Tech Tip #16: Firewalls and Traffic Monitors

Overview: Firewalls and examining database traffic with Wireshark and other tools

As a client/server environment, the PSQL database relies on good communications from the application running on the workstation to the engine running on the server. This communications is all handled over a computer network, which may be wired or wireless, simple or complex.

Most networks consist of multiple devices. All requests start out at the application, which then communicate with the PSQL Client on the workstation, which then communicates with the local operating system, which then talks to the network card driver, which talks to the network card itself, which talks to a network cable (or wireless medium). From there, the communications channel can go through hubs, switches, routers, copper cables, fiber cables, wireless spectrum, and more. Eventually, the request gets to the server, where it is passed up through the NIC into the NIC driver, to the PSQL communications module, and then *finally* to the database engine. Whew.

In some cases, firewalls installed on the clients, servers, and inside the network itself can also get in the way, either slowing traffic down or blocking it entirely. This only makes the problem harder to figure out. The database components will normally report communications problems with status codes in the 3000 range. Luckily, there are easy things to check when you are having communications problems.

The first thing we want to do is verify that we can **PING** the server by name. Here is a simple example:

```
C:\>ping psqlserver
Pinging psqlserver.empire.goldstarsoftware.com [192.168.1.9] with 32 bytes of data:
Reply from 192.168.1.9: bytes=32 time<1ms TTL=128
Reply from 192.168.1.9: bytes=32 time=1ms TTL=128
Reply from 192.168.1.9: bytes=32 time<1ms TTL=128
Reply from 192.168.1.9: bytes=32 time=1ms TTL=128
```

We are actually looking for two critical items here. First, the server name has to be translated into the correct IP address. If you're not sure, go to the server and run IPCONFIG to see what IP addresses are shown there. Second, we want to make sure that this basic communications is working. If this fails for either reason, fix the network (or name resolution) so that it works before anything else.

The second thing we want to check is whether we can communicate to the database TCP ports, which is 3351 (Btrieve) and 1583 (SQL/ODBC). We can do this with the **TELNET** tool. Sadly, Microsoft has opted to remove this useful troubleshooting tool from a default Windows installation, so you may need to install it manually before using it. [I suppose when the OS tops 60GB, saving this 80K is so important.] Anyway, from a command line, simply execute `TELNET <servername> 3351` (or `1583`) and verify that you do not get an error. (You should get a blank screen, which you can simply close.) If **PING** works, but you get an error from **TELNET**, then the odds are that a firewall is blocking traffic. Again, disable the firewall, or open up the needed ports, and you may find it works just fine now.

Once you have traffic getting to the server, you may want to observe the details of that traffic. This is where a network analyzer, such as **Wireshark**, can help you. A network analyzer can capture network traffic and display it so that you can see what is happening. The built-in timestamps tell you how long it took to process each request (or how slow the network is), and the data being sent can be easily deciphered from the resulting data buffers, making it possible to observe a process in its entirety. This is especially useful if you have a long-running process and you want to find out which piece is taking the longest to run. You can download a current version copy of **Wireshark** or obtain additional information about it from the web site http://www.wireshark.org. If you want more information about the TCP communications process, we strongly recommend books and training sessions from **Laura Chappell** (http://www.packet-level.com/) as a great starting point.

Reading data from a network trace with **Wireshark** is not too hard if you know what you are doing, but it can be mind-numbing at times as you stare at a sea of hexadecimal digits. This is the primary reason why Goldstar Software created the tools **SQLInterceptor** and **BtrvInterceptor**. These command-line tools take data from the network wire (or from a saved PCAP file) and extract just the SQL/ODBC (TCP port 1583) or Btrieve (TCP port 3351) packets and display information about what was contained within the packets. This can actually allow you to see database requests going to the server in real time!

Here's a short example of a SQLInterceptor trace on a few simple queries from the PCC:

```
C:\>sqlinterceptor /n
SQLInterceptor Version 1.23: 11/14 (C)2013 Goldstar Software Inc.
Registered to Goldstar Software Inc. (Site License GS)
1. \Device\NPF_{5B93960A-1335-4212-9AF2-D4F6B3677255} (VMware Virtual Ethernet Adapter)
2. \Device\NPF_{B3645EF7-CB31-415A-875C-6FEBA0C2F64D} (VMware Virtual Ethernet Adapter)
3. \Device\NPF_{7674243E-8237-4E53-B729-1CD8C4F54B46} (Broadcom NetLink (TM) Gigabit Ethernet)
Enter the interface number (1-3):3

SQLInterceptor is now listening on Broadcom NetLink (TM) Gigabit Ethernet...
Press Ctrl-C to stop monitoring.
11:38:48.769646 0127 Reqst 000C ExecDirect select * from "Person"
11:38:48.851614 1514 Reply 000C ExecDirect (0)
11:38:49.493377 0134 Reqst 0043 ExecDirect SELECT COUNT(*) from "Person"
11:38:49.495348 0171 Reply 0043 ExecDirect (0)
11:39:02.380488 0154 Reqst 004D ExecDirect select * from "Person" where First_Name = 'James'
11:39:02.382175 1514 Reply 004D ExecDirect (0)
```

Note that you can see the time it takes to process the query here with some simple subtraction. The last query took 1,687us (microseconds) to process, or just over 1 ½ millisecond. This tool can thus be used not only to see frequent queries, but also to find queries that need to be optimized for performance reasons. Additional command-line options allow you to display additional columns and types of requests (like the actual FETCH requests. The related tool, **BtrvInterceptor**, provides the same capability for the Btrieve interface. You can find these tools here:

http://www.goldstarsoftware.com/sqlinterceptor.asp
http://www.goldstarsoftware.com/btrvinterceptor.asp

Tools like this are a great boon for developers when it comes to performance analysis. You can use them to analyze your processes and see how much work is being done, and to look for ways to optimize the communications, which is often the slowest component (i.e. that with the highest latency) of any client/server environment. For example, these tools can show you that you are reading one byte at a time from an INI file, or even reading the same text file over and over again. It can show you which SQL queries are running slowly, and how many Btrieve records are read to satisfy an operation.

Sometimes, performance is best optimized through faster hardware, faster networks, and better configurations. However, a vast majority of performance issues are caused by inefficient code, mainly because developers test on small data sets. Reading 10,000 records for a process isn't bad, but if it take 1,000,000 reads to satisfy a process on a user's data set, it'll take a while. Optimizing the process to read ½ the data set may not sound like much, but it will save a LOT more time for the user!