Contramedidas SQL Injection:

Existen dos contramedidas principales a la hora de combatir el SQL Injection:

* Desinfectar la entrada del usuario

Al estar utilizando una versión anterior de PHP (5.4.45) es aconsejable ejecutar toda la entrada del usuario a través de una función llamada mysql\_real\_escape\_string(). Básicamente, lo que hace es eliminar todos los caracteres especiales de una cadena para que pierdan su significado cuando los utilice la base de datos.

Por ejemplo, si tiene una cadena como “ I'm a string ”, un atacante puede utilizar el carácter de comilla simple (') para manipular la consulta de la base de datos que se está creando y provocar una inyección SQL. Ejecutarlo a través mysql\_real\_escape\_string() produce “ I\'m a string “, que agrega una barra invertida a la comilla simple, evitándola. Como resultado, la cadena completa ahora se pasa como una cadena inofensiva a la base de datos, en lugar de poder participar en la manipulación de consultas.

Hay un inconveniente con esta contramedida: es una técnica realmente antigua que acompaña a las formas más antiguas de acceso a bases de datos en PHP. A partir de PHP 7, esta función ya no existe, lo que nos lleva a nuestra próxima solución…

* Utilizar declaraciones preparadas y parámetros enlazados

Una declaración preparada es una característica que se utiliza para ejecutar las mismas (o similares) declaraciones SQL repetidamente con alta eficiencia.

Las declaraciones preparadas básicamente funcionan así:

1. Preparar: se crea una plantilla de declaración SQL y se envía a la base de datos. Ciertos valores se dejan sin especificar, llamados parámetros (etiquetados como "?"). Ejemplo: INSERT INTO MyGuests VALUES(?, ?, ?)
2. La base de datos analiza, compila y realiza la optimización de consultas en la plantilla de declaración SQL y almacena el resultado sin ejecutarlo.
3. Ejecutar: en un momento posterior, la aplicación vincula los valores a los parámetros y la base de datos ejecuta la instrucción. La aplicación puede ejecutar la sentencia tantas veces como quiera con diferentes valores.

En comparación con la ejecución directa de sentencias SQL, las sentencias preparadas tienen tres ventajas principales:

-Las declaraciones preparadas reducen el tiempo de análisis ya que la preparación de la consulta se realiza solo una vez (aunque la declaración se ejecuta varias veces)

-Los parámetros vinculados minimizan el ancho de banda al servidor, ya que necesita enviar solo los parámetros cada vez, y no toda la consulta.

-Por ultimo y mas relevante en relacion a nuestro TP, las declaraciones preparadas son muy útiles contra las inyecciones de SQL, porque los valores de los parámetros, que se transmiten más tarde usando un protocolo diferente, no necesitan secuencias de escape (‘\’). Si la plantilla de declaración original no se deriva de una entrada externa, no se puede producir la inyección SQL.  
Resumiendo, las declaraciones preparadas son una forma de realizar consultas a bases de datos de manera más segura y confiable. La idea es que en lugar de enviar la consulta sin procesar a la base de datos, primero le decimos a la base de datos la estructura de la consulta que enviaremos. Esto es lo que queremos decir con "preparar" una declaración. Una vez que se prepara una declaración, pasamos la información como entradas parametrizadas para que la base de datos pueda "llenar los vacíos" conectando las entradas a la estructura de consulta que enviamos antes. Esto elimina cualquier SQL Injection que puedan tener las entradas, lo que hace que se traten solamente como variables en todo el proceso.