox
[https://tryhackme.com/room/sqlilab]

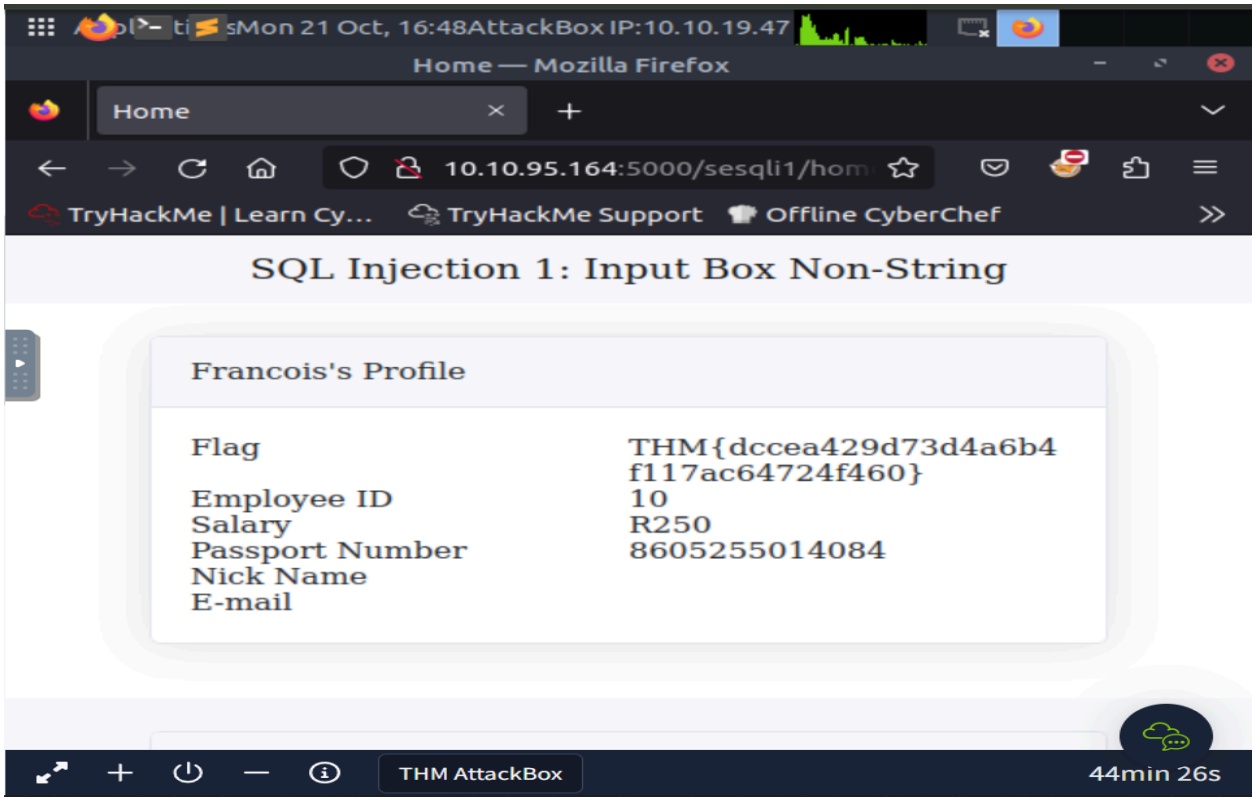
SQLmap is an open-source tool used as part of a penetration test to detect and exploit injection flaws. SQLmap is particularly useful as it saves time by automating the process of detecting and exploiting SQL injection.

