Progresswiz

Secure Communications with OpenEdge and SSL

PAUL KOUFALIS
PRESIDENT
PROGRESSWIZ CONSULTING

Progresswiz Consulting

- Based in Montréal, Québec, Canada
- Providing technical consulting in Progress®, UNIX, Windows, MFG/PRO and more
- Specialized in performance tuning, system availability and business continuity planning
- …and security of Progress-based systems

Progresswiz

Agenda

- Introduction
- Why encrypt?
- How to…
  - Start a server
  - Connect clients: ABL, JDBC and ODBC
- Questions

Progresswiz

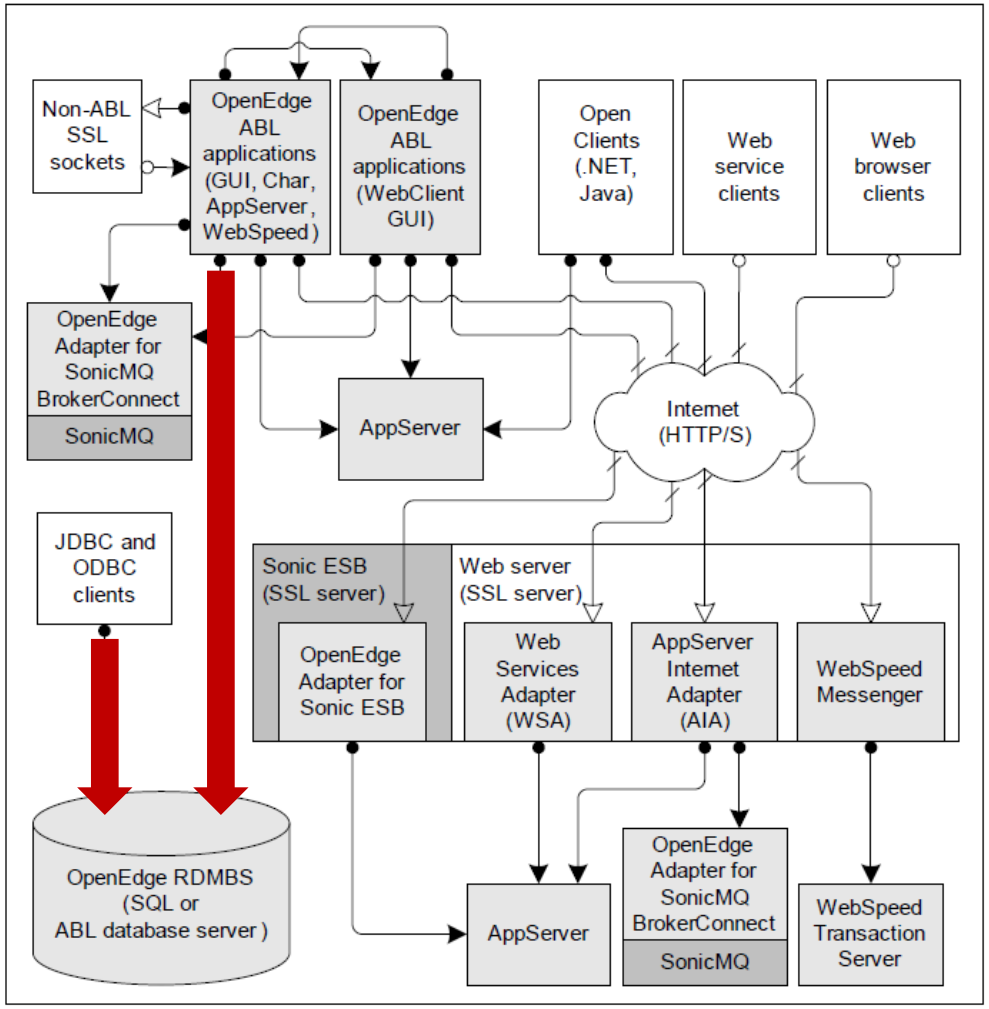
Introduction

- What is SSL?
  - *Secure Sockets Layer*
  - Encrypts communications between client and server
- OpenEdge SSL
  - SSL tunneling over TCP/IP

