

Offensive Security

Penetration Test Report for Cyber Security Base - Course Project I

v.1.0

aquawhitehat+mooc@gmail.com

Table of Contents

1.0 Offensive Security Cyber Security Base Course Project I Penetration Test Report	3
1.1 Introduction	3
1.2 Objective	3
1.3 Requirements.....	4
2.0 Setting up system	4
Running application	4
3.3 Breach's disclosure.....	6
3.0 Conclusion	14

1.0 Offensive Security Cyber Security Base Course Project I

Penetration Test Report

1.1 Introduction

This report aims to disclose all breach and guide the recipient to reproduce and fix them.

1.2 Objective

Assignment is defined as follow:

“You will then write a brief (1000 words) report that outlines how the flaws can be first identified and then fixed. Your report has to be within 20% of this limit, otherwise our merciless automated robots will fail your submission. For the identification process, we suggest that you use tools that have been used in the course, such as Owasp ZAP. Once these two tasks have been completed, you will review five projects from other course participants.”

Owasp top 10 can be found here:

https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf

Default package can be found here:

<https://github.com/cybersecuritybase/cybersecuritybase-project>.

Vulnerable application can be found here:

<https://drive.google.com/open?id=1kP7w1iXlauLUddI6Ep62CEgyjPByrkxQ>

1.3 Requirements

The minimum requirements to be able to reproduce different steps are:

- Netbeans: Consider downloading the TMC package provided in module 2 set 1

http://update.testmycode.net/installers/tmc-netbeans_org_mooc/

- Owasp Zap:

https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

- Kali Linux:

<https://www.kali.org/downloads/>

- 10,000 most used passwords list:

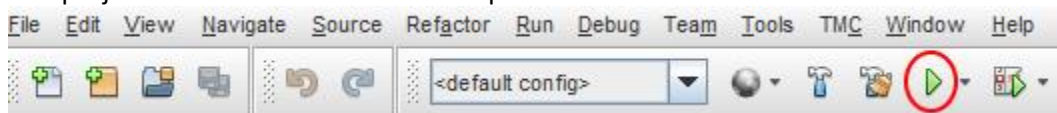
<https://drive.google.com/open?id=1N5leLptQy1GfXf4yAYfETcL9uty8C5Hv>

2.0 Setting up system

Running application

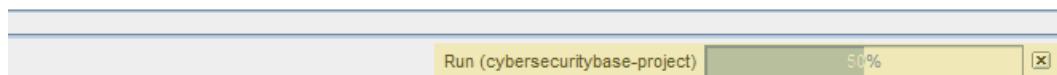
In order to run the web application.

- 1) Download the vulnerable application.
- 2) Decompress it in your project folder
- 3) In Netbeans: File> Open Project / Ctrl + Shift + O
- 4) Browse to the project folder location, select the vulnerable application project, and click open Project
- 5) Run project: F6 or Green arrow at the top of Netbeans IDE:

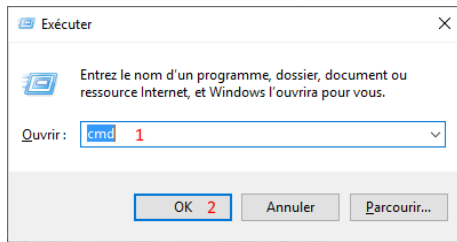


- 6) If everything goes well, the application will load, check output console:

Started CyberSecurityBaseProjectApplication in 6.048 seconds (JVM running for 6.741)



- 7) By default application is loaded to localhost:8090, or [serverIP]:8090
- 8) In order to know [serverIP], type "Windows+R"
- 9) Execute "cmd":



10) In command prompt, type “ipconfig /all”

11) Record your IP address, it will be used for Kali attack:

```
Carte Ethernet Ethernet 2 :

Suffixe DNS propre à la connexion. . . :
Description. . . . . : Realtek PCIe GbE Family Controller
Adresse physique . . . . . :
DHCP activé. . . . . : Oui
Configuration automatique activée. . . : Oui
Adresse IPv6 de liaison locale. . . . : 1
Adresse IPv4. . . . . : 192.168.0.15(préfér  )
Masque de sous-r  seau. . . . . : 255.255.255.0
Bail obtenu. . . . . : lundi 17 d  cembre 2018 07:39:26
Bail expirant. . . . . : jeudi 27 d  cembre 2018 18:49:10
Passerelle par d  faut. . . . . : 192.168.0.254
Serveur DHCP . . . . . : 192.168.0.254
IAID DHCPv6 . . . . . :
DUID de client DHCPv6. . . . . :
Serveurs DNS. . . . . :
NetBIOS sur Tcpip. . . . . : Activ  
```

12) Here, the external web application will be accessible by browsing to: 192.168.0.15:8090

3.3 Breach's disclosure

Vulnerability Exploited: [A2:2017-Broken Authentication & Insufficient Logging & Monitoring](#)

System Vulnerable: http://localhost:8090

Vulnerability Explanation:

Hackers have access to hundreds of thousands of passwords, they can use them to brute force any logging page is no monitoring is set up

Step 1: Launch Kali

Step 2: Start a terminal

Step 3: Enter command:

```
hydra -l Customer -P /root/Downloads/passwords.txt -t 16 [your_server_IP] http-post-form  
"/login:username=^USER^&password=^PASS^&Login:Invalid username and password." -V -s 8090
```

[server_ip] is for me replaced by: 192.168.0.15.

