**BLG 460E SECURE PROGRAMMING**

HW4

**Ahmet Göktuğ SEVİNÇ**

**150140120**

**13.05.2019**

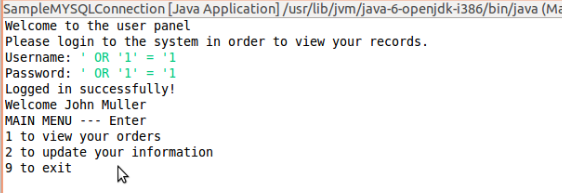
**Q1)**

**-Can you login to the system without knowing any username and any password?**

If we enter username and password as: **‘ OR ‘1’ = ‘1**

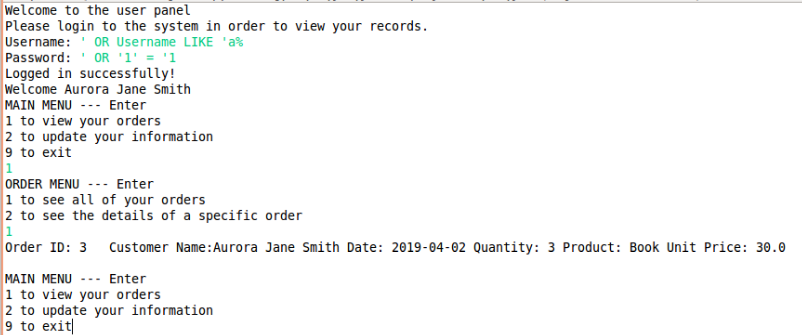
Query becomes: **WHERE Username = ‘’ OR ‘1=1’ AND Password = ‘’ OR ‘1=1’**

So we can login the system.



**-Can you obtain any information about users (e.g., names, orders, details)?**

From the above query we already have access to the last user who is John Muller, other than that, we can get information about users whom usernames starts with specific strings. For example, if we use; **‘ OR Username LIKE ‘a%** statement, we can get person whom username starts with ‘a’ latter. After that we can view his/her orders and details.



**-By means of SQL injections, is it possible to add new entries to the tables or remove any entries?**

For data manipulation operations we need executeUpdate() function, so we can use updateUserInformation() function after we login. We can execute multiple queries to perform these operations. For example, for the phone number, if we use;

*' WHERE Username='"+username+"'; DELETE FROM UserTable WHERE Username='"+username+"'; --*

We can delete all of the entries from UserTable. Or if we use;

*' WHERE Username='"+username+"'; INSERT INTO UserTable (Name, Password, CustomerID) VALUES (‘deneme’, ‘12345’, 9); --*

We can insert an entry to the UserTable

**-What are the possible countermeasures against injection attacks. Give at least two types of countermeasures. Explain how they work. Provide implementations in the report.**

1 - We can use parametrized queries. By this way, parameters will be supplied at execution time and whole statement will be a parameter instead of being a seperate query. For example, for the username and password operations we can use;

*String selectTableSQL = "SELECT \* from UserTable WHERE Username=? AND Password=?";*

*PreparedStatement prepStmt = conn.prepareStatement(selectTableSQL);*

*prepStmt.setString(1, t\_username);*

*prepStmt.setString(2, t\_password);*

*ResultSet rs = prepStmt.executeQuery();*

2 – We can try to build a regex for sql queries. This might not prevent all of the attacks completely but it works against simple attacks. For example, we can write such function;

public String MysqlRealScapeString(String str){

String data = null;

if (str != null && str.length() > 0) {

str = str.replace("\\", "\\\\");

str = str.replace("'", "\\'");

str = str.replace("\0", "\\0");

str = str.replace("\n", "\\n");

str = str.replace("\r", "\\r");

str = str.replace("\"", "\\\"");

str = str.replace("\\x1a", "\\Z");

data = str;

}

return data;

*taken from:* [*https://stackoverflow.com/questions/1812891/java-escape-string-to-prevent-sql-injection*](https://stackoverflow.com/questions/1812891/java-escape-string-to-prevent-sql-injection)

So, with that function, attacks will be prevented.

**Q2)**

**2.1)**

**- What information can be gathered with an HTTP GET request?**

GET request can be cached, so with get request important information such as username and password can be gathered.

**- What do the buttons do?**

Buttons sends information to the attacker servers.

**- What are the diferences between the outputs in the terminals of both files?**

In sdattacker’s terminal cookie information is obtained, too. Also, hosts are different obviously.

**2.2)**

**-Which JavaScript function allows the cookie display?**

listCookies().

**- How can you prevent JavaScript to reach the cookies?**

We can make cookies httponly. So that javascript can not reach them.

**2.4) Expired Cookies**

expires = datetime.datetime(2019, 5, 13, 23, 32, 00) + datetime.timedelta(seconds=30)