Progresswiz

Introduction

- OE supports SSL at multiple connection points (black dots and arrows)
- This presentation introduces two of these



Progresswiz

Types of Cryptography

- For OE SSL we concentrate on two:
  - Symmetric Key Cryptography
  - Asymmetric Key Cryptography
    - AKA public/private key cryptography
- Each type has advantages and disadvantages
  - Speed and security are the two biggies

Progresswiz

Symmetric Key Cryptography

- Based on a common key or password
- Both client and server must share same key
- DES, AES are some buzzwords you may have heard
  - Ex.: your WiFi connection
- Good for encrypting bulk data
- Hard to securely share key with random clients

Progresswiz

Asymmetric Key Cryptography

- Based on a public/private key pair
- Easier to deploy securely
  - Just give out the public key!
- More processor-intensive than symmetric key crypto
  - Not ideal for exchanging big chunks of data

Public/Private Keys Simple Explanation

- Based on a 3-way trust relationship:
  - Server tells client “I am Server X”
    - Here is my certificate as proof
- Client must validate certificate
- Enter trusted third party
  - The Certificate Authority (CA)

Progresswiz

Certificate Authority

- Independent third party trusted by client and service provider
  - RSA, Thawte, Verisign…
  - Or you can be your own CA (OpenSSL)
- “Server X” sends CA a “Certificate Request”
- CA returns a signed digital certificate
  - AKA the “public key”
  - This is the certificate that server gives to potential client to assert its identity

Progresswiz

Certificate Authority

- For client to trust server's certificate, he must trust the CA
- Client maintains a store of CA *root* certificates of his trusted CA's
  - Uses these root certificates to validate the server's digital certificate

Why Encrypt?

