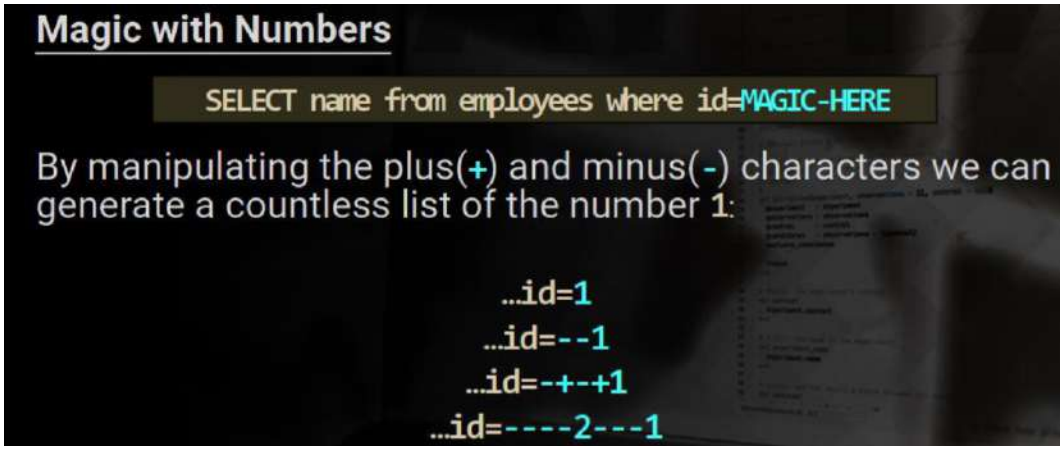
# Module 8 - SQLi Filter Evasion and WAF Bypassing

## DBMS Gadgets

- ▼ **Comments //used for obfuscation ^_^**
  - ▼ **mysql**
    - # Hash
    - /* C-style (MySQL 5.5.30 or higher)
    - — - SQL
    - ;%00 NULL byte
  - ▼ **Oracle**
    - /* C-style
    - — - SQL
- ▼ **Functions and Operators**
  - ▼ **mysql**



## Magic with Numbers

```
SELECT name from employees where id=MAGIC-HERE
```

We'll also introduce Bitwise Functions here; that is, functions that performs bit arithmetic operations. For example, we can generate the number 1 as follows:

```
...id=1&1
...id=0|1
...id=13^12
...id=8>>3
...id=~-2
```

## Magic with Numbers

```
SELECT name from employees where id=MAGIC-HERE
```

We can also use Logical Operators like these:

```
...id=NOT 0
...id=!0
...id=!1+1
```

```
...id=1&&1
...id=1 AND 1
...id=!0 AND !1+1
```

```
...id=1 || NULL
...id=1 || !NULL
...id=1 XOR 1
```

## Magic with Numbers

```
SELECT name from employees where id=MAGIC-HERE
```

A number can be also generated using functions that have nothing to do with numbers. For example, we can use Regular Expression Operators to match a string and then get 0 or 1, like the following:

```
...id={anything} REGEXP '.*'
...id={anything} NOT REGEXP '{randomkeys}'
...id={anything} RLIKE '.*'
...id={anything} NOT RLIKE '{randomkeys}'
```

## Magic with Numbers

```
SELECT name from employees where id=MAGIC-HERE
```

Additionally, some Comparison Operators are useful for generating numbers as well:

```
...id=GREATEST(0,1)
...id=COALESCE(NULL,1)
...id=ISNULL(1/0)
...id=LEAST(2,1)
```

## Magic with Numbers

```
SELECT name from employees where id=MAGIC-HERE
```

Unfortunately, in SQL Server we cannot use two equal signs concatenated:

```
...id=1
...id=--1
...id=-+-+1
id=-+-+-+-+-+-+1
id=-+-+-+-+-+-+-+-+-+-+1*-+-+-+-+-+-+-+-+-+-+1
```

## Magic with Numbers

```
SELECT name from employees where id=MAGIC-HERE
```

The set of Bitwise Operators are much simpler in MySQL, so we can only manipulate using & (AND), | (OR) and ^ (XOR).

Naturally, if we want to do binary shifting, then we need to combine them.

**Magic with Numbers**

```
SELECT name from employees where id=MAGIC-HERE
```

While MySQL proposes only four logical operators, there are other operators that can also be leveraged for testing the whether or not some conditions are true. In SQL Server, these are all grouped in one table Logical Operators. However, there are no short forms, so &&, ||, etc. are not valid in this DBMS.