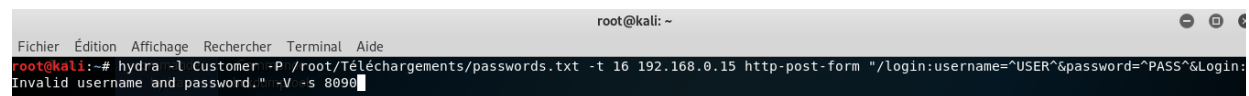
Step 4: Wait for hydra to bruteforce password of Customer account.

Vulnerability Fix:

- 1- Forbid user to take a password in a list of passwords
- 2- Activate CSRF to make it more complicated to brute force
- 3- Monitor logging, if you detect a lot of fail, block the account, this may seem it is being attacked.

Severity: **Critical**

Proof of Concept Code Here:



The screenshot shows a terminal window titled 'root@kali: ~'. The command entered is: `hydra -l Customer -P /root/Téléchargements/passwords.txt -t 16 192.168.0.15 http-post-form "/login:username=^USER^&password=^PASS^&Login:Invalid username and password." -V -s 8090`. The output shows the command being executed and the server responding with 'Invalid username and password.'

```
root@kali: ~
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "abc123" - 14 of 10001 [child 13] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "mustang" - 15 of 10001 [child 14] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "michael" - 16 of 10001 [child 15] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "shadow" - 17 of 10001 [child 3] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "master" - 18 of 10001 [child 0] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "jennifer" - 19 of 10001 [child 4] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "111111" - 20 of 10001 [child 7] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "2000" - 21 of 10001 [child 1] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "jordan" - 22 of 10001 [child 2] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "superman" - 23 of 10001 [child 5] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "harley" - 24 of 10001 [child 8] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "1234567" - 25 of 10001 [child 6] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "vacation" - 26 of 10001 [child 15] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "fuckme" - 27 of 10001 [child 13] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "hunter" - 28 of 10001 [child 11] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "fuckyou" - 29 of 10001 [child 9] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "trustno1" - 30 of 10001 [child 10] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "ranger" - 31 of 10001 [child 12] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "buster" - 32 of 10001 [child 14] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "thomas" - 33 of 10001 [child 3] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "tigger" - 34 of 10001 [child 0] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "robert" - 35 of 10001 [child 4] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "soccer" - 36 of 10001 [child 7] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "fuck" - 37 of 10001 [child 1] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "batman" - 38 of 10001 [child 2] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "test" - 39 of 10001 [child 5] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "pass" - 40 of 10001 [child 8] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "killer" - 41 of 10001 [child 6] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "hockey" - 42 of 10001 [child 13] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "george" - 43 of 10001 [child 11] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "charlie" - 44 of 10001 [child 9] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "andrew" - 45 of 10001 [child 10] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "michelle" - 46 of 10001 [child 14] (0/0)
[ATTEMPT] target 192.168.0.15 - login "Customer" - pass "love" - 47 of 10001 [child 12] (0/0)
[8090][http-post-form] host: 192.168.0.15 login: Customer password: vacation
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2018-12-17 19:46:47
```

Note: Other software can be used, such as Owasp Zap, by using fuzzing function.

Vulnerability Exploited: A6:2017-Security Misconfiguration

System Vulnerable: <http://localhost:8090/login>

Vulnerability Explanation:

System almost always have default logging username and password, for a large part of network device, pirate have access to this list online, some tools even load them by default, this can be used to access system easily.

