

The logo for Progresswiz, featuring the word "Progresswiz" in a blue, sans-serif font. The letters are slightly shadowed, giving it a 3D effect. The logo is positioned in the upper right corner of the slide.

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

Agenda

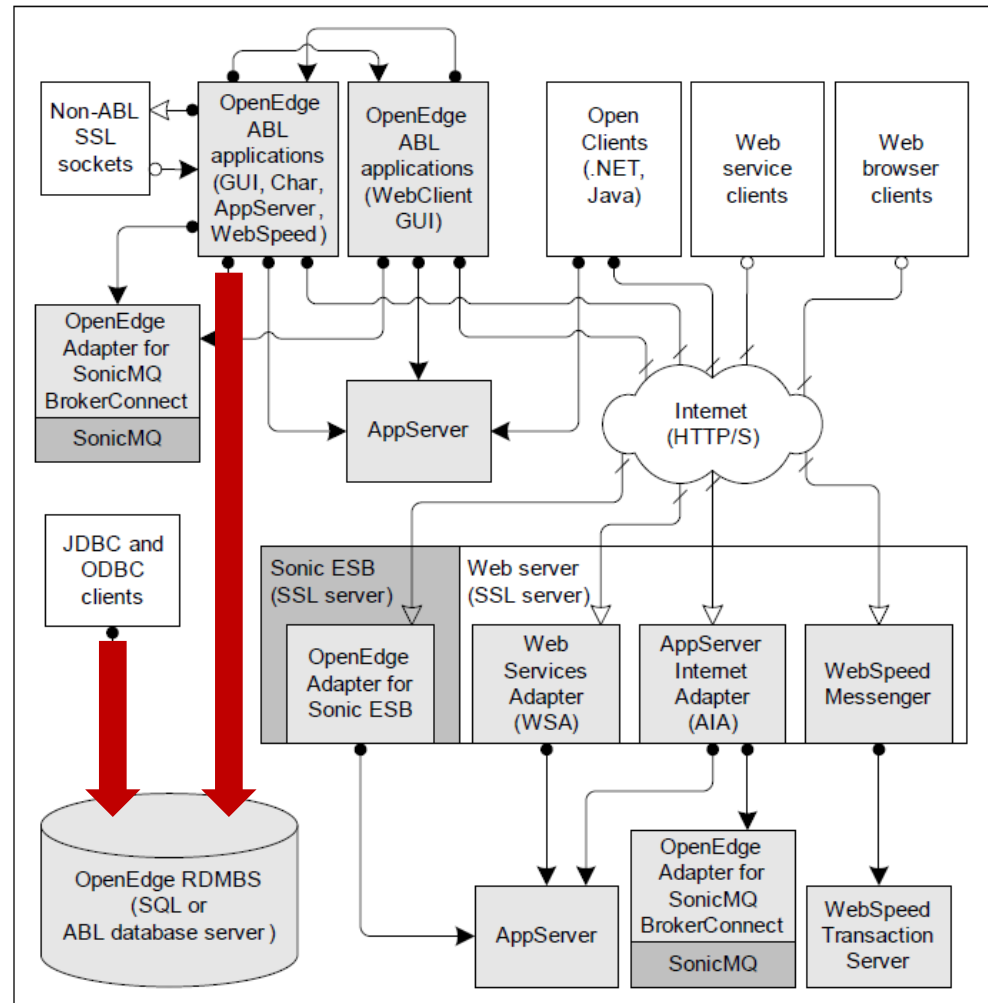
- Introduction
- Why encrypt?
- How to...
 - Start a server
 - Connect clients: ABL, JDBC and ODBC
- Questions

Introduction

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

Introduction

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



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

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

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)

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

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

Example

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

WireShark

Realtek RTL8187 Wireless LAN USB NIC (Microsoft's Packet Sched...)