▼ **Oracle**

**Magic with Numbers**

```
SELECT name from employees where id=MAGIC-HERE
```

Oracle is much more restrictive! If we want to use arithmetic operators, then we must create a valid expression to avoid the ORA-00936: missing expression error:

```
…id=1
…id=--1
…id=-+-+1
id=-(-1)
id=-(1)*-(1)
```

**Magic with Numbers**

Due to the fact that almost everything must be an expression, in order to combine values, functions and operators into expressions, we can use the following list of Conditions mixed to Expressions.

For example:

```
SELECT name from employees where id=some(1)
```

▼ **Intermediary Character**

▼ **mysql & mssql**

## Universal Whitespace Chars

```
SELECT[CHAR]name[CHAR]from[CHAR]employees
```

In MySQL, the "UNIVERSAL" characters allowed as whitespaces are:

| Codepoint | Character |
|---|---|
| 9 | U+0009 CHARACTER TABULATION |
| 10 | U+000A LINE FEED (LF) |
| 11 | U+000B LINE TABULATION |
| 12 | U+000C FORM FEED |
| 13 | U+000D CARRIAGE RETURN (CR) |
| 32 | U+0020 SPACE |

▼ **Oracle**

## Universal Whitespace Chars

```
SELECT[CHAR]name[CHAR]from[CHAR]employees
```

In Oracle, the list shrinks back to "*normal*". There are 7 characters in total, making it only one more than MySQL. In MySQL, the NULL char is a way to comment out queries, but in Oracle it is a valid space.

| Codepoint | Character |
|---|---|
| 0 | U+0000 NULL |
| 9 | U+0009 CHARACTER TABULATION |
| 10 | U+000A LINE FEED (LF) |
| 11 | U+000B LINE TABULATION |
| 12 | U+000C FORM FEED |
| 13 | U+000D CARRIAGE RETURN (CR) |
| 32 | U+0020 SPACE |

▼ **MySQL/MSsql/Oracle**

## Plus Sign

In all the DBMSs we can use the "PLUS SIGN" to separate almost all the keywords except FROM.

For example:

```
SELECT+name FROM employees WHERE+id=1 AND+name LIKE+'J%'
```

## Other Characters

In addition to the previous characters, in all the DBMSs (pending the right context) we can also use Parenthesis (), Operators, Quotes and of course the C-style comments /**/.

# ▼ Constants and Variables

mysql reserved words

- we can obfuscate these keywords by manipulating uppercase and lowercase characters.
  - SelECT, SELect
- SHOW VARIABLES;
- use @@ to retreive a specific value.
  - eg: @@fi_boolean_syntax
- define a custom variable:
  - SET @myvar={expression}
  - SET @myvar:={expression}

MSSQL

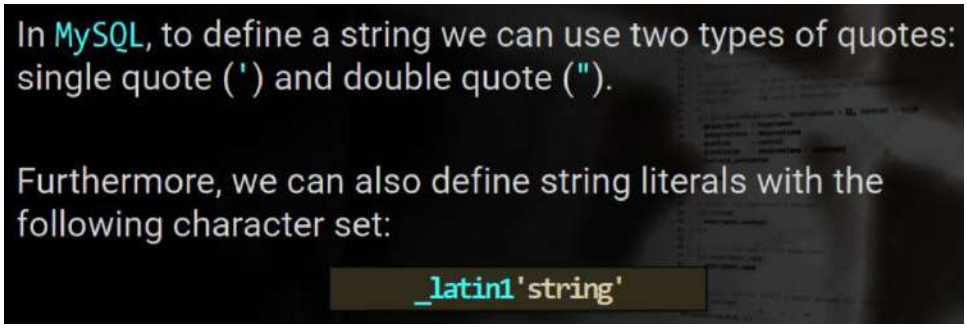
---

Oracle reserved words

- we can use CREATE TABLE `DATABASE` (id number);
  - because `DATABASE` keyword is not reserved.

