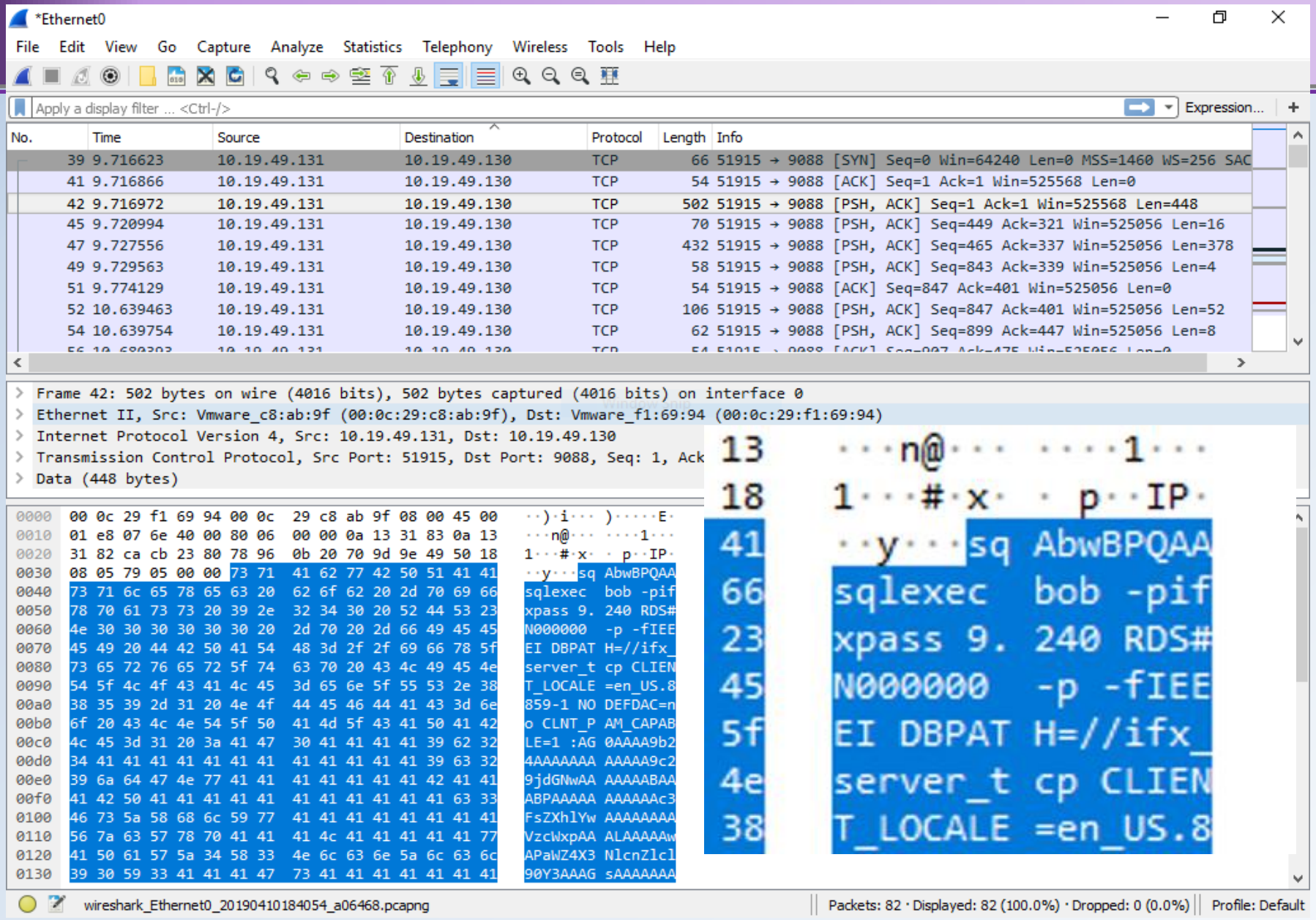
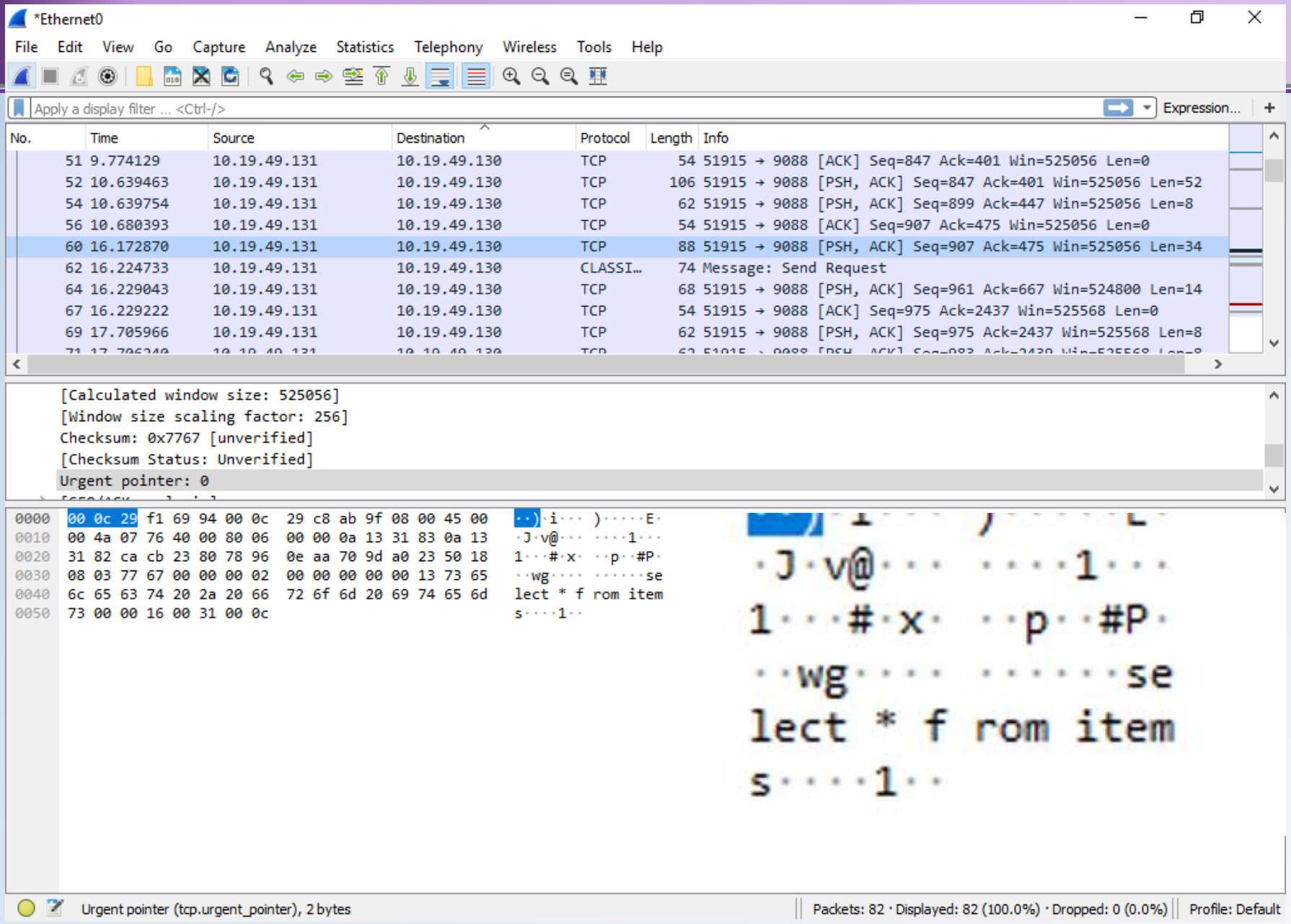
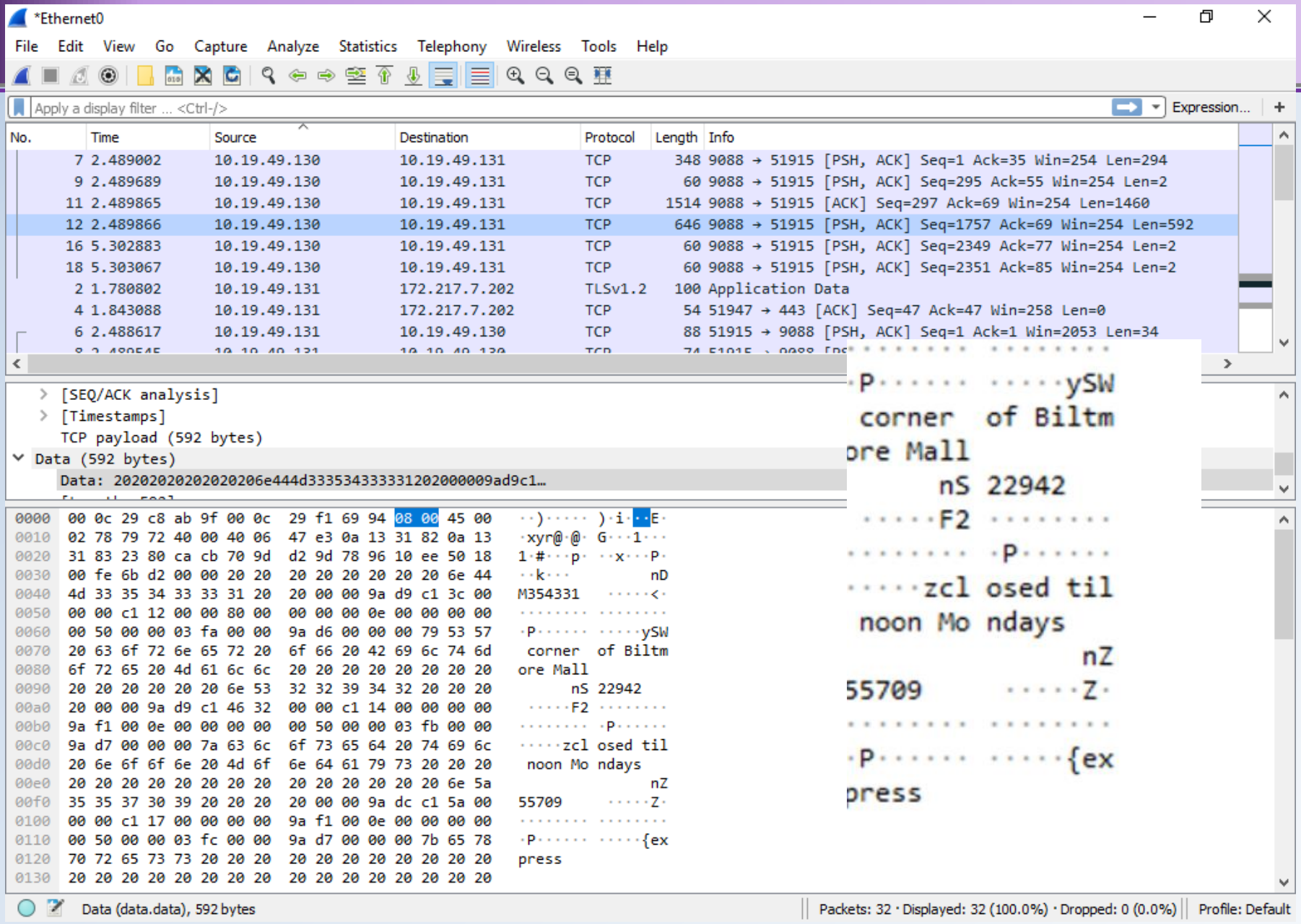
Securing Informix Connections with SSL

Thomas Beebe
tom@xdbsystems.com

11/21/2024


*Ethernet0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
No. Time Source Destination Protocol Length Info
39 9.716623 10.19.49.131 10.19.49.130 TCP 66 51915 → 9088 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SAC
41 9.716866 10.19.49.131 10.19.49.130 TCP 54 51915 → 9088 [ACK] Seq=1 Ack=1 Win=525568 Len=0
42 9.716972 10.19.49.131 10.19.49.130 TCP 502 51915 → 9088 [PSH, ACK] Seq=1 Ack=1 Win=525568 Len=448
45 9.720994 10.19.49.131 10.19.49.130 TCP 70 51915 → 9088 [PSH, ACK] Seq=449 Ack=321 Win=525056 Len=16
47 9.727556 10.19.49.131 10.19.49.130 TCP 432 51915 → 9088 [PSH, ACK] Seq=465 Ack=337 Win=525056 Len=378
49 9.729563 10.19.49.131 10.19.49.130 TCP 58 51915 → 9088 [PSH, ACK] Seq=843 Ack=339 Win=525056 Len=4
51 9.774129 10.19.49.131 10.19.49.130 TCP 54 51915 → 9088 [ACK] Seq=847 Ack=401 Win=525056 Len=0
52 10.639463 10.19.49.131 10.19.49.130 TCP 106 51915 → 9088 [PSH, ACK] Seq=847 Ack=401 Win=525056 Len=52
