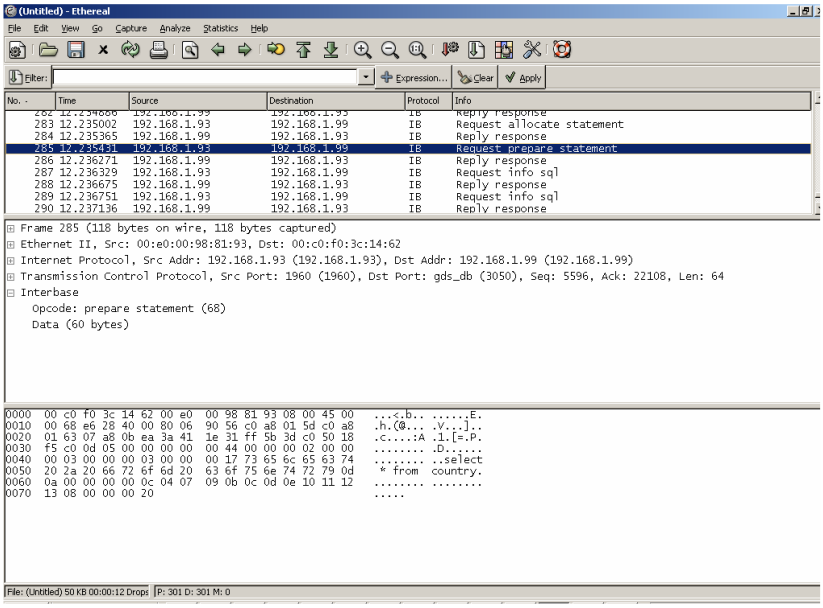
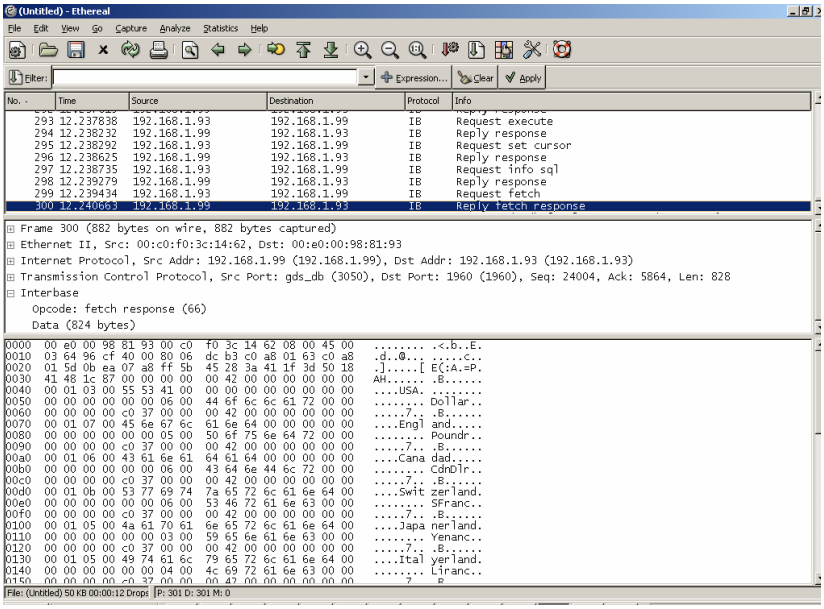
Secure connections to Firebird with Stunnel

A little background, I’ve recently worked for some customers very interested in securing their data and some quick tests showed that database connections where a weak point.

In an unsecured environment it is possible for an eavesdropper to collect all queries run and recordsets returned, as those Ethereal screenshots illustrate:



and the results are:



Thus the need to encrypt data running between database and clients, Googleing around for a prebuilt solution I stubled into a very nice article about securing VNC connections with Stunnel on Securityfocus.com, what follows is derived from that article (http://www.securityfocus.com/infocus/1677) and adapted to Firebird.

A little background about my environment, I'm running the webserver on 192.168.1.93 (from now on "the client") and Firebird on 192.168.1.99 (from now on "the server").