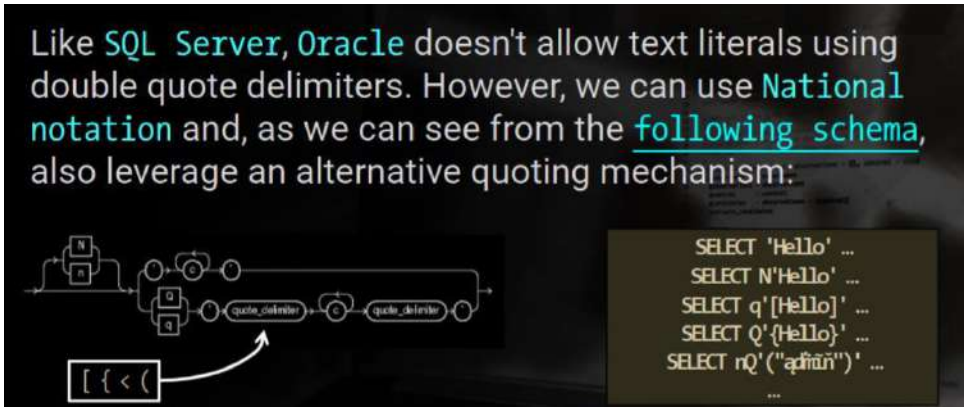
# ▼ Strings

## ▼ Regular Notation

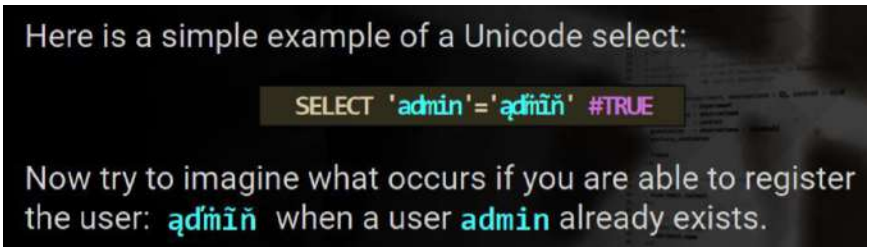


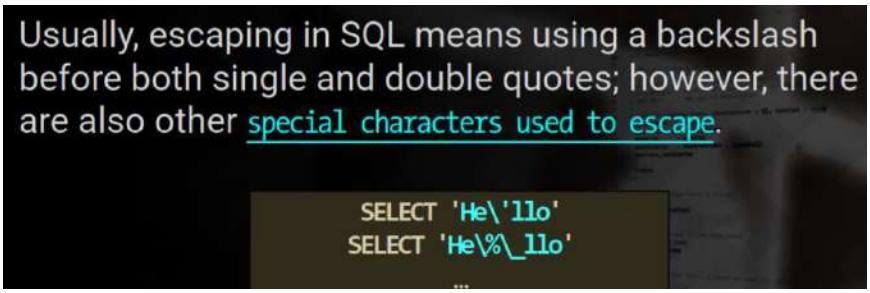

- ''

- “”
- also we can represent strings with placing character set before them
  - eg: _latin1’string’;
  - eg: _ascii”HOLA”;
- National character set:
  - SELECT N’mystring’; #or n
- X’hexValue’
  - SELECT x’4F48045’
  - SELECT 0x4F485045
- b’literal’
  - SELECT ‘a’= B’110001’ #True