54 10.639754 10.19.49.131 10.19.49.130 TCP 62 51915 → 9088 [PSH, ACK] Seq=899 Ack=447 Win=525056 Len=8
Frame 42: 502 bytes on wire (4016 bits), 502 bytes captured (4016 bits) on interface 0
Ethernet II, Src: Vmware_c8:ab:9f (00:0c:29:c8:ab:9f), Dst: Vmware_f1:69:94 (00:0c:29:f1:69:94)
Internet Protocol Version 4, Src: 10.19.49.131, Dst: 10.19.49.130
Transmission Control Protocol, Src Port: 51915, Dst Port: 9088, Seq: 1, Ack
Data (448 bytes)
0000 00 0c 29 f1 69 94 00 0c 29 c8 ab 9f 08 00 45 00
0010 01 e8 07 6e 40 00 80 06 00 00 0a 13 31 83 0a 13
0020 31 82 ca cb 23 80 78 96 0b 20 70 9d 9e 49 50 18
0030 08 05 79 05 00 00 73 71 41 62 77 42 50 51 41 41
0040 73 71 6c 65 78 65 63 20 62 6f 62 20 2d 70 69 66
0050 78 70 61 73 73 20 39 2e 32 34 30 20 52 44 53 23
0060 4e 30 30 30 30 30 30 20 2d 70 20 2d 66 49 45 45
0070 45 49 20 44 42 50 41 54 48 3d 2f 2f 69 66 78 5f
0080 73 65 72 76 65 72 5f 74 63 70 20 43 4c 49 45 4e
0090 54 5f 4c 4f 43 41 4c 45 3d 65 6e 5f 55 53 2e 38
00a0 38 35 39 2d 31 20 4e 4f 44 45 46 44 41 43 3d 6e
00b0 6f 20 43 4c 4e 54 5f 50 41 4d 5f 43 41 50 41 42
00c0 4c 45 3d 31 20 3a 41 47 30 41 41 41 41 39 62 32
00d0 34 41 41 41 41 41 41 41 41 41 41 41 39 63 32
00e0 39 6a 64 47 4e 77 41 41 41 41 41 41 42 41 41
