**Coventry University**

# **VT6005CEM – Security**

# **Lab 5 – Data validation and prepare statement**

**Intended Learning Outcomes**

Upon completion of this tutorial/lab, you should be able to:

* Use **preg\_match** function and regular expression to validate data
* Use prepare statement to prevent SQL injection

Introduction

Data validation is the process of ensuring that a program operates on clean, correct and useful data. It can provide security of data that are inputted and processed to the system. In this lab, we will use **preg\_match** function and regular expression to validate data.

SQL injection is an attack method that input a part of SQL statements into an entry field for execution. By using this technique, attacker can circumvent the authentication process or run any customized malicious SQL statement. In this lab, we will use prepare statement to prevent SQL injection.

Exercise 1: PHP Data Validation

In PHP we can use **preg\_match** function to check whether a string follow a particular format. The signature of function is shown below:

The following PHP program check whether an account balance’s format is correct (at least 1 integer digit with a “.” symbol, then followed by 2 digits. For example: 15.50, 100.00, 0.0)

|  |
| --- |
| $balance\_str = "10.50";  if (preg\_match("/^[0-9][0-9]\*\.[0-9]{2}$/", $balance\_str))  $msg = "The balance string is correct";  else  $msg = "The balance string is incorrect";  echo $msg; |

The following statement:

|  |
| --- |
| if (preg\_match("/^[0-9][0-9]\*\.[0-9]{2}$/", $balance\_str)) |

Check whether the 2nd parameter **$balance\_str** match the 1st parameter **$regex**. If **$balance\_str** matches **$regex**, return 1. Otherwise return 0.

**Coding Exercise**

1. Download **lab4\_student.zip** and study the following PHP files:

* **register\_form.php**
* **register\_result.php**
* **login\_form.php**
* **login\_result.php**

1. Extract **user.sql** and Import this file to database **test**
2. Open **register\_result.php**, fill in the code according to the following instructions:
3. Write a regular expression and put suitable parameters to check the user ID is composed within 6 to 12 alphanumeric characters.
4. Write a regular expression and put suitable parameters to check the password is composed within at least 8 alphanumeric characters and allow "?" and "!" symbol.
5. Write regular expression and put suitable parameters to check the name is composed with English characters starts with capital letter.
6. Write a regular expression and put suitable parameters to check the email address.
7. Write a regular expression and put suitable parameters to check the color hex code.

Exercise 2: PHP Prepared Statement

A prepared statement is a feature used to execute the same (or similar) SQL statements repeatedly with high efficiency.

Prepared statements basically work like this:

**Prepare:**

An SQL statement template is created and sent to the database. Certain values are left unspecified, called parameters (labeled "?").

Example: **INSERT INTO MyGuests VALUES(?, ?, ?)**

The database parses, compiles, and performs query optimization on the SQL statement template, and stores the result without executing it.

**Execute:**

At a later time, the application binds the values to the parameters, and the database executes the statement. The application may execute the statement as many times as it wants with different values

Compared to executing SQL statements directly, prepared statements are very useful against SQL injections, because parameter values, which are transmitted later using a different protocol, need not be correctly escaped. If the original statement template is not derived from external input, SQL injection cannot occur.

Assume that we have constructed the following database:

|  |  |  |  |
| --- | --- | --- | --- |
| **post\_id** | **name** | **subject** | **comment** |
| 1 | Leo | I’m so boring | Anyone want to chat with me? |
| 2 | Tom | I can chat with you | Do you know programming? |
| 3 | Seth | I can chat with you too | Do you know information security? |
| 4 | Leo | I know both of them | Yes, I know programming and information security! |

The following PHP program retrieve data from database using prepare statement

|  |
| --- |
| $conn = new mysqli("localhost", "root", "", "test");  if ($conn->connect\_error)  {  die("Connection failed: ". $conn->connect\_error);  }  $name = "%".$\_POST["name"]."%";  $subject = "%".$\_POST["subject"]."%";  $search\_sql = $conn->prepare("SELECT \* FROM comment WHERE name LIKE ? and subject LIKE ?");  $search\_sql->bind\_param("ss", $name, $subject);  $search\_sql->execute();  $search\_sql->bind\_result($result\_pid, $result\_name, $result\_subject, $result\_comment);  while($search\_sql->fetch())  {  echo "Post ID: " . $result\_pid . "<br>";  echo "Name: " . $result\_name . "<br>";  echo "Subject: " . $result\_subject . "<br>";  echo "Comment: " . $result\_comment . "<br><hr>";  }  mysqli\_close($conn); |

1. The following statement:

|  |
| --- |
| $search\_sql = $conn->prepare("SELECT \* FROM comment WHERE name LIKE ? and subject LIKE ?"); |

Create a prepared statement. This statement consists of SQL statement template and the variable part (denoted as "?" symbol).

1. The following statement:

|  |
| --- |
| $search\_sql->bind\_param("ss", $name, $subject); |

Bind the parameter to the prepared statement.

The 1st parameter **"ss"** is binding string. Itmeans binds 2 parameters, whether the 1st and 2nd parameters are string.

The following table shows the meaning of character in a binding string

|  |  |
| --- | --- |
| **Characters** | **Description** |
| i | corresponding variable has type integer |
| d | corresponding variable has type double |
| s | corresponding variable has type string |
| b | corresponding variable is a blob and will be sent in packets |

1. The following statement:

|  |
| --- |
| $search\_sql->execute(); |

Execute the SQL statement with binding parameters

1. The following statement:

|  |
| --- |
| $search\_sql->bind\_result($result\_pid, $result\_name, $result\_subject, $result\_comment); |

Bind the result set to the variable:

Column name **post\_id** bind to **$result\_pid**

Column name **name** bind to **$result\_name**

Column name **subject** bind to **$result\_subject**

Column name **comment** bind to **$result\_comment**

1. The following statement:

|  |
| --- |
| while($search\_sql->fetch())  {  echo "Post ID: " . $result\_pid . "<br>";  echo "Name: " . $result\_name . "<br>";  echo "Subject: " . $result\_subject . "<br>";  echo "Comment: " . $result\_comment . "<br><hr>";  } |

Fetch the rows one by one and show them on the web page.

**Coding Exercise**

1. Open **register\_result.php**, fill in the code according to the following instructions:
2. Write a prepare statement to search whether the user name exists
3. Write a parameter binding statement to bind the result set to variable **$id** and **$pwd** to the prepared statement you have written in (2a)
4. Write prepare statement to insert all form data to the **user** table
5. Write a parameter binding statement to bind variables **$id, $pwd, $name,** $email and $color to the prepared statement you have written in (2c)
6. Open **login\_result.php**, fill in the code according to the following instructions:
7. Write a prepare statement to search the corresponding user row (i.e. the username exists and password is correct)
8. Write a parameter binding statement to bind the result set to variable **$id** and **$pwd** to the prepared statement you have written in (2e)