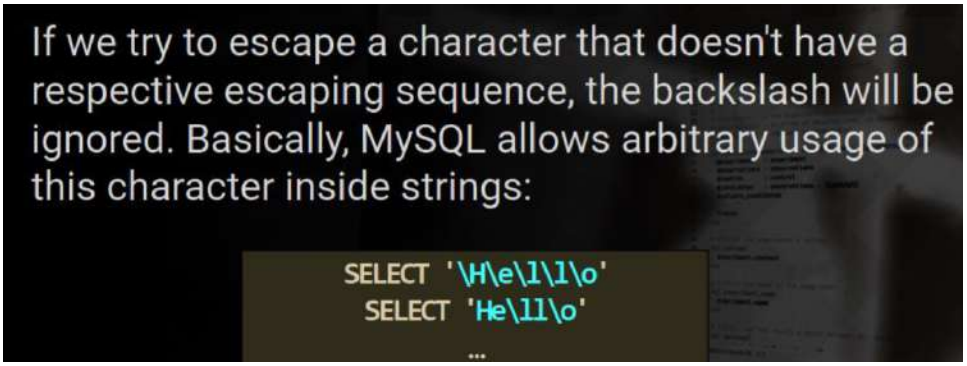

## ▼ Unicode






## ▼ Escaping

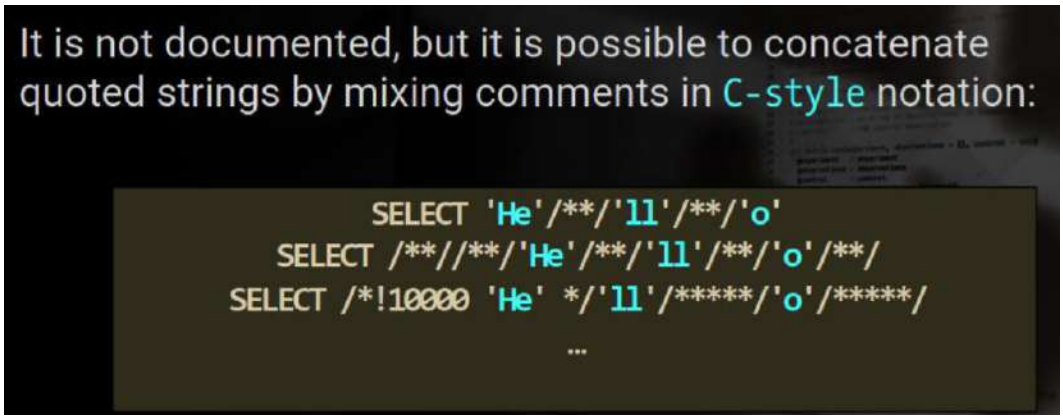
- **Special escape characters:**



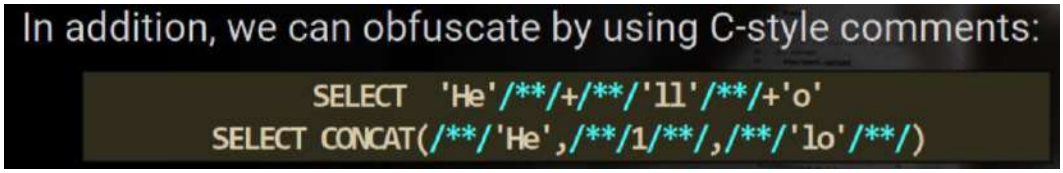

```
SELECT * FROM Users WHERE Name = 'O''Brien' // Single Quote (
SELECT "Last Name", "First Name" FROM "Employee Information"
SELECT * FROM Products WHERE Description LIKE '%\%%' //Bashsl
//Percent Sign (%) and Underscore (_) in LIKE Operator
SELECT * FROM Employees WHERE Name LIKE '%\_%' ESCAPE '\'
```

## ▼ Concatenation

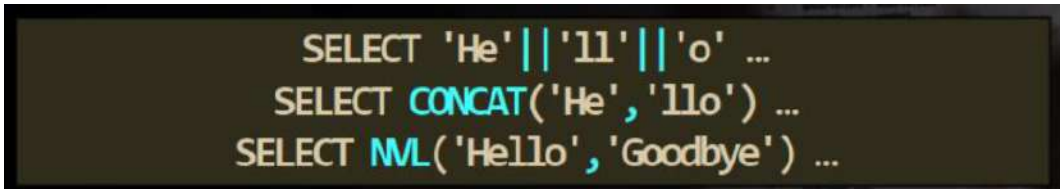
- in **MySQL** function **CONCAT** and **CONCAT_WS** (ws stands for with separator)

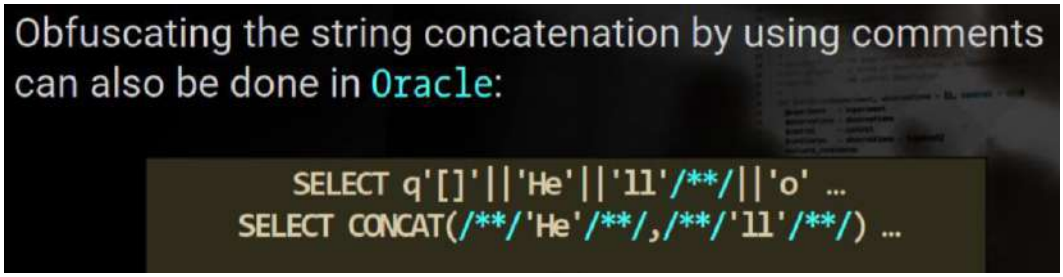



- in **SQL SERVER** concatenation can be done by using both the **+** operator and the function **CONCAT**




- in **Oracle**, the concatenation operator is **||** and also we can use **CONCAT** and **NVL**




## ▼ Integers

Numbers rule the world and also the filters. Typically, we use digits to represent numbers; however, there are other interesting and useful methods used during the obfuscation process.

A generic example that can be useful in understanding how to construct a number is using the `PI` function. This function returns the value of `π (pi 3.141593...)`. We can use this result mixed with either `FLOOR` and obtain the value `3`, or with `CEIL` and obtain the value `4`.

We can continue using system functions like `version()` and obtain `5,6` or also continue to perform arithmetic operations.

For example, we can do `ceil(pi()*3)` to obtain the number `10`.