Eddie Goynes
HW2 Network Security
931719883

ECE 478 Network Security: Homework #2

Disclamer

This submission reflects my own understanding of the homework and solutions. All of the ideas are my own, unless I explicitly acknowledge otherwise.

1. From looking at the source code, what can you determine about the schema of the database?

   Reading over the code it looks like you are fetching rows at a time, which seem to contain a username and password. The code makes me think that the table is just two rows, one of usernames and the other one of passwords. I do not know how many columns there are based on the code. There could be hidden rows that I cannot see it the code, if they are not referenced. The table could have a unique ID for each element for instance.

2. Looking at the source code, identify and describe exactly why the code is vulnerable to SQL injection.

   In the code where you have $U = param("u"); and the same for the password you are just pasting the text that the user typed into the form boxes. These variables have not had any special characters removed like the $ or <. Then a few lines later when you access the database, "select * from users where uname='$U'. A user is able to place a single quote ' to escape your SQL command making it so they can inject what ever they write after the single quote into the SQL database query statement.

3. Does any information from the database get included in the visible output (if so, where)?

   When there is a successful login there is a message printed out to the browser. "msg "Login successful. Welcome, $first_column";" . The actual data printed from the database looks like the row that is selected by fetch row, where the username and password are correct. This is the only place I see in the code that anything is printed out, so I will have to use this print out to extract all the data I want to retrieve from this database. It's called first column so I suspect that I will only be able to retrieve one item at a time when doing the SQL injection attacks.

4. Describe an SQL injection attack that allows you to determine how many columns are in the users table.
   Select

   As the hint states to use the UNION operator I first looked at the W3schools page to see how to use the UNION operation. It allows me to do another SELECT statement in the SQL statement. I use the ‘ single quote to escape out of the uname=’’ therefore stopping that part of the statement and allowing me to put any SQL statement. The UNION lets me do another SELECT, then the – comments out the rest of the statement. The “FROM users” gets from the users table. The 1,1 will return 1’s for the columns in the table. Basically if I try to have it return 1’s for less, or more columns than the table has I will get and error.

   I first try this since I think I have 2 columns.

   ' UNION select 1,1 FROM users --
   Info: Can't prepare SQL statement: SELECTs to the left and right of UNION do not have the same number of result columns

   Get this result in error, so I try to see if there are three columns.

   Then I was trying 3 columns.

   ' UNION select 1,1,1 FROM users --
   Info: Login successful. Welcome, 1

   Looks like it works, this was an easy way to see that there are 3 columns.

   Now trying 4 columns I get the same error. So the table must have 3 columns.

   ' UNION SELECT 1,1,1,1 FROM users --
   Info: Can't prepare SQL statement: SELECTs to the left and right of UNION do not have the same number of result columns

5. Although we already know that the table is named users, describe an attack by which we could have used SQL injection to find out the name of the table without having access to the source code of the script (but knowing that it used SQLite).

   Since you said that all SQL_Lite tables have a sqlite_master table I use the FROM operator.

   About 75% way through this page it says that name is the name of the SQL table.
   http://www.sqlite.org/fileformat2.html
   I need the extra ones so I do not get an error trying to return too few columns.
   At the end it will print out the name now.

   ' UNION SELECT name,1,1 FROM sqlite_master --

   Info: Login successful. Welcome, users

6. Describe an attack that allows you to learn the SQL schema of the users table. (table schema= SQL statement used to create the table)

   This code gives me the Schema. I used the exact code that I used in problem five, but after reading this site I saw that the SQL element of the sqlite_master table holds the schema.
   http://www.sqlite.org/fileformat2.html

   Command used.
   ' UNION SELECT sql,1,1 FROM sqlite_master --

   Info: Login successful. Welcome, CREATE TABLE users (uname varchar(10), pwd varchar(20), realname varchar(100))

7. Describe an attack that allows you to learn the entire contents of the users table!
   Hint: The script executes an SQL statement and only looks at a single row of the result. So you'll have to extract the database piece-by-piece.

I can use the following code to extract the first column out of the users table.

Using the same general form as the previous problems I was able to look at the hint that you give us and come up with this code. X is the schema name of what I want returned.

I started out with this command, which returned admin, the first user name.

' UNION SELECT uname,1,1 FROM users --

I then was able to use this command to get the next username.

' UNION SELECT MIN(uname),1,1 FROM users WHERE uname > "admin" --

Replacing admin with Arnold allowed me to get the next username dave.

' UNION SELECT MIN(uname),1,1 FROM users WHERE uname > “arnold" --

To populate the rest of this table I just used variations of this statement replacing the uname with pwd, or realname. To get the passwords I did.

' UNION SELECT pwd1,1 FROM users WHERE uname = “admin” –

Getting the other passwords I just changed the user name. The realname I used almost the same syntax, but replaced the uname from the first statement with realname.

Uname varchar(10)	Pwd varchar(20)	Realname varchar(100)
admin	0ff13b80f2ed3b89e11ad0a1020d7859	Mike Rosulek
arnold	b9318e2c5b8ae9cb6910825f095dec2c	Arnold J Rimmer
dave	58910ed1dcf0e263b180c44f33c2b714	Dave Lister
kryten	4fb3eb037a1053a7efc802686329f03d	Kryten 2X4B 523P

8. Suggest a patch to the code, to fix the SQL injection vulnerability

Like stated in the notes we could use ? as a placeholder.

The new code would look like.

My $sql = “select * from users where uname=? and pwd=?”;

The question marks will be replaced by the user input when they are inserted into the table. This way the user input will not be plopped down directly in this SQL statement allowing the users code to inject SQL statements.