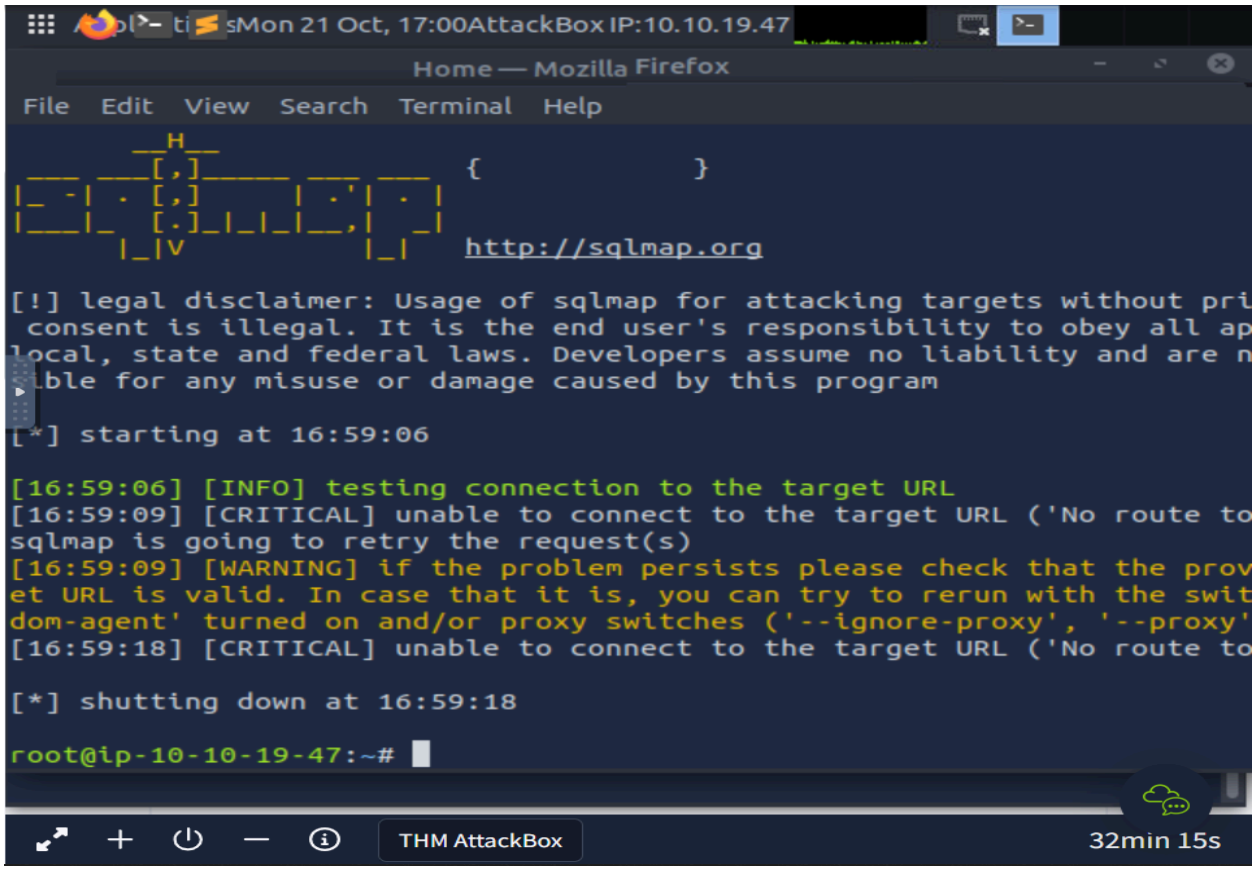
INPUT FROM ATTACKBOX :




Mon 21 Oct, 16:44
AttackBox IP:10.10.19.47
Login — Mozilla Firefox
Login
10.10.95.164:5000/sesqli1/login
TryHackMe | Learn Cy...
TryHackMe Support
Offline CyberChef
SQL Injection 1: Input Box Non-String
Log in
ProfileID
Password
Log in
THM AttackBox
47min 44s


Mon 21 Oct, 16:46
AttackBox IP:10.10.19.47
Login — Mozilla Firefox
Login
10.10.95.164:5000/sesqli1/login
TryHackMe | Learn Cy...
TryHackMe Support
Offline CyberChef
SQL Injection 1: Input Box Non-String
Log in
1or 1=1-- -
Password
Log in
THM AttackBox
45min 48s


