Basic introduction to the Firebird Security Database, port redirection and multi-instances?

Fikret Hasovic, February 2022

Introduction

To improve the security of your Firebird database(s), you may need to deviate from the default configuration; you might decide to have a separate security database, or to use the database itself as a security database. You might also decide to redirect ports as well.

Security Databases

Since version 3, Firebird supports an unlimited number of security databases. Any database may act as a security database and can be a security database for itself.

Use databases.conf to configure a non-default security database. This example configures c:\db\private.security.fdb as the security database for the first and second databases:

```
first = c:\db\first.fdb
{
   SecurityDatabase = c:\db\private.security.fdb
}

second = c:\db\second.fdb
{
   SecurityDatabase = c:\db\private.security.fdb
}
```

Here we use a third database as its own security database:

```
third = c:\db\third.fdb
{
   SecurityDatabase = third
}
```

Creating an Alternative Security Database

To start using a separate, non-default security database, the first step is to create it, unless it already exists. An embedded isql connection is used:

```
> isql -user sysdba
```

```
SQL> create database ' c:\db\private.security.fdb';
```

Now connect to any database which will be served by the security database you are currently preparing, in order to create its SYSDBA user:

```
SQL> connect first;
SQL> create user sysdba password 'sysdba-in-private-security-password';
SQL> commit;
SQL> exit;
```

Follow the same instructions to initialize the database as its own security database.

Redirecting TCP/IP ports

Redirecting TCP/IP ports in Windows

Windows 10 has built-in support for port forwarding but it's not exposed in the Settings interface.

To port forward 127.0.0.1:3070 to 127.0.0.1:3051 in Windows 10/11:

1. Launch an Administrator Command Prompt.

2. Run “netsh interface portproxy add v4tov4 listenaddress=127.0.0.1 listenport=3070 connectaddress=127.0.0.1 connectport=3051”.

Notice how the values given for “listenaddress”, “listenport”, “connectaddress” and “connectport” correspond to those we stated earlier. You'll need to make adjustments here to suit the rule you're adding. For example, if you want to route to port 3052 on the device with IP 127.0.0.1, you'd use these values for the “connectport” and “connectaddress” respectively.

When you press enter, the rule will be added and applied immediately. Try connecting the listening address and port in your IBExpert – you should see the content being served by the connect address and port.

When the time comes to remove the rule, return to an administrator command prompt and use the following command:

netsh interface portproxy delete v4tov4 listenaddress=127.0.0.1 listenport=3070

Again, you'll need to adjust the specified listening address and port to match the values you're using. The rule will be deleted immediately and will no longer apply to new network requests.

Redirecting TCP/IP ports in Linux

With Linux you can choose between Port Forwarding and Port Redirection:

Port forwarding using iptables in Linux

First of all, you need to check if port forwarding is enabled or not on your server. For better understanding, I will be using eth0 as a reference interface and all command executions will be related to eth0 here.

To check if port forwarding is enabled in Linux, you can use the following:

```
# sysctl -a |grep -i eth0.forwarding
```

This is the result after executing the command:

```
net.ipv4.conf.eth0.forwarding = 0
net.ipv6.conf.eth0.forwarding = 0
```

Since both values are zero, port forwarding is disabled for ipv4 and ipv6 on the interface eth0.

Or you can use the process filesystem to check if port forwarding is enabled or not:

```
# cat /proc/sys/net/ipv4/conf/eth0/forwarding
```

Or, if you are using ipv6:

```
# cat /proc/sys/net/ipv6/conf/eth0/forwarding
```

Result after executing the command is 0 or 1.

Now we need to first enable port forwarding on our system, then we will configure the port forwarding rules in iptables.

```
# sysctl net.ipv4.conf.eth0.forwarding=1
```

or/and

```
# sysctl net.ipv6.conf.eth0.forwarding=1
```

To make it persistent over reboots, add parameters in /etc/sysctl.conf:

```
# echo "net.ipv4.conf.eth0.forwarding = 1">>/etc/sysctl.conf
# echo "net.ipv6.conf.eth0.forwarding = 1">>/etc/sysctl.conf
# sysctl -p
```

Now that we have port forwarding enabled on our server, we can go ahead with configuring the port forwarding rules using iptables.