Basically what follows is a simple description of what I did, you can use it as a checklist, for screenshots and an handy batch file that creates the certification authority and the certificates you should really check the original article.
Anyway:

1. Download Stunnel and the two required libraries, libeay32.dll and libssl32.dll from www.stunnel.org
2. Put them in c:\program files\stunnel on the client and server pc
3. Obtain the client and server certificates from a certification authority (refer to the original document for instructions about doing this with OpenSSL and an handy DOS tool), basically you'll have to:

    a. Generate the CA's certificate with a command like:

```
openssl req -new -x509 -keyout C:\CA\private\CAkey.pem -out
C:\CA\CAcert.pem -config c:\progra~1\OpenSSL\openssl.conf
```

The parameters have the following meaning:

1. req: name of the certificate management utility

2. –new: generate a new certificate request
3. –x509: self sign the certificate request
4. –keyout *file*: the file in which the generated private key will be stored
5. –out *file*: the file in which the generated certificate will be stored
6. –config *file*: the OpenSSL configuration file to be used

Other parameters can be: -days *number*: the validity period of the generated certificate in days.

Remember the PEM passphrase you'll be asked to specify!!!

b. Generate the server's keys and certificate request with a command like:

```
openssl req -nodes -config C:\Progra~1\OpenSSL\openssl.conf -new -
newkey rsa:1024 -keyout C:\CA\temp\vnc_server\server.key -out
C:\CA\temp\vnc_server\server.req
```

The parameters have the following meaning:

1. –newkey rsa:1024: type (RSA) and length (1024) of the generated key
2. Other parameters have the same meaning as above

c. Sign the server's certificate:

```
openssl ca -config C:\Progra~1\OpenSSL\openssl.conf -policy
policy_anything -notext -in C:\CA\temp\vnc_server\server.req -out
C:\CA\temp\vnc_server\server.crt
```

The parameters have the following meaning:

1. –policy *policy_anything*: defines the policy to be used to determine a match between certificate and CA infos, the following name matches with the same section of the openssl.conf file
2. –notext:
3. –in: input request file
4. –out: output signed file

You will be asked for the passphrase used in setting up the CA's certificate

d. Generate the client's keys and certificate request with a command like:

```
openssl req -nodes -config C:\Progra~1\OpenSSL\openssl.conf -new -
newkey rsa:1024 -keyout C:\CA\temp\vnc_client\client.key -out
C:\CA\temp\vnc_client\client.req
```

e. Sign the client's certificate:

```
openssl ca -config C:\Progra~1\OpenSSL\openssl.conf -policy
policy_anything -notext -in C:\CA\temp\vnc_client\client.req -out
C:\CA\temp\vnc_client\client.crt
```

4. Now configure Stunnel on the server, by creating a c:\program files\stunnel\stunnel.conf file with the following content:

```
CAfile = CAcert.pem
CApath = certificates
cert = server.pem
client = no
verify = 3

[firebird]
accept = 192.168.1.99:443
connect = 127.0.0.1:3050
```

This will forward all connections to port 443/tcp to local port 3305/tcp only if valid certificates are present (verify = 3 requires valid certificates on both sides).
You can add a `debug = 7` after `verify = 3` to help debugging.

5. Now we have to copy CACert.pem (certification authority's certificate), server.pem (server's cerificate) to c:\program files\stunnel
6. Rename client certificate this way:
    a. Issue the following command: `openssl x509 -hash -noout -in client.crt`
    b. Note the value returned
    c. Rename client.crt to the *value returned*.0 (.0 instead of .crt)
7. Then copy the client's certificate to c:\program files\stunnel\certificates after renaming it
8. Change the client's stunnel.conf this way:

```
CAfile = CAcert.pem
CApath = certificates
cert = client.pem
client = yes
verify = 3
debug = 7

[firebird]
accept = 127.0.0.1:3050
```

```
connect = 192.168.1.99:443
```

