INCS-745 Intrusion Detection and Hacker Exploits: Assignment 4

Hedao Tian
1256221

Task 1: Get Familiar with SQL Statements
There is an existing database(Users) containing a table named credential. When logging in as root and use database Users, I could show credential tables and uses "SELECT * FROM credential WHERE Name = 'Alice'".
Task 2: SQL Injection Attack on SELECT Statement
Task 2.1: SQL Injection Attack from webpage.
By typing 'or Name='admin';#. I could modify the query to "SELECT * FROM credential WHERE name= ''or Name='admin';#" This makes the sql query returns everything from admin without checking the password.
Task 2.2: SQL Injection Attack from command line.
Since we push name and password values from html form EID to the unsafe_home.php file. We hardcode the html request to
'http://www,seedlabsqlinjection.com/unsafe_home.php?username=%27or+Name%3D%27admin%27%3B%23&Password=''.
We get the original HTML code comes back to the terminal.
Task 2.3: Append a new SQL statement.
I tried to delete record of Ted by typing 'or 1=1;DELETE FROM credential WHERE Name='Ted'#
However, the attack failed and I found that the execution of two command is banned from accessing MySQL when PHP is involved.
Task 3.1: Modify your own salary.
Since the edit profile page is not linked to the main page, I couldn't modify it from webpage. But I could inject code to unsafe_edit_backend.pnp through keyword 'NickName'(I checked the file and found this variable contains the user input from the profile edit page's first line). So I input 'www.seedlabsqlinjection.com/unsafe_edit_backend.php?NickName=%27%2Csalary%3D%27123456%27%20WHERE%20Name%3D%27Alice%27%3B%23' to the http request to the file. And I could see the salary is changed.
Task 3.2: Modify other people' salary.
To change Boby's salary, just change some keywords as below:
'www.seedlabsqlinjection.com/unsafe_edit_backend.php?NickName=%27%2Csalary%3D%271%27%20WHERE%20Name%3D%27Boby%27%3B%23'
Task 3.3: Modify other people' password.
Modifying Boby's password to 123, hashed to:
40BD001563085FC35165329EA1FF5C5ECBDBBEEF
'www.seedlabsqlinjection.com/unsafe_edit_backend.php?NickName=%27%2CPassword%3D%2740BD001563085FC35165329EA1FF5C5ECBDBBEEF%27%20WHERE%20Name%3D%27Boby%27%3B%23'
Task 4: Countermeasure — Prepared Statement
Since the prepared statement code was already injected into 'safe_home.php'. We just need to change the form posting direction from unsafe_home.php to it and restart the server by "sudo service apache2 restart". And then the same attack from task 2 cannot be achieved. Because

the SQL string was compiled first without knowing the data and then the data was binded right before execution of the string. Thus the malicious code was not injected during the compilation.