Database access controls with DBD::Proxy and DBI::ProxyServer

Jacinta Richardson
<jarich@perltraining.com.au>

Perl Training Australia

Remote connections

- Not all databases handle connections over networks
- Some will if you install special client software (if it works on your architecture)
- Some will if you install certain libraries (if you system administrator will let you)
- Some just can't.

Why remote connections?

- Separate machines for separate jobs
- Load balancing
- Security
- Different architectures (Unix webserver, MS Access database)

DBD::Proxy to the rescue

- DBD::Proxy and DBI::ProxyServer let you make remote connections.
- Easily
- Portably
- Usually only needs a single line change to your code
- Or there abouts...

Bonus!

- Using DBD::Proxy and DBI::ProxyServer also gives you:
  - Host and user dependant access control
  - Host and user based encryption
  - Query restrictions
  - Compression

A brief into to connecting with DBI

```
use DBI;

my $dbh = DBI->connect("dbi:mysql:database=osdc",
                       $username, $password,
                       { AutoCommit => 1 })
        or die $DBI::errstr;
```

- The first argument is the data source name
  - The string `dbi`
  - The name of the driver
  - The actual data source (the database name)

Native remote connections

- Many databases do support remote connections.
- These usually just require the host and port to be added to your data source name (dsn)

```
use DBI;

my $dsn = "dbi:mysql:database=osdc;".
          "host=$hostname;port=$port";

my $dbh = DBI->connect($dsn, $username, $password,

                       { AutoCommit => 1 })
          or die $DBI::errstr;
```

Proxied remote connections

- Remote connections using DBD::Proxy requires a minor change to your dsn.
- You need to provide the dsn for the remote machine as well

```
my $host_dsn = "dbi:mysql:database=osdc";
my $dsn = "dbi:Proxy:hostname=$hostname;port=$port;".
          "dsn=$host_dsn";

my $dbh = DBI->connect($dsn, $username, $password,
                       { AutoCommit => 1 })
          or die $DBI::errstr;
```

Access Controls

- Access controls are set in the clients list in the DBI::ProxyServer configuration.
- These include:
  - mask
  - users
  - sql
  - accept
- Any of the mask, users and sql keys may be omitted.
- This behaves sensibly.

An example

```
clients => [
           {
              users => ["jarich", "pjf"],
              allow => 1,
           },
           {
              mask => "osdc.com.au$",
              sql => {
                        room => 'select room '.
                       'from locations where talk = ?'
                     },
              allow => 1,
           },
           {
              allow => 0
           }
        ],
```

Using the SQL

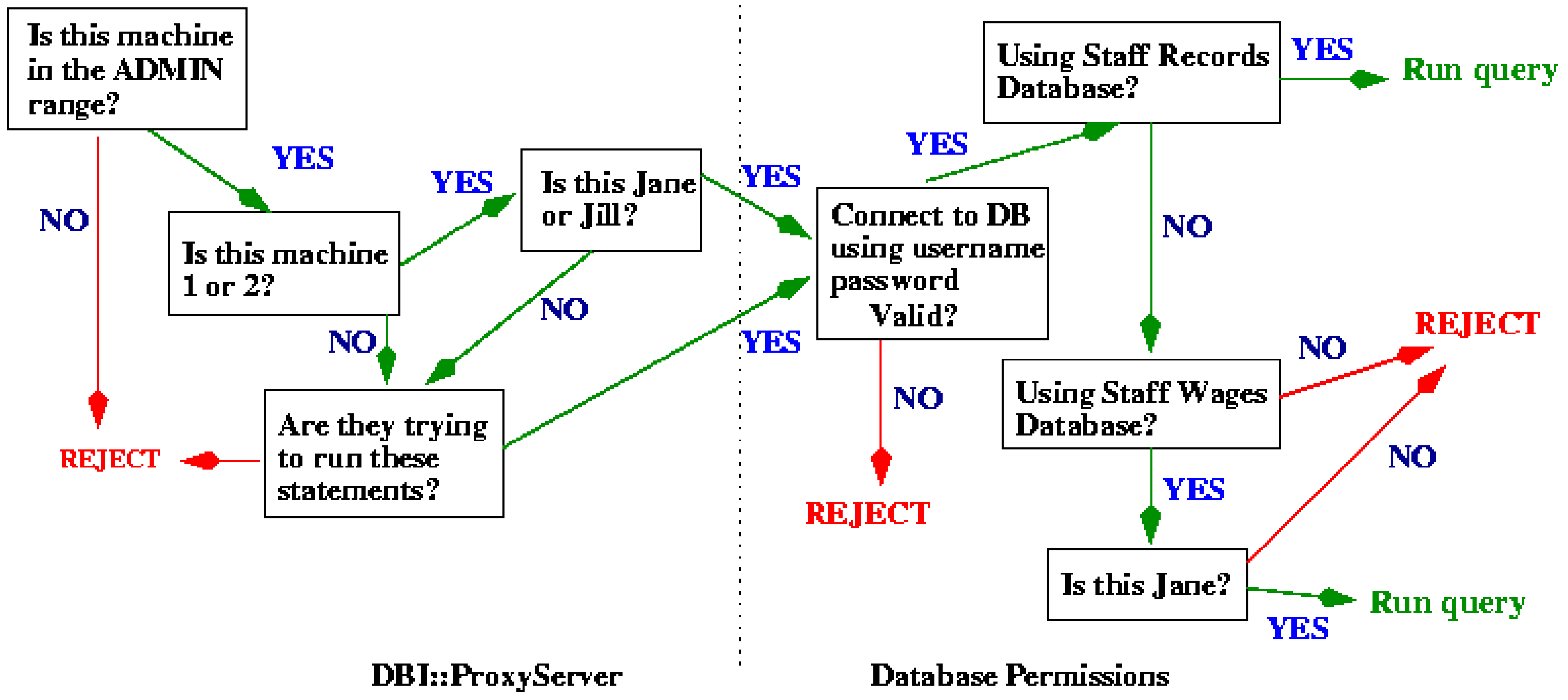
- To run an SQL command you call it with its short name, rather than the actual SQL.

```
$sth = $dbh->prepare("room");
$sth->execute($talk_name);
```