Step 1: Browse to <http://localhost:8090/login>

Step 2: Log with username = admin and password = admin

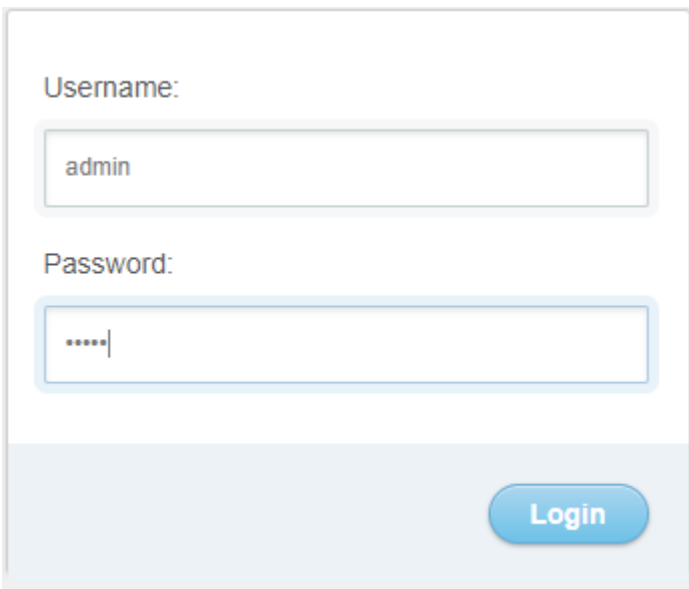
You have gained admin access

Vulnerability Fix:

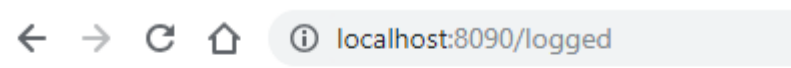
- 1- Before releasing the application, delete default password, generate a random one that you'll provide with application.
- 2- When setting up a system, always check no default logging are still active

Severity: Critical

Proof of Concept Code Here:



A screenshot of a web application's login interface. It features a light gray background with a white login box. Inside the box, there is a 'Username:' label followed by a text input field containing the word 'admin'. Below this is a 'Password:' label followed by a password input field with masked characters (dots). At the bottom right of the login box is a blue 'Login' button.



Your are inside the matrix!!

Vulnerability Exploited: [A5:2017-Broken Access Control](#)

System Vulnerable: <http://localhost:8090/login>

Vulnerability Explanation:

User right needs to be define very clearly, it's important that user can only access user function, and only admin have access to admin function. In this case, user and admin have the same right because roles are not well defined and used.

Step 1: Browse to <http://localhost:8090/login> or click login button on first page.

Step 2: Login as admin as seen before.

Step 3: Go back to <http://localhost:8090/login>, login with username: Customer and password: vacation

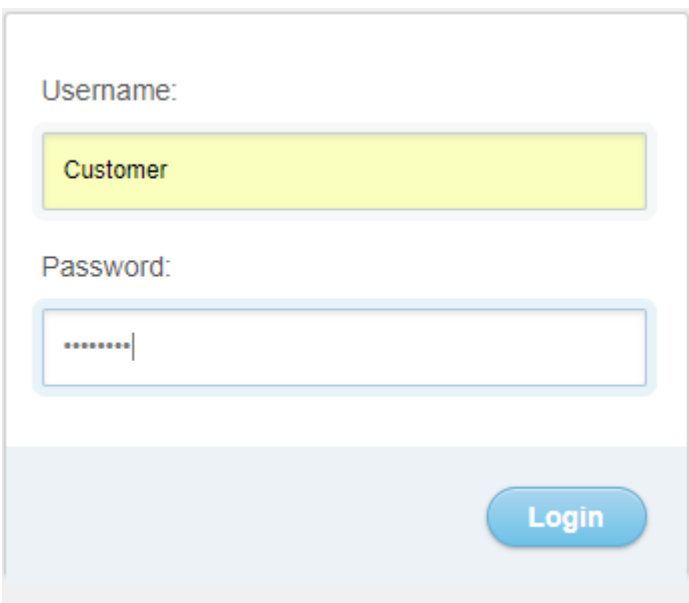
Step 4: Admin and Users have the same rights! Both can see all user password and other data

Vulnerability Fix:

Add roles handling to users, in order to do so go to `MySimpleUrlAuthenticationSuccessHandler` > `determineTargetUrl`, uncomment line 60 to 81, and use proper role, then block access to page when unwanted.

Severity: **Critical**

Proof of Concept Code Here:



A screenshot of a web application's login page. It features a light blue header area. Below the header, there is a form with two input fields: 'Username:' and 'Password:'. The 'Username:' field contains the text 'Customer'. The 'Password:' field is masked with dots. At the bottom right of the form, there is a blue 'Login' button. The entire form is enclosed in a light gray border.

Your are inside the matrix!!

[Return](#)

Already signed up (don't look at password and role!)

Name of our wonderful customer	Address (not supposed to disclose, please ignore)	Secret password	Role
Pigeon 1	Rob him	pass1	User
UberCustomer	At home	I'm the boss	Admin
Customer	In10K	vacation	User

Vulnerability Exploited: [A7:2017-Cross-Site Scripting \(XSS\)](#)

System Vulnerable: <http://localhost:8090/>

Vulnerability Explanation:

XSS is the ability to inject javascript into the webapplication, this is possible if the user input is not validated, in order to avoid it, it's important perform user input sanitization and validation.

