

Q1 SQL Injection

2 Points

This homework has instant feedback. When you click "Save Answer," if the answer is correct, you will see an explanation. You can resubmit as many times as you want.

You find the following Java code in the client login section of an online banking website:

```
/**
 * Check whether a username and password combination is valid.
 */
ResultSet checkPassword(Connection conn, String username, String password)
    throws SQLException {
    String query = "SELECT userID FROM Customers WHERE username = '"
        + username + "' AND password = SHA256('" + password + "')";
    Statement s = conn.createStatement();
    return s.executeQuery(query);
}
```

Assume that before issuing a request, the bank's server calls `checkPassword` and ensures that the returned `ResultSet` contains exactly one `userID`. If the set returns any other number of `userID` records, the bank fails the request. Otherwise the request is issued as the user represented by `userID`.

Assume `executeQuery` can execute multiple SQL queries separated by a semicolon `;`.

Q1.1

1 Point

Which usernames could an attacker enter in order to delete the Customers table? Select all that apply.

☐

bob; DROP TABLE Customers; --

☐

bob'; DROP TABLE Customers;

☒

bob'; DROP TABLE Customers; --

☐

bob'; DROP TABLE Customers;' --

Q1.2

1 Point

What username could an attacker enter in order to issue a request as user Admin, without having to know the password?

☒

Admin'; --

☒

Admin' AND 1=1; --

☒

bob' AND 1=0; SELECT userID FROM Customers WHERE username = 'Admin'; --

☐

Admin' AND 1=1

☐

Admin' AND 'bob'='bob

Q2 Web Security True/False

6 Points

Q2.1

1 Point

Using parameterized SQL is a good defense against SQL injection.

- ☒ True
- ☐ False

Q2.2

1 Point

Two Javascript scripts embedded in pages running in two different tabs on a user's browser can never access the resources of each other.

- ☐ True
- ☒ False

Q2.3

1 Point

If Eve takes control of `https://maps.google.com`, a site which Alice uses regularly, Eve can read all Alice's cookies from `https://mail.google.com`.

- ☐ True
- ☒ False

Q2.4

1 Point

Browsers have a private browsing mode, which lets you visit websites without using any cookies.

☐ True

☒ False

Q2.5

1 Point

Because of the cookie policy, you cannot be tracked across domains by cookies.

☐ True

☒ False

Q2.6

1 Point

The same-origin policy prevents XSS (cross-site scripting) attacks.

☐ True

☒ False

Q3 Snapitterbook

3 Points

This question walks you through some common web security exploits and defenses, using a vulnerable social networking website called Snapitterbook.

For simplicity, throughout this question we define a helper function `render(endpoint, content)`, which displays `content` at the specified `endpoint` of the website. For example, `render("/hello", "Hello World")` will cause users to see `Hello World` when they visit `https://snapitterbook.com/hello`.

Q3.1

1 Point

Snapitterbook displays a user's status on their wall. When a user updates their status with `new_status`, the following Python code snippet runs:

```
query = "INSERT INTO statuses VALUES (?, ?);"

# add the status to the database using parameterized SQL
db.execute(query, username, new_status)
```

When someone visits a user's wall, the following Python code snippet runs:

```
query = "SELECT status from statuses WHERE user = ?"

# fetch the status from the database using parameterized SQL
status = db.execute(query, username)

# display the user's status on their wall
render("/{}/wall".format(username), status)
```

What type of attack is this code vulnerable to?

- ☐ SQL injection
- ☒ Stored XSS
- ☐ Reflected XSS
- ☐ CSRF
- ☐ None of the above, the code is secure.

Q3.2

1 Point

If a user tries to access the profile of a nonexistent user, Snapitterbook runs the following code snippet to display an error:

```
# assume username is read from a URL parameter
username = escape_sql(username)

# check if username exists
if not is_valid_username(username):
    render("/profile/{}".format(username), "{} does not exist".format(username))
```

For example, if a user goes to

`https://snapitterbook.com/profile/jason`, they will see the message

`jason does not exist`.

What type of attack is this code vulnerable to?

- ☐ SQL injection
- ☐ Stored XSS
- ☒ Reflected XSS
- ☐ CSRF
- ☐ None of the above, the code is secure.

Q3.3

1 Point

Snapitterbook users can repost their friend's statuses by clicking a button on their friend's profile. When a user clicks the button, the following HTTP request is made:

```
https://snapitterbook.com/repost?post=<id of post>
```

The user's cookies are sent along with the request so that the server knows which user is reposting the status.

What type of attack is this code vulnerable to?

- ☐ SQL injection
- ☐ Stored XSS
- ☐ Reflected XSS
- ☒ CSRF
- ☐ None of the above, the code is secure.

Q4 Feedback

0 Points

What's something we could do to make the class better? Or, what did you find most difficult or confusing from lectures or the rest of class, and what would you like to see explained better? If you have feedback, submit your comments here.

If you have feedback, submit your comments on [this form](#). Your name will not be connected to any feedback you provide, and anything you submit here will not affect your grade.

Leaving feedback is completely optional! To encourage submissions, please grab the magic word from the feedback form (available in the first page) and enter it here.

Homework 5

● Graded

15 Minutes Late

 Select each question to review feedback and grading details.

Student

Yiyun Chen

Total Points

11 / 11 pts

Question 1

SQL Injection

2 / 2 pts

1.1 (no title)

1 / 1 pt

1.2 (no title)

1 / 1 pt

Question 2

Web Security True/False

6 / 6 pts

2.1 (no title)

1 / 1 pt

2.2 (no title)

1 / 1 pt

2.3 (no title)

1 / 1 pt

2.4 (no title)

1 / 1 pt

2.5 (no title)

1 / 1 pt

2.6 (no title)

1 / 1 pt

Question 3

Snapitterbook

3 / 3 pts

3.1 (no title)

1 / 1 pt

3.2 (no title)

1 / 1 pt

3.3 (no title)

1 / 1 pt

Question 4

[Feedback](#)

0 / 0 pts