## Input Validation

A screenshot of a computer

AI-generated content may be incorrect.

One nice thing about an allow listing is that it gives you more control over what is acceptable input and inherently excludes more invalid inputs.

## Parameterized Statements

A screenshot of a computer program

AI-generated content may be incorrect.

For Q1, the incorrect code snippet is combining data with the command, which exposes potential for attacks because it is easier to manipulate the way the code is interpreted when you mix those two (data and commands) together.

For Q2, we have basically the same scenario, where data and sql statements are being combined in the incorrect scenario while the data is processed in isolation I the correct scenario. This allows the incoming input data to be processed separate from the sql statement which is defined in the code ahead of time.

## Protecting Data

A screenshot of a computer program

AI-generated content may be incorrect.

For Q1, the incorrect scenario is directly storing the password and checking it against the users password.

Q2 shows that the hashed password is being stored instead because it applies the hashing and salting algorithms on the password, then checks if the hash result exists locally to validate the correctness.

A screenshot of a computer

AI-generated content may be incorrect.

Q3 is clearly applying some data cleaner utilities on sensitive info like the credit card number before putting it into the storage object.

## Preventing Cross-Site Scripting

A screenshot of a computer program

AI-generated content may be incorrect.

The correct code sample here for Q1 shows that the input is being passed into an escape utility specifically for HTML which helps to sanitize the HTML before it gets processed.

A screenshot of a computer

AI-generated content may be incorrect.

Q2 highlighted ensures that “&#39” is used which is the html code for a single quote. This helps to make sure the correct version of the character is being used when processing the input.

Q3 code on the right is safer because the input is not processed directly as HTML; it is instead targeting the text content of the html which does not need to be processed as HTML. If processed as HTML, someone could add a script tag and insert malicious javascript code that could potentially do some major damage to the server running the code.

A screenshot of a computer

AI-generated content may be incorrect.

In Q4, the highlighted is showing correct separation of keeping each element separate as opposed to an interpolated string that combines them, which has vulnerabilities that make is subject to Cross-Site Scripting which is also known as code injection.