File Edit View Go Capture Analyze Statistics Telephony Tools Help

Filter: ip.addr == 192.168.0.103

No.	Time
563	60.843476
564	60.845185
565	60.845212
566	60.847059
567	60.847084
568	60.986606

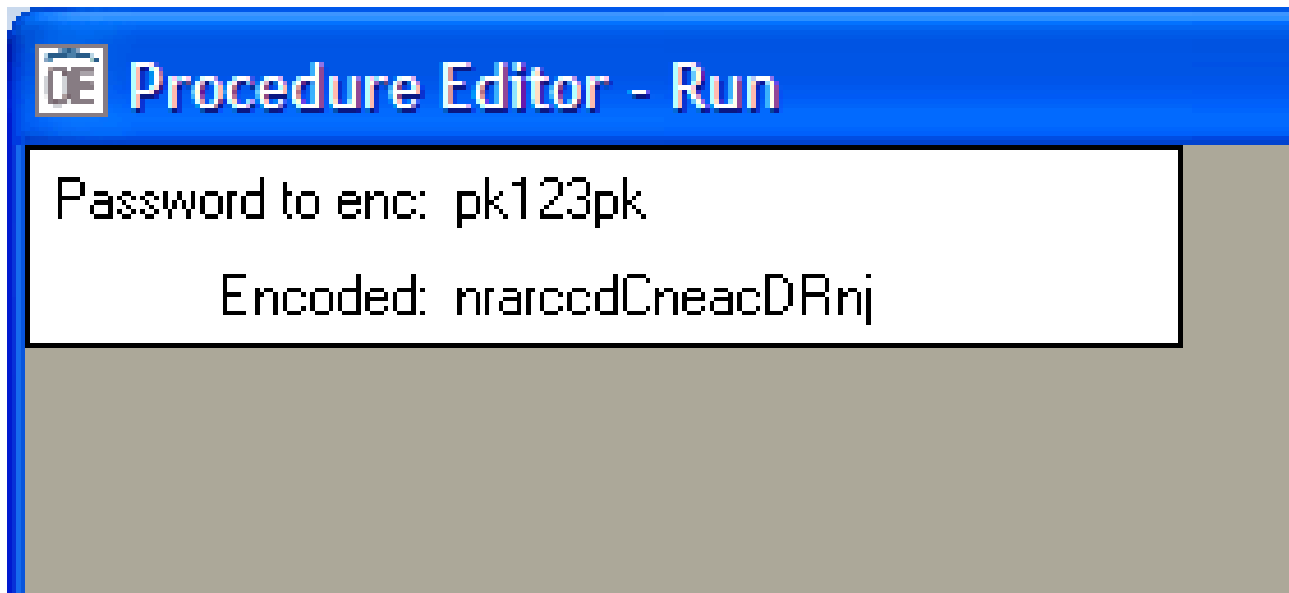
```
0000 00 14 a5 d4 f4 1e 00 15
0010 00 94 1b df 40 00 80 06
0020 00 67 0b b8 08 61 c8 eb
0030 f7 98 b3 24 00 00 00 00
0040 00 18 00 01 00 00 00 00
0050 00 00 00 00 01 b0 00 02
0060 00 00 14 08 04 00 40 01
0070 00 00 00 0c 00 00 0f 02
0080 01 81 fa 00 0b 00 01 02
0090 e8 10 4c 69 66 74 20 4c
00a0 6e 67
```

g a] E .
@ . \ i . . d . .
\$ P .
. 8 . l .
. @ h .
. Lift Line skiing
ng

Realtek RTL8187 Wireless LAN USB NIC ... Packets: 797 Displayed: 411 M... Profile: Default

Example

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



WireShark

Realtek RTL8187 Wireless LAN USB NIC (Microsoft's Packet Scheduler) : Capturing - Wireshark