9. Now for the client, copy CACert.pem and client.pem in the c:\program files\stunnel directory
10. Rename server certificate this way:
    a. Issue the following command: `openssl x509 -hash -noout -in server.crt`
    b. Note the value returned
    c. Rename client.crt to the *value returned*.0 (.0 instead of .crt)
11. Copy the server's certificate to c:\program files\stunnel\certificates after renaming it
12. Now start Stunnel on the server
13. Then start Stunnel on the client
14. Now connect to Firebird (I've tested the connection with an ASP page) as it was running on localhost (i.e. a connectionstring like `"DRIVER={Firebird/Interbase(r) Driver}; DBNAME=127.0.0.1:C:\Program Files\Firebird\Firebird_1_5\examples\EMPLOYEE.FDB; UID=SYSDBA; PWD=masterkey"`)
15. Now it's time to make Stunnel run as a service on Windows, go to a command prompt, cd to "c:\program files\stunnel" and type "stunnel-version here.exe –install" (more on this at http://www.wurd.com/cl_ssl_stunnel.php), in my case it was "stunnel-4.04.exe –install".
16. Re-test to check that nothing has broken
17. If you are interested in knowing more about Stunnel you can check http://www-106.ibm.com/developerworks/library/s-stun.html and more detailed informations about certificate generation and using third party certificates with Stunnel can be found at http://www.thefiengroup.com/downloads/teamware/StunnelOpenSSLIntegration.pdf .

While using Stunnel to secure connections is good to bind Firebird to local IP address (127.0.0.1) so that no one can bypass stunnel and attempt direct connections to Firebird, this is accomplished by modifying firebird.conf as follows:

```
.....

# Allows incoming connections to be bound to the IP address of a
# specific network card. It enables rejection of incoming connections
# through any other network interface except this one. By default,
# connections from any available network interface are allowed.
#
# Type: string
#
#RemoteBindAddress =
RemoteBindAddress = 127.0.0.1
.....
```

An attempt to connect directy from another host will end up in failure, with the database server actively refusing connection.

An incentive to use such a setup is awareness of the fact that Firebird only uses the first eight characters of the password, making brute force attacks easier.

Another important thing, from a security point of view at least is to set Firebird to run as an unprivileged user, preventing people that gained access to it from being easily able to compromise the whole system.

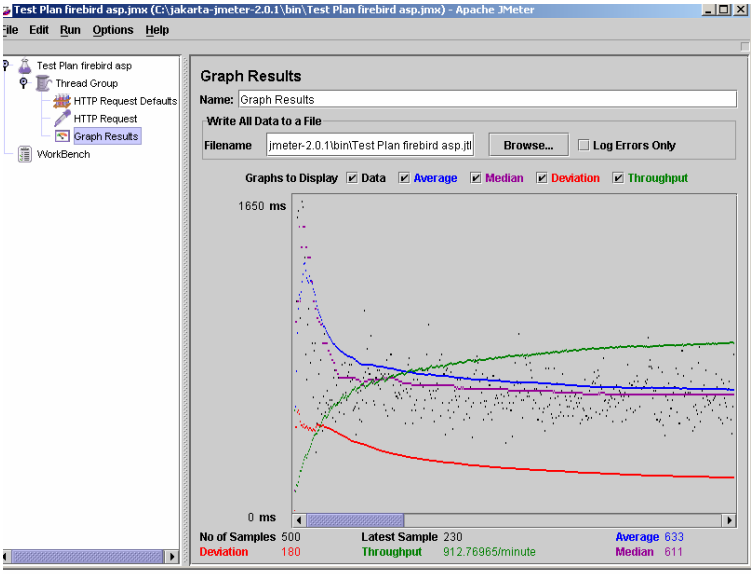
Benchmarking:

After setting up this advanced security tool I decided to check if it has an acceptable impact on performance, so I configured Apache Jmeter to run a query that returned a medium sized resultset against the standard EMPLOYEE.FDB database supplied with Firebird.

I just downloaded Apache's Jmeter (http://jakarta.apache.org/jmeter/index.html) which is a great java based stress test tool, it can be used to test a lot of services (Http, Https, FTP, Databases, LDAP, Webservices …). I set up a basic test plan according to the tutorial available on Jmeter's site and fired it, here are the results.

Benchmark results with direct connection (1255 queries per minute):

Benchmark results with stunnel (912 queries per minute):



So there is a little performance loss (more or less 27%), but I'm pretty shure that it is worthwile in many cases. Be aware that this benchmark results are for a web page that displays results of a query that goes to Firebird through an encrypted tunnel, so with a grater overhead than a straight Jdbc connection to Firebird through the same tunnel.