- Sensitive information
- Passwords

- Network sniffers are free and easy to use
  - Ex.: WireShark

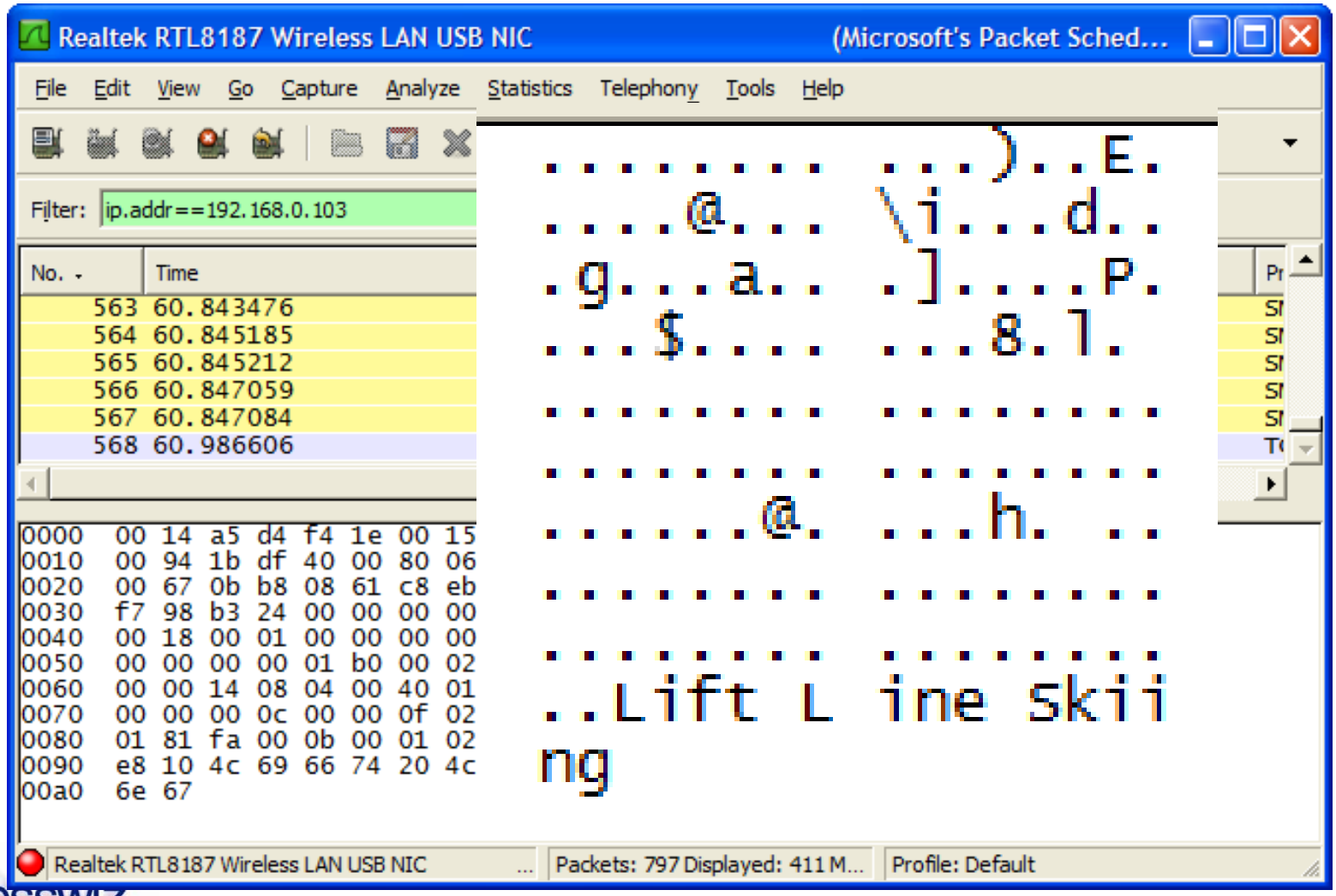
Progresswiz

Example

```
FOR EACH customer FIELDS(name):
  DISPLAY NAME.
END.
```

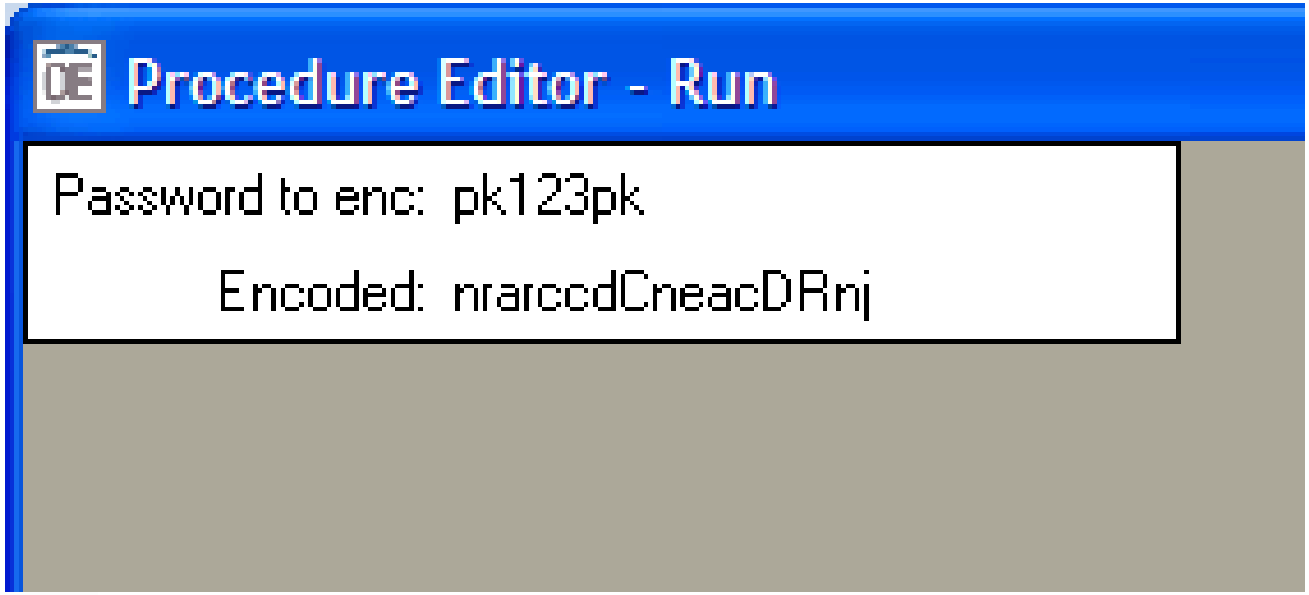
Progresswiz

WireShark