00f0 41 42 50 41 41 41 41 41 41 41 41 41 41 41 63 33
0100 46 73 5a 58 68 6c 59 77 41 41 41 41 41 41 41 41
0110 56 7a 63 57 78 70 41 41 41 4c 41 41 41 41 41 77
0120 41 50 61 57 5a 34 58 33 4e 6c 63 6e 5a 6c 63 6c
0130 39 30 59 33 41 41 41 47 73 41 41 41 41 41 41 41
13 ···n@··· ····1···
18 1···#·x· · p··IP·
41 ··y···sq AbwBPQAA
66 sqlexec bob -pif
23 xpass 9. 240 RDS#
45 N000000 -p -fIEE
5f EI DBPAT H=//ifx_
4e server_t cp CLIEN
38 T_LOCALE =en_US.8
wireshark_Ethernet0_20190410184054_a06468.pcapng
Packets: 82 · Displayed: 82 (100.0%) · Dropped: 0 (0.0%)
Profile: Default


*Ethernet0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
No. Time Source Destination Protocol Length Info
51 9.774129 10.19.49.131 10.19.49.130 TCP 54 51915 → 9088 [ACK] Seq=847 Ack=401 Win=525056 Len=0
52 10.639463 10.19.49.131 10.19.49.130 TCP 106 51915 → 9088 [PSH, ACK] Seq=847 Ack=401 Win=525056 Len=52
54 10.639754 10.19.49.131 10.19.49.130 TCP 62 51915 → 9088 [PSH, ACK] Seq=899 Ack=447 Win=525056 Len=8
56 10.680393 10.19.49.131 10.19.49.130 TCP 54 51915 → 9088 [ACK] Seq=907 Ack=475 Win=525056 Len=0
60 16.172870 10.19.49.131 10.19.49.130 TCP 88 51915 → 9088 [PSH, ACK] Seq=907 Ack=475 Win=525056 Len=34