Mon 21 Oct, 16:48AttackBox IP:10.10.19.47
Home — Mozilla Firefox
Home
10.10.95.164:5000/sesqli1/hom
TryHackMe | Learn Cy...
TryHackMe Support
Offline CyberChef
SQL Injection 1: Input Box Non-String
Francois's Profile
Flag
THM{dccea429d73d4a6b4f117ac64724f460}
Employee ID
10
Salary
R250
Passport Number
8605255014084
Nick Name
E-mail
THM AttackBox
44min 26s


Mon 21 Oct, 17:00AttackBox IP:10.10.19.47
Home — Mozilla Firefox
File Edit View Search Terminal Help
http://sqlmap.org
[!] legal disclaimer: Usage of sqlmap for attacking targets without pri
consent is illegal. It is the end user's responsibility to obey all ap
local, state and federal laws. Developers assume no liability and are n
ble for any misuse or damage caused by this program
[*] starting at 16:59:06
[16:59:06] [INFO] testing connection to the target URL
[16:59:09] [CRITICAL] unable to connect to the target URL ('No route to
sqlmap is going to retry the request(s)
[16:59:09] [WARNING] if the problem persists please check that the prov
et URL is valid. In case that it is, you can try to rerun with the swit
dom-agent' turned on and/or proxy switches ('--ignore-proxy', '--proxy'
[16:59:18] [CRITICAL] unable to connect to the target URL ('No route to
[*] shutting down at 16:59:18
root@ip-10-10-19-47:~#
THM AttackBox
32min 15s

Results and Procedure for Automating SQL Injection Using SQLMap

Steps Taken:

We began by logging into TryHackMe and navigating to the SQL Injection lab. After clicking on the blue attack box, we initiated the lab by selecting the "Start Machine" button. This led us to the attack box, where we opened a browser and entered the URL: `http://10.10.95.164:5000`. This page presented us with several labs, and we focused on the first one.

Upon clicking “Go to Challenge,” we were directed to a sample login page that is vulnerable to SQL injection. To test this vulnerability, we entered a specific payload into the "profile ID" field while inputting a random value in the password field and submitted the form. As a result, we successfully logged in (as shown in Page 4) and obtained a flag for that section.

To automate this SQL injection testing, we utilized SQLMap. We opened a terminal in the attack box and executed the following command:

sqlmap -u 'http://10.10.101.165:5000/sesqli1/login?profileID=q&password=a' -p 'profileID' --level=3 --risk=3
```

Explanation of Commands:

- **-u**: Specifies the target URL.
- **-p**: Indicates the parameter in the URL to be tested for SQL injection.
- **--level=3**: Instructs SQLMap to employ more advanced and comprehensive SQL injection techniques.
- **--risk=3**: Directs SQLMap to perform aggressive SQL injection attempts without subtlety.

For our testing purposes, we modified the command to avoid executing the attack directly. Had we run the command as is, SQLMap would have conducted a broad range of SQL injection tests against the target. Upon completion, SQLMap confirmed that the `profileID` parameter is vulnerable and provided details about the backend database version, enabling the crafting of more targeted SQL injection techniques for further exploitation.
```

Mitigation Strategies Against SQL Injection Vulnerabilities:

1. **Parameterized Queries**: Always use parameterized queries or prepared statements, which separate SQL logic from data input, making it difficult for attackers to manipulate the query.

2. **Input Validation**: Implement strict input validation to ensure that user inputs conform to expected formats. This can include whitelisting acceptable input patterns.

3. **Least Privilege Principle**: Limit database user permissions to only what is necessary for their function. For example, do not use administrative accounts for routine application database interactions.

4. **Use of ORM**: Utilize Object-Relational Mapping (ORM) frameworks, which provide an abstraction layer over raw SQL and help mitigate SQL injection risks.

5. **Regular Security Audits**: Conduct regular security assessments and code reviews to identify and remediate potential vulnerabilities proactively.

6. **Web Application Firewalls (WAF)**: Deploy a WAF to help detect and block potential SQL injection attacks before they reach your application.

By implementing these strategies, organizations can significantly reduce their risk of SQL injection vulnerabilities and enhance their overall security posture.