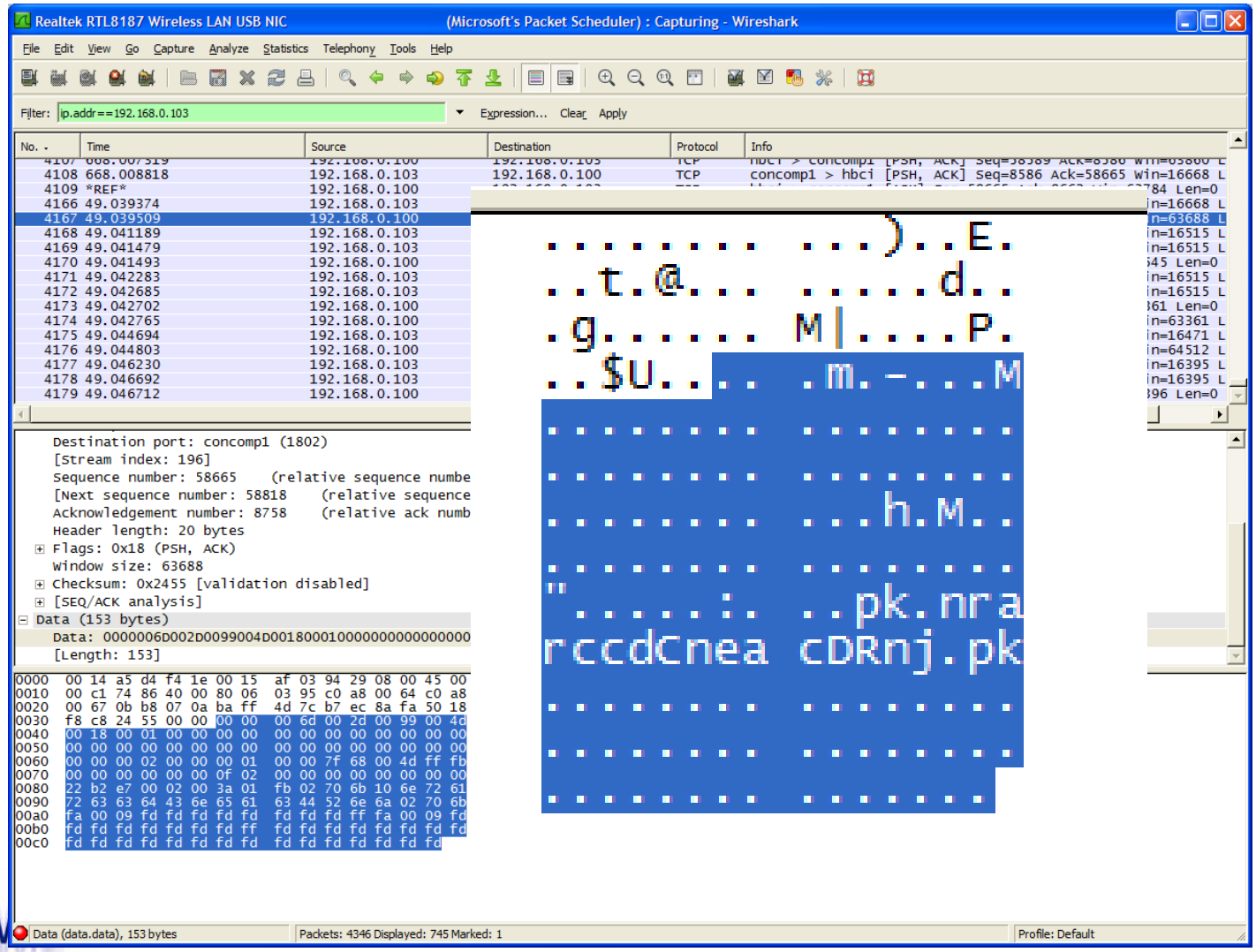


Example

- Encoded password comes back from server to client
- Client does the authentication itself



WireShark



WireShark – SQL Connections

- Sqlexp sends a weakly obfuscated password to _sqlsrv2
  - Includes real password length
- I did not take the time to break it – sorry!

