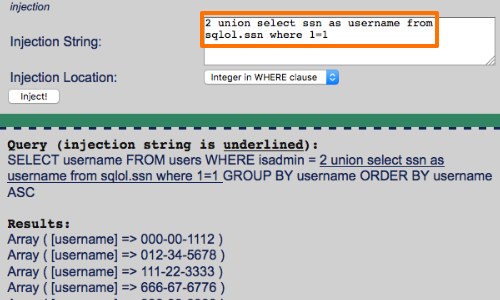
# [SQL Injection](https://portswigger.net/web-security/sql-injection): Bypassing Common Filters

In some situations, an application that is vulnerable to [SQL injection (SQLi)](https://portswigger.net/web-security/sql-injection) may implement various input filters that prevent you from exploiting the flaw without restrictions. For example, the application may remove or sanitize certain characters or may block common SQL keywords. In this situation, there are numerous tricks you can try to bypass filters of this kind.

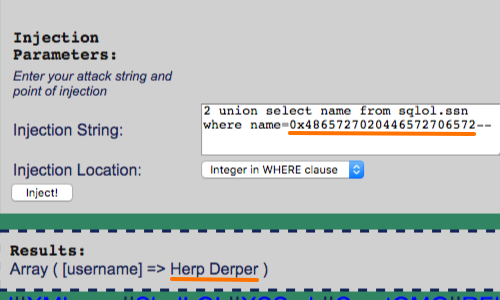
The example uses a version of the "Magical Code Injection Rainbow" taken from OWASP's Broken Web Application Project. [Find out how to download, install and use this project](https://www.owasp.org/index.php/OWASP_Broken_Web_Applications_Project).

### Avoiding Blocked Characters



If the application removes or encodes some characters that are often used in [SQLi](https://portswigger.net/web-security/sql-injection) attacks, you may still be able to perform an attack.

For example, the single quotation mark is not required if you are injecting into a numeric data field or column name.

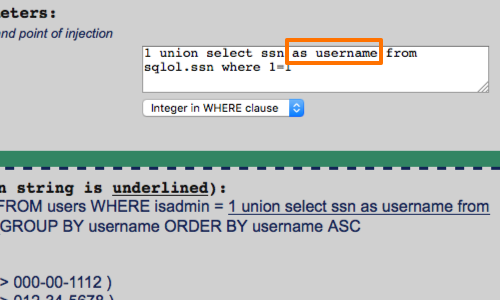


If you do need to introduce a string in to your attack payload, you can do this without needing to use quotes. In MySQL, the following statement:

SELECT username FROM users WHERE isadmin = 2 union select name from sqlol.ssn where name='herp derper'--

is equivalent to:

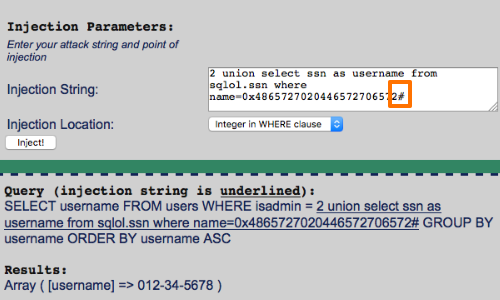
SELECT username FROM users WHERE isadmin = 2 union select name from sqlol.ssn where name=0x4865727020446572706572--



If the comment symbol is blocked, you can often craft your injected data such that it does not break the syntax of the surrounding query.

In the example opposite we have altered the structure of the query with the AS keyword.

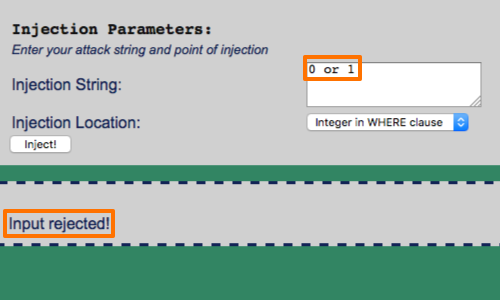
The MySQL AS keyword is used to specify an alternate name to use when referring to either a table or a column in a table.



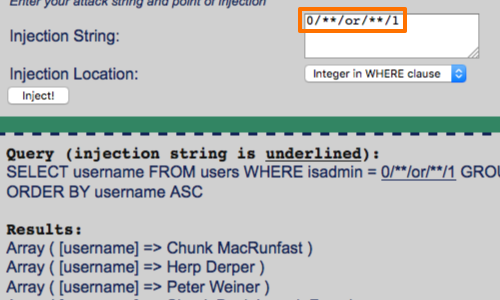
Additionally, in some cases you can use different characters to comment out the rest of the query.

Here we have used the # character.

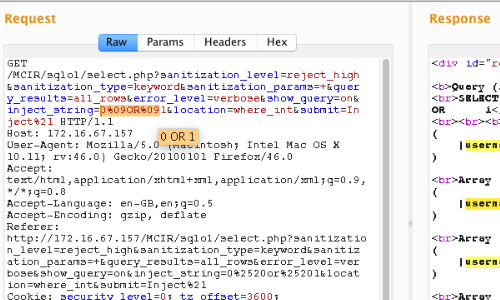
### Avoiding Whitespace



If the application blocks or strips from your input, you can use comments to simulate whitespace within your injected data.



You can insert inline comments into SQL statements in the same way as for C++, by embedding them between the symbols /\* and \*/.



Here we can see that our input:

0/\*\*/or/\*\*/1

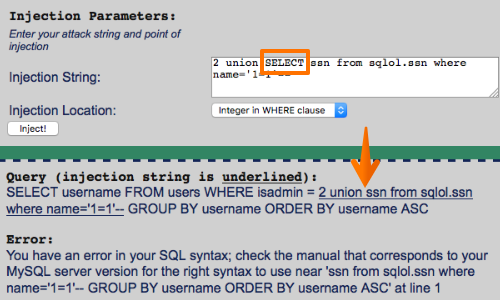
Is equal to:

0 or 1

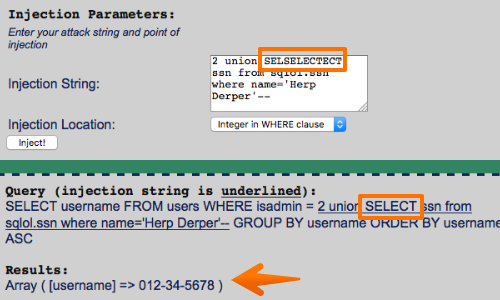
Additionally, in MySQL, comments can even be inserted within keywords themselves, which provides another means of bypassing some input validation filters while preserving the syntax of the actual query:

SEL/\*\*/ECT

### Stripped Input



Some input validation routines employ a simple blacklist and either block or remove any supplied data that appears on this list. In this instance, you should try looking for common defects in validation and canonicalization mechanisms.



For example, if the SELECT keyword is being blocked or removed, you can try the following bypasses:

SeLeCt

%00SELECT

SELSELECTECT

%53%45%4c%45%43%54

%2553%2545%254c%2545%2543%2554