File Edit View Go Capture Analyze Statistics Telephony Tools Help

Filter: ip.addr==192.168.0.103 Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Info
4107	008.007319	192.168.0.100	192.168.0.103	TCP	hbcj > concomp1 [PSH, ACK] Seq=38389 Ack=8380 win=63688 L
4108	668.008818	192.168.0.103	192.168.0.100	TCP	concomp1 > hbcj [PSH, ACK] Seq=8586 Ack=58665 win=16668 L
4109	*REF*	192.168.0.100	192.168.0.103	TCP	hbcj > concomp1 [PSH, ACK] Seq=8380 Ack=38389 win=63688 L
4166	49.039374	192.168.0.103	192.168.0.100	TCP	concomp1 > hbcj [PSH, ACK] Seq=8586 Ack=58665 win=16668 L
4167	49.039509	192.168.0.100	192.168.0.103	TCP	hbcj > concomp1 [PSH, ACK] Seq=38389 Ack=8380 win=63688 L
4168	49.041189	192.168.0.103	192.168.0.100	TCP	concomp1 > hbcj [PSH, ACK] Seq=8586 Ack=58665 win=16668 L
4169	49.041479	192.168.0.103	192.168.0.100	TCP	concomp1 > hbcj [PSH, ACK] Seq=8586 Ack=58665 win=16668 L
4170	49.041493	192.168.0.100	192.168.0.103	TCP	hbcj > concomp1 [PSH, ACK] Seq=38389 Ack=8380 win=63688 L
4171	49.042283	192.168.0.103	192.168.0.100	TCP	concomp1 > hbcj [PSH, ACK] Seq=8586 Ack=58665 win=16668 L
4172	49.042685	192.168.0.103	192.168.0.100	TCP	concomp1 > hbcj [PSH, ACK] Seq=8586 Ack=58665 win=16668 L
4173	49.042702	192.168.0.100	192.168.0.103	TCP	hbcj > concomp1 [PSH, ACK] Seq=38389 Ack=8380 win=63688 L
4174	49.042765	192.168.0.100	192.168.0.103	TCP	hbcj > concomp1 [PSH, ACK] Seq=38389 Ack=8380 win=63688 L
4175	49.044694	192.168.0.103	192.168.0.100	TCP	concomp1 > hbcj [PSH, ACK] Seq=8586 Ack=58665 win=16668 L
4176	49.044803	192.168.0.100	192.168.0.103	TCP	hbcj > concomp1 [PSH, ACK] Seq=38389 Ack=8380 win=63688 L
4177	49.046230	192.168.0.103	192.168.0.100	TCP	concomp1 > hbcj [PSH, ACK] Seq=8586 Ack=58665 win=16668 L
4178	49.046692	192.168.0.103	192.168.0.100	TCP	concomp1 > hbcj [PSH, ACK] Seq=8586 Ack=58665 win=16668 L
4179	49.046712	192.168.0.100	192.168.0.103	TCP	hbcj > concomp1 [PSH, ACK] Seq=38389 Ack=8380 win=63688 L

Destination port: concomp1 (1802)
[Stream index: 196]
Sequence number: 58665 (relative sequence number)
[Next sequence number: 58818 (relative sequence number)]
Acknowledgement number: 8758 (relative acknowledgment number)
Header length: 20 bytes
Flags: 0x18 (PSH, ACK)
Window size: 63688
Checksum: 0x2455 [validation disabled]
[SEQ/ACK analysis]
Data (153 bytes)
data: 0000006002b0099004b001800010000000000000000
[Length: 153]

0000 00 14 a5 d4 f4 1e 00 15 af 03 94 29 08 00 45 00
0010 00 c1 74 86 40 00 80 06 03 95 c0 a8 00 64 c0 a8
0020 00 67 0b b8 07 0a ba ff 4d 7c b7 ec 8a fa 50 18
0030 f8 c8 24 55 00 00 00 00 00 6d 00 2d 00 99 00 4d
0040 00 18 00 01 00 00 00 00 00 00 00 00 00 00 00
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0060 00 00 00 02 00 00 01 00 00 7f 68 00 4d ff fb
0070 00 00 00 00 00 0f 02 00 00 00 00 00 00 00 00
0080 22 b2 e7 00 02 00 3a 01 fb 02 70 6b 10 6e 72 61
0090 72 63 63 64 43 6e 65 61 63 44 52 6e 6a 02 70 6b
00a0 fa 00 09 fd fd fd fd fd fd fd fd fd fd fd fd fd
00b0 fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
00c0 fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd

Data (data.data), 153 bytes Packets: 4346 Displayed: 745 Marked: 1 Profile: Default

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

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

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

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

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

Procedure to Enable SSL

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

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.