OpenEdge Implementation

- OpenEdge uses public/private key crypto to initiate SSL connection
- Server and client exchange session-unique symmetric key
  - Still using asymmetric cryptography
- Data communications are then encrypted using symmetric key encryption
  - More suitable for bulk data encryption
  - Other key holders cannot decrypt exchanges

Progresswiz

Certificate Validation - Important Notes

- In the OpenEdge implementation:
  - The client validates the server identity
  - There is no mechanism for the server to validate the client's identity
  - There is no mechanism for the client to check if the CA revoked the server's digital certificate

Progresswiz

Certificate and Key Management

- OpenEdge manages key in $DLC
  - $DLC/keys
  - $DLC/certs (CA certificates)
- Available OE tools:
  - $DLC/bin/pkiutil
  - $DLC/bin/certutil
  - $DLC/java/jdk/bin/keytool

Progresswiz

Default Key

- OE provides a test key “default_server”
  - Not for use in production

```
Keystore entry: default_server
Certificate:
subject= /C=US/ST=NH/O=Progress Software
   Corporation/OU=Server Technologies/CN=Default
   Progress SSL Server
issuer= /C=US/ST=NH/O=Progress Software
   Corporation/OU=Server Technologies/CN=Progress Server
   Certificate Authority
notBefore=Feb 25 22:04:12 2004 GMT
notAfter=Feb 22 22:04:12 2014 GMT
```

Progresswiz

JDBC/ODBC

- Non-OE clients must manage their own certificates
  - Java: keytool
  - ODBC: Certificate file
  - .Net: Microsoft Certificate Store Mgmt

Progresswiz

Procedure to Enable SSL

- Create the key request
- Have request signed by CA
- Import signed certificate
- Start database
- Connect clients
  - ABL, JDBC, ODBC

Progresswiz

Create New Request

```
C:\apps\openedge\wrk102a>pkiutil -newreq eupug
Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
......++++++++++++
writing new private key to 'C:\apps\openedge\oe102a/keys/requests/eupug.pk1'

Country Name (2 letter code) [US]:CA
State or Province Name (full name) []:QC
Locality Name (eg, city) []:
Organization Name (eg, company) []:Progresswiz
Organizational Unit Name (eg, section) []:
Server DNS name []:pckoup

You may now use the file C:\apps\openedge\oe102a/keys/requests/eupug.pk10 to
request a new Digital Certificate from a CA Certificate Authority.

After you obtain the new Digital Certificate from the CA use the -import
   command to insert the certificate into the keystore.
```

Progresswiz

Sign Request

- Send .pk10 file to CA
- I am my own CA so I signed request myself:

```
C:\apps\OE_WRK\SSL>openssl ca -config pk_ssl.cnf
    -notext -out eupug.crt -infiles
    c:\apps\OE102A03\keys\requests\eupug.pk10

<details snipped...>

Certificate is to be certified until Nov 1 02:19:29
   2011 GMT (365 days)
Sign the certificate? [y/n]:y
```

Progresswiz

Import Signed Certificate

- On server side

```
C:\apps\OE_WRK\SSL> pkiutil -import eupug eupug.crt
Importing private key alias eupug:
Importing certificate file eupug.crt

Enter keystore password to alias eupug:
```

Validate Import

```
C:\apps\openedge\wrk102a>pkiutil –list

Keystore entry: gupq
Certificate:
subject= /C=CA/ST=QC/O=Progresswiz/CN=pckoup
issuer= /C=CA/ST=Some-State/O=Internet Widgits Pty Ltd
notBefore=Feb  1 02:19:29 2010 GMT
notAfter=Feb  1 02:19:29 2011 GMT
```