Uses

- The ability to set SQL query restrictions
  - Reduces susceptibility to SQL injection attacks
  - Forms a known set of allowed SQL statements (thus aiding the creation of indexes)
  - Discourages new developers from making mistakes such as forgetting to use placeholders.
- More importantly, used with the database's access rights, you can fine tune access restrictions.

For example


Is this machine in the ADMIN range?
YES
NO
REJECT
Is this machine 1 or 2?
YES
NO
Is this Jane or Jill?
NO
Are they trying to run these statements?
REJECT
YES
YES
Connect to DB using username password Valid?
NO
REJECT
YES
Using Staff Records Database?
YES
Run query
NO
Using Staff Wages Database?
NO
REJECT
YES
Is this Jane?
NO
YES
Run query
DBI::ProxyServer
Database Permissions

Encrypted connections

- Who can snoop the traffic on your network?
- Does your database contain sensitive data?
  - Client names?
  - Phone numbers?
  - Addresses?
  - Credit card information (heaven forbid!)?
- Does your database handle encryption for remote connections?
- Easily?

Two phase encryption

- DBD::Proxy and DBI::ProxyServer support two phase encryption.
- Shared host keys and ciphers are used during login and authorisation
- Shared user keys and ciphers are used thereafter (if provided)

Setting up encryption - client side

- Once again, this just involves changing the dsn.

```
my $host_dsn = "dbi:mysql:database=osdc";

my $dsn = "dbi:Proxy:hostname=$hostname;port=$port;".
          "cipher=$cipher;key=$key;".
          "usercipher=$usercipher;userkey=$userkey;".
          "dsn=$host_dsn";

my $dbh = DBI->connect($dsn, $username, $password,
                       { AutoCommit => 1 })
          or die $DBI::errstr;
```

Server side encryption

- And this just involves adding a bit to the configuration hash.

```
clients => [
            {
              users => [
                        {
                           name => "jarich",
                           cipher => Crypt::IDEA->new
                              (pack
('H*',$jarichkey)),
                        },
                       ],
              cipher => Crypt::IDEA->new
                             (pack('H*', $jhostkey)),
              allow => 1,
            },
]
```

Compression

- Client side

```
my $host_dsn = "dbi:mysql:database=osdc";

my $dsn = "dbi:Proxy:hostname=$hostname;port=$port;".
          "cipher=$cipher;key=$key;".
          "usercipher=$usercipher;userkey=$userkey;".
          "compression=gzip;".
          "dsn=$host_dsn";

my $dbh = DBI->connect($dsn, $username, $password,
                       { AutoCommit => 1 })
          or die $DBI::errstr;
```

- Server side (start proxy with a compression flag)

```
dbiproxy --compression gzip
```

In conclusion...

- DBD::Proxy and DBI::ProxyServer provide
  - Remote access to databases which don't support this natively
  - Access controls
  - Query restrictions
  - Encryption
  - Compression
- All easily achieved by changing your dsn in your call to connect.

Questions?

Perl Training Australia

This talk was proudly brought to you by

Jacinta Richardson
<jarich@perltraining.com.au>

Perl Training Australia
http://www.perltraining.com.au