3.1 Task 1: Get Familiar with SQL Statements

- print all the profile information of Alice

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

3.2 Task 2: SQL Injection Attack on SELECT Statement

Task 2.1 : SQL Injection Attack from webpage.

* Log in as Admin to see all members’ information
* Typed: ‘ or Name=’admin’;#

Graphical user interface, application

Description automatically generated

Table

Description automatically generated

Task 2.2 SQL Injection Attack from command line

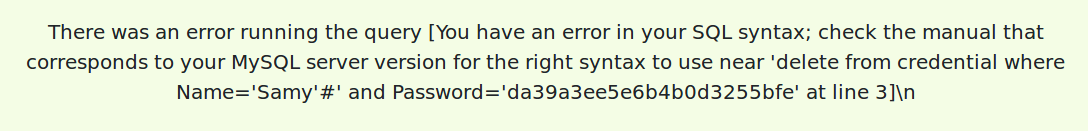
* Edited given getdata.php to use the database(Users) and table(credential) for this lab.
* Edited version was named as “**getdata3.php**,” which name has been edited to “**task2-2.php**” for lab submission.
* Send HTTP request using curl command
* Used HTTP Header Live to know what to use to encode the special character “#,” which is %23
* Typed: Curl ‘www.seedlabsqlinjection.com/getdata3.php?EID=%27%20OR%201=1%20%23&Password=’
* Result: could see all the information (note that salaries were changed as I proceeded beforehand for some practice)

Text

Description automatically generated

Task 2.3 : Append a new SQL statement.

* Try to delete a record from the database. I chose to delete Samy’s record.
* Typed : a ‘;delete from credential where Name=’Samy’#
* Failed to attack against MySQL because mysqli::query() API in PHP’s mysqli extension does not allow multiple queries to run in the database server.
* Another concern: assume that it is allowed but what if the attacker does not know the table name?



Note that we can use $mysqli->multi\_query() instead of $mysqli->query() to defeat this case.

3.3 : SQL Injection Attack on UPDATE Statement

Task 3.1 : Modify your own salary. We assume we are Alice.

* Typed : ‘, salary=’100000’ where EID=’10000’;#

Graphical user interface, application

Description automatically generated

Table

Description automatically generated

Task 3.2 : Modify other people’ salary. We want to modify Boby’s salary to 1

* Typed : ‘, salary=1 where Name=’boby’ #

Graphical user interface, application

Description automatically generated

Table

Description automatically generated

Task 3.3 : Modify other people’ password

* Modify Boby’s password to something we(Alice) know.
* Assume, as Alice, that we know the hash value of our own password, which generated via SHA1 hash function.
* Copy the hash value of Alice’s password (seedalice) we got from task 1: fdbe918bdae83000aa54747fc95fe0470fff4976
* Typed : ', password='fdbe918bdae83000aa54747fc95fe0470fff4976' where EID='20000';#

Graphical user interface, application

Description automatically generated

* Log in as Boby with password “seedalice” instead of “seedboby”
* Result : was able to log in to Boby’s account

Table

Description automatically generated

3.4 Task 4: Countermeasure — Prepared Statement- fix the SQL injection vulnerability exploited in, as my choice, task 2.2 ( SQL Injection Attack from command line in order to get admin’s privilege, which can get all members’ information)

- made change to original php (.php) to get data(name, salary, snn for all members) in order to use prepared statements that separate code and data.

- it was saved as “**getdata\_prepare4.php**,” which will be submitted as “**task4.php**.”

- Result: table for all information did not show up.

Text

Description automatically generated