Getting started with MySQL Proxy

Giuseppe Maxia
QA Developer - MySQL AB

Sofia - OpenFest 2007
Agenda

- Overview
- Some fun
- Injecting queries
- Filtering and rewriting queries
- Working with results
- Proxy for logging and debugging
- Replication goodies
- Q&A
Proxy (< lat. procuratio)

= Someone taking care of someone else's interests

A server proxy is something acting on behalf of another server
Overview

MySQL Proxy

MySQL Server

1

2

3

4

- query injection
- filtering
- rewriting
- macro expansion

Client
Overview

PROXY

CORE
Overview

PROXY

CORE

- connection hook
- read query hook
- read result hook
Overview

PROXY CORE

connection hook

read query hook

read result hook
Overview

**PROXY CORE**
- connection hook
- read query hook
- read result hook

**Lua script**
- function
- function
- function
Overview
Overview

Why not ...

{ Perl ? PHP? Javascript? [whatever]? }
Overview
Overview

SMALL ( < 200 KB)
Overview

SMALL ( < 200 KB)
DESIGNED for EMBEDDED systems
Overview

SMALL ( < 200 KB)

DESIGNED for EMBEDDED systems

Widely used (lighttpd)
Overview

**SMALL** (< 200 KB)

**DESIGNED** for **EMBEDDED** systems

Widely used (lighttpd)

lighttpd, like MySQL Proxy, was created by Jan Kneschke
Overview

Very popular among game writers
Overview

Very popular among game writers
Overview

Very popular among game writers
Some fun

LIVE
function read_query(packet)
    if packet:byte() == proxy.COM_QUERY
        then
            local query = packet:sub(2)
            print("Hello world! Seen query: ", query)
        end
    end
Injecting queries (1)
Injecting queries (2)
function read_query(packet)
    -- ...
    proxy.queries:append(2, query1)
    proxy.queries:append(1, packet)
    proxy.queries:append(3, query2)

    return proxy.PROXY_SEND_QUERY
end
injecting

function read_query_result(inj)

    if res.id == 1 then
        return proxy.PROXY_SEND_RESULT
    else
        -- do something
        return proxy.PROXY_IGNORE_RESULT
    end
filtering queries

• Like injecting
• but without the original
working with results
working with results

• return the original result
working with results

• return the original result
• return a fake result
working with results

• return the original result
• return a fake result
• return an error
working with results

• return the original result
• return a fake result
• return an error
• alter the original result
working with results

• return the original result
• return a fake result
• return an error
• alter the original result
• return something different (affected/retrieved)
debugging
debugging

• Put a Proxy in between
debugging

• Put a Proxy in between
• use a sensible script to see what's going on (e.g. tutorial-packets.lua or tutorial-states.lua)
debugging

server

proxy

diagnostics
text

client
debugging scripts

server

proxy

client

diagnostics text

diagnostics text
Chained proxies: doubled features

server

proxy

pivot tables

proxy

loops

client
Testing

server

proxy

fake packets

client

e.g. connectors
logging via Proxy

# client (1)
mysql> drop table t1;
Query OK, 0 rows affected (0.05 sec)

mysql> create table t1 (i int);
Query OK, 0 rows affected (0.02 sec)
logging via Proxy

# proxy (1)
2007-08-24 11:37:28 296 --
drop table t1 >{0}

2007-08-24 11:37:35 296 --
create table t1 (i int) >{0}
logging via Proxy

# client (2)
mysql> insert into t1;
ERROR 1064 (42000): You have an error in your SQL syntax;
logging via Proxy

# proxy (2)

2007-08-24 11:37:43 296 --
insert into t1 >{0} [ERR]
logging via Proxy

# client (3)

mysql> insert into t1 values (1), (2);
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from t1;
+------+
| i    |
+------+
|    1 |
|    2 |
+------+
2 rows in set (0.00 sec)
logging via Proxy

# proxy (3)

2007-08-24 11:38:00 296 --
insert into t1 values (1),(2) >\{2\}

2007-08-24 11:38:03 296 --
select * from t1 <\{2\}
Rerouting traffic

Client → MySQL Server
3306/tcp

Client
3306/tcp
redirect by iptables
from 3306 to 4040

MySQL Server

4040/tcp

MySQL Proxy

(logging?)
Rerouting traffic

(1) do

```bash
sudo iptables -t nat \
-1 PREROUTING \
-s ! 127.0.0.1 -p tcp \
--dport 3306 -j \nREDIRECT --to-ports 4040
```
Rerouting traffic

(1) undo

```
sudo iptables -t nat \
   -D PREROUTING \
   -s ! 127.0.0.1 -p tcp \
   --dport 3306 -j \n   REDIRECT --to-ports 4040
```
Rerouting traffic

Client

MySQL Server

3306/tcp

Client

MySQL Server

3306/tcp

redirect by iptables from 3306 to 4040

4040/tcp

MySQL Proxy

(logging?)
replication goodies

Normally, clients must be replication-aware.
With a proxy, clients can be replication unaware.
Live examples

- loops
- logs
- shell access
- pivot tables
- more ...
Q&A

Any questions?

slides at http://datacharmer.org