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 -infile
    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
```

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
```

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

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

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

Connecting an ABL Client

```
$DLC/bin/_progres sslldb -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?

```
+----- Error -----+
| SSL error 12072 - SSL Client handshake failure (-54) unable to get local |
| issuer certificate: for 39d36856.0 in C:\apps\OpenEdge\oe102a\certs |
|                               occurred. (12168) |
| Error starting SSL handshake with the OpenEdge database server. (12167) |
|-----|
|                               <OK> |
+-----+
```

■ DB.lg

- “Usernum 1 terminated abnormally”

- OpenEdge
 - Starting a server
 - Connecting an ABL client

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

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.
```

- Connecting JDBC client

ODBC Client

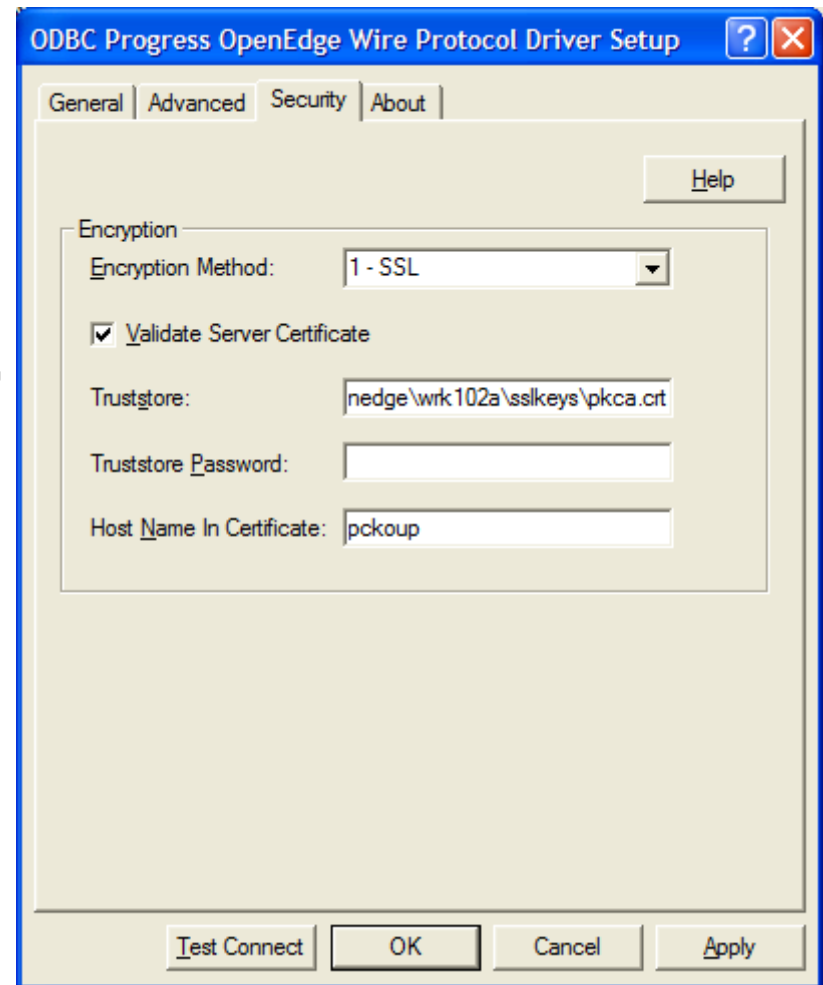
- A little more complicated
- Very little information in documentation
- Nothing in KB (last time I checked)

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



Questions?



More Questions or Comments?

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

www.progresswiz.com