Consultas SQL “Preparadas”

El uso de consultas preparadas es necesario por razones de seguridad y para prevenir la inyección de SQL.

Cuando se construye una consulta SQL concatenando directamente los valores de entrada proporcionados por los usuarios, se crea una vulnerabilidad conocida como inyección de SQL. Los atacantes pueden manipular los datos de entrada de manera malintencionada e insertar código SQL adicional en la consulta, lo que puede conducir a la exposición de datos sensibles, la modificación no autorizada de la base de datos o incluso la eliminación de datos.

Las consultas preparadas resuelven este problema al separar la consulta de los datos de entrada. En lugar de concatenar los valores directamente en la consulta, las consultas preparadas utilizan marcadores de posición (por ejemplo, "?") para representar los valores. Luego, los valores se vinculan a la consulta utilizando el método **bind_param()** o una función similar. Esto permite que la base de datos distinga claramente entre la estructura de la consulta y los datos proporcionados, evitando así la inyección de SQL.

Consultas SQL “Preparadas”

...

```
$sql = "SELECT * FROM usuarios WHERE usuario = ? AND contraseña = ?";
$stmt = $conn->prepare($sql);
$stmt->bind_param("ss", $usuario, $contraseña);
$stmt->execute();
$resultado = $stmt->get_result();
```

...

Consultas SQL “Preparadas”

Explicación paso a paso del código:

$stmt = $conn->prepare($sql); - En esta línea, se crea un objeto de sentencia preparada ($stmt) utilizando el método prepare() del objeto de conexión a la base de datos ($conn). La sentencia SQL que se prepara es la que se encuentra en la variable $sql.

$stmt->bind_param("ss", $usuario, $contraseña); - En esta línea, se utiliza el método **bind_param()** del objeto de sentencia preparada **($stmt)** para vincular los parámetros de la consulta con los valores proporcionados.

El primer argumento de **bind_param()** especifica los tipos de datos de los parámetros. En este caso, se utilizan dos caracteres "**s**" para indicar que los parámetros son cadenas de texto.

Los siguientes argumentos son las variables que contienen los valores de los parámetros. En este caso, **$usuario** y **$contraseña** son las variables que contienen los valores que se desean insertar en la consulta.

La función **bind_param()** se utiliza para evitar la inyección de SQL al vincular los valores proporcionados como parámetros en lugar de concatenarlos directamente en la consulta. Esto proporciona una capa adicional de seguridad al interactuar con la base de datos.