Step 1: Browse to <http://localhost:8090/>

Step 2: Fill each area,

.Name: Any name

.Address: `<script>alert('XSS Exploit worked');</script>`

.Password: any password

Step 3: Browse to <http://localhost:8090/login> and log with username: Customer and password: vacation.

Step 4: Submit, and look at happen when you come back to this page.

A pop up will show up, with the message "XSS Exploit worked".

Vulnerability Fix:

- 1- In the table code, replace utext by text, this will consider whatever is written as text, and not anything else, but it's not enough
- 2- Set up input validation, escape the '<' and '>' and also their equivalent in code, and remove keyword like 'script' or any html/php/javascript tag..

Severity: **Critical**

Proof of Concept Code Here:

Sign up to the event using this form

Name:

Address:

Password:

← → × ⌂ ⓘ localhost:8090/logged

Your are inside the matrix!!

Already signed up (don't look at password and role!)

Name of our wonderful customer	Address (not supposed to disclose, please ignore)	Secret password	Role
Pigeon 1	Rob him	pass1	User
UberCustomer	At home	I'm the boss	Admin
Customer	In10K	vacation	User
Aquadrox			

localhost:8090 says
XSS Exploit worked

Vulnerability Exploited: [A3:2017-Sensitive Data Exposure](#)

System Vulnerable: http://localhost:8090/login

Vulnerability Explanation:

If data is not well encrypted, a pirate will be able to use man in the middle attack to sniff all your data, gain access to your account, and steal your identity.

Step 1: Launch Owasp Zap.

Step 2: Launch a sniffed browser

Step 3: Go to http://localhost:8090/login

Step 4: Login using for example username=Customer and password=vacation

Step 5: Go back to Owasp Zap and check what you see:

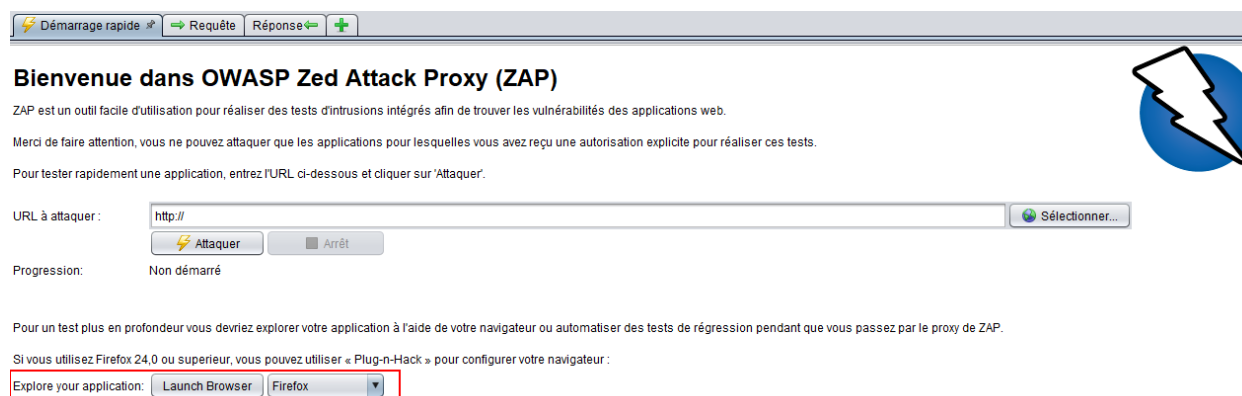
username=Customer&password=vacation

Step 6: So anyone can sniff username and password because nothing is encrypted.

Vulnerability Fix: Implement https protocol to encrypt every transfer. And activate csrf to make pirate work more complicated.

Severity: **Critical**

Proof of Concept Code Here:



3.0 Conclusion

I tried to implement SQL injection flaw, but I was unable to set up an embedded SQL server to my project, I found several but it takes a lot of time, I'll try to submit another version of this project later and I do want to add it.

SQL injection is also a very powerful attack, the pirate can really get access to a lot of sensitive data, and once he is inside it's very hard to spot him.

It's actually also very easy to block it has we seen before.

Designing this application, I realized something, it's usually very simple to avoid all these breaches, it's just a matter of taking time to patch the flow, and close the door.

But we are lazy and want to go faster so we don't take the necessary time to implement all of these.

In addition, in huge team, maybe we can forget to speak together, and then create a breach, I think it's very important to add a pentester in all dev team.

On flaw may be quite tricky, the out of date dependencies, even if you fix it when releasing the application, it will happen later for sure, one has to think about maintaining all application and versioning in order to avoid it to happen. I think this may become one of the worse on internet on the years to come.