Progresswiz

Import CA Certificate on Client

- Only if not using one of the standard certificates

```
C:\apps\openedge\wrk102a\ssl>certutil -import
   ..\sslkeys\pkca.crt

Importing trusted certificate to alias name: 39d36856
```

- New file $DLC/certs/39d36856.0

Progresswiz

Distribute CA certificate

- Again – only if not using standard CA certificate
  - Remember – OE ships with a number of root CA certificates
- Every client must import the new CA certificate
  - OE client with certutil -import

Progresswiz

Passwords

- When the signing request was created a password was entered
- Use genpassword to encrypt that password

```
$DLC/bin/genpassword –password toto
243d3b28
```

Start the Database

```
C:\apps\openedge\wrk102a\ssl>_mprosrv ssl -H
   pckoup -S 5000 -ssl -keyalias eupug
-keyaliaspasswd 37273f36
```

- In the db.lg file:

```
SSL Encryption has been enabled for ALL TCP/IP
   connections to this database
SSL Key Alias Name (-keyalias): gupq
```

Warning

- Impossible to mix encrypted and non-encrypted client/server brokers
- All brokers will start using the same SSL key

Progresswiz

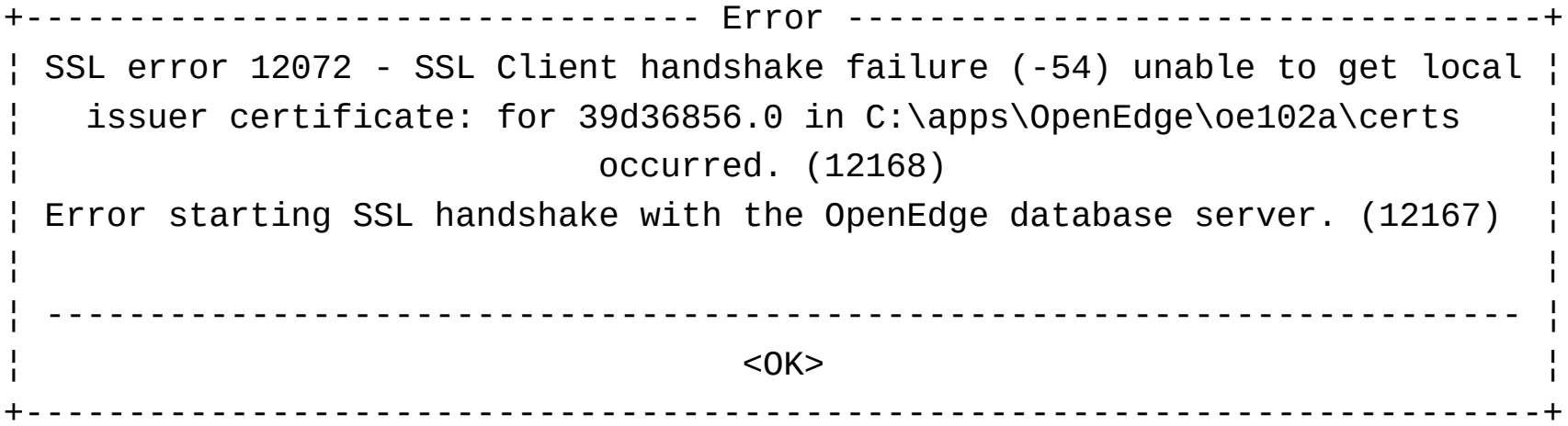
Connecting an ABL Client

```
$DLC/bin/_progres ssldb –H pckoup –S 5000
```

- Note – no need to specify –ssl
  - The server tells the client at connection time
- Nothing in the db.lg confirms that the connection is SSL-enabled
  - Only the broker startup message

Error Messages

- Missing the CA certificate?