Here I will forward port 3051 to port 3070 on 172.31.40.29 (or any IP).

```
# iptables -A PREROUTING -t nat -i eth0 -p tcp --dport 3051 -j DNAT --to
172.31.40.29:3070
```

```
# iptables -A FORWARD -p tcp -d 172.31.40.29 --dport 3070 -j ACCEPT
```

The first command tells us to redirect packets coming to port 80 to IP 172.31.40.29 on port 8080. Now the packet also needs to go through the FORWARD chain, so we allow this in in the second command.

The command to verify the port forwarding rules is:

```
# iptables -t nat -L -n -v
```

To save the iptables rules and make them persistent over reboots use the command below:

```
# iptables-save
```

https://www.ibexpert.net/ibe/pmwiki.php?n=Doc.PortRedirection?action=edit

redirection using iptables in Linux

Our requirement is to redirect port 3051 to port 3070 on the same server. This can be done by adding rules in the PREROUTING chain. So run the command below:

```
# iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 3051 -j REDIRECT --
to-port 3070
```

If you have an interface name other than eth0 then you need to edit your command accordingly. You can even add your source and destinations as well in same command using –src and –dst options. Without them, it's assumed to be any source and any destination.

Verify the port redirect rule in iptables using the following command:

```
# iptables -t nat -L -n -v
```

To save iptables rules and make them persistent over reboots use below command:

```
# iptables-save
```

Multiple Firebird instances on Linux

I have set-up Ubuntu 20.04.3 LTS as my test environment, some other Linux distro's may have different installation directions.

Install packages

```
sudo apt-get -y install libncurses5 libtommath1
sudo ln -s libtommath.so.1 /usr/lib/x86_64-linux-gnu/libtommath.so.0
wget -O-
```

```
https://github.com/FirebirdSQL/firebird/releases/download/v4.0.1/Firebird-4.
0.1.2692-0.amd64.tar.gz|tar -zxC /tmp
```

As a result, you will have all Firebird installation files ready in /tmp/.

And then install Firebird:

```
cd /tmp/Firebird-4.0.1.2692-0.amd64
sudo ./install.sh
```

Check that Firebird is running - the command below will show all processes related with Firebird or running under the user firebird:

```
ps aux | grep firebird
```

To install a secondary instance, or any other, just execute

```
sudo ./install.sh -path <install path>
```

or, in our case here:

```
sudo ./install.sh -path /opt/firebird3051
```

Using install.sh, the script services files will be created automatically; they just need to be enabled and activated afterwards:

/lib/systemd/system/firebird.service is the default service, and enabled to start on boot. However, using the install.sh script, you will find files like lib/systemd/system/ firebird.opt_firebird3051.service or any other name you specified as the install path above.

To check the service status, you can execute the following:

```
sudo systemctl status firebird.opt_firebird3051.service
```

Default behavior is that the service is disabled:

```
firebird.opt_firebird3051.service - Firebird Database Server
     Loaded: loaded (/lib/systemd/system/firebird.opt_firebird3051.service;
disabled; vendor preset: enabled)
     Active: inactive (dead)
       Docs: https://firebirdsql.org/en/firebird-rdbms/
```

To enable the service, execute:

```
sudo systemctl enable firebird.opt_firebird3051.service
```

Alternatively, if you also wish to enable and start the service at the same time you may execute:

```
sudo systemctl enable —-now firebird.opt_firebird3051.service
```

Don’t forget to specify RemoteServicePort in the corresponding firebird.conf.

Alternatively, you can manually copy /opt/firebir to /opt/firebird3053 for example:

```
sudo mkdir /opt/firebird3053
sudo cp -rf /opt/firebird/* /opt/firebird3053/
```

Now set the correct port now using:

```
sudo nano firebird.conf
```

And then copy firebird.service to the new service:

```
cd /lib/systemd/system/
cp firebird.service firebird.opt_firebird3053.service
```

Change the entry inside that file from

```
ExecStart=/opt/firebird/bin/fbguard -daemon -forever
```

to

```
ExecStart=/opt/firebird3053/bin/fbguard -daemon -forever
```

Don’t forget to apply correct file permissions and allow the user firebird from the group firebird to execute Firebird:

```
sudo chown firebird:firebird *
```

Now you can enable the service and start it as explained above:

```
sudo systemctl enable –-now firebird.opt_firebird3053.service
```

Conclusion

Firebird is quite flexible with different ideas for improving security and enabling multiple instances to work together, as you can see above. This information will help you to improve privacy and make access to your data much more secure. And with multiple instances you can fine-tune your server configuration or test new Firebird versions.

Back to top of page