- DB.lg
  - “Usernum 1 terminated abnormally”

Demo

- OpenEdge
  - Starting a server
  - Connecting an ABL client

JDBC

- Create a Java keystore
  - Use $DLC/jdk/bin/keytool

```
C:\apps\openedge\wrk102a\sslkeys>keytool -import -alias
   ca -file pkca.crt -keypass ca -keystore eupugstore -
   storepass eupug123
<snip...>
Trust this certificate? [no]:  y
Certificate was added to keystore
```

- Keystore file created

Test JDBC Connection

- Using Squirrel SQL client

```
jdbc:datadirect:openedge://localhost:5000;databas
   eName=ssl;EncryptionMethod=ssl;Truststore=c:\ap
   ps\openedge\wrk102a\sslkeys\gupqstore;TrustStor
   ePassword=gupq123
```

Progresswiz

Test JDBC Connection

- Without SSL parameters in URL:

```
SSL_102A_NOSSLPARAMS: [DataDirect][OpenEdge JDBC
  Driver]SSL Mismatch.  Encryption method in
  client and server must match.
```

Demo

- Connecting JDBC client

Progresswiz

ODBC Client

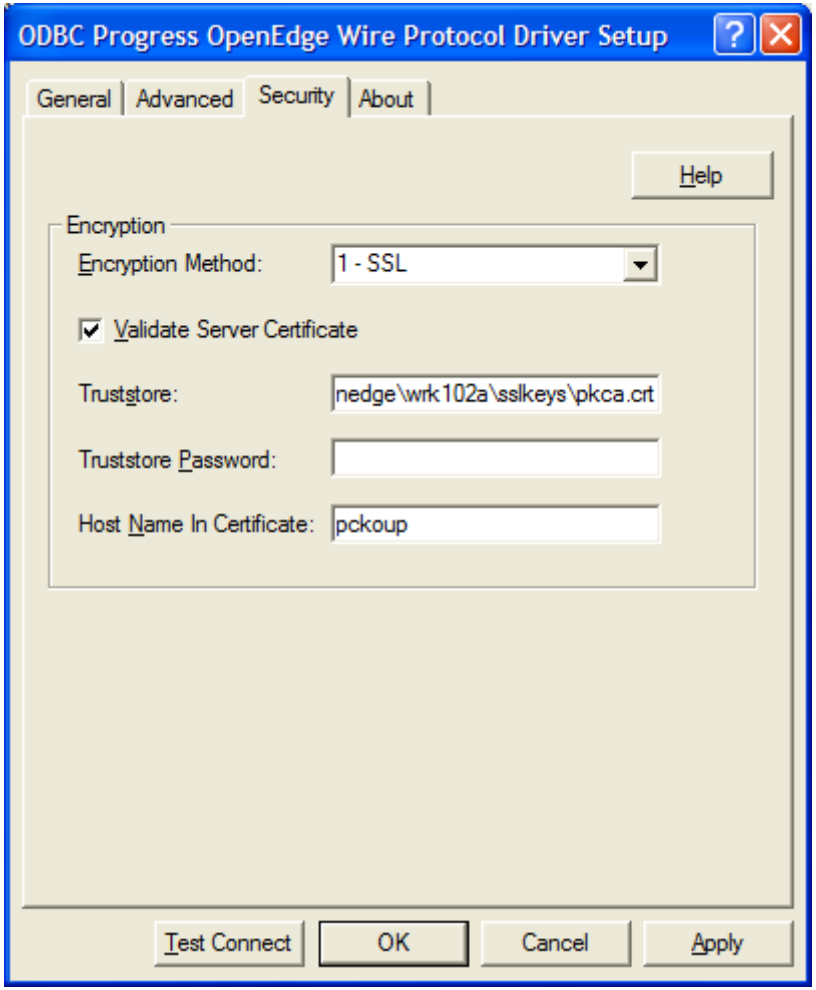
- A little more complicated
- Very little information in documentation
- Nothing in KB (last time I checked)

ODBC Client

- OE 10.1C SP4 and later
  - Before that buggy
- Copy pgcrypto.dll and pgss123.dll to %WINDIR%
  - Not sure if fixed in 10.2B
- Create DSN in ODBC Administrator

ODBC DSN

- Create DSN as usual
- Specify SSL encryption
- Full path of CA certificate in “TrustStore” field
- No password necessary



Progresswiz

Questions?

Progresswiz

More Questions or Comments?

- Email me at pk@progresswiz.com
- Presentations, tools and more available at

www.progresswiz.com

Progresswiz