62 16.224733 10.19.49.131 10.19.49.130 CLASSI… 74 Message: Send Request
64 16.229043 10.19.49.131 10.19.49.130 TCP 68 51915 → 9088 [PSH, ACK] Seq=961 Ack=667 Win=524800 Len=14
67 16.229222 10.19.49.131 10.19.49.130 TCP 54 51915 → 9088 [ACK] Seq=975 Ack=2437 Win=525568 Len=0
69 17.705966 10.19.49.131 10.19.49.130 TCP 62 51915 → 9088 [PSH, ACK] Seq=975 Ack=2437 Win=525568 Len=8
[Calculated window size: 525056]
[Window size scaling factor: 256]
Checksum: 0x7767 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
0000 00 0c 29 f1 69 94 00 0c 29 c8 ab 9f 08 00 45 00
0010 00 4a 07 76 40 00 80 06 00 00 0a 13 31 83 0a 13
0020 31 82 ca cb 23 80 78 96 0e aa 70 9d a0 23 50 18
0030 08 03 77 67 00 00 00 02 00 00 00 00 00 13 73 65
0040 6c 65 63 74 20 2a 20 66 72 6f 6d 20 69 74 65 6d
0050 73 00 00 16 00 31 00 0c
Urgent pointer (tcp.urgent_pointer), 2 bytes
Packets: 82 · Displayed: 82 (100.0%) · Dropped: 0 (0.0%)
Profile: Default


*Ethernet0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
No. Time Source Destination Protocol Length Info
7 2.489002 10.19.49.130 10.19.49.131 TCP 348 9088 → 51915 [PSH, ACK] Seq=1 Ack=35 Win=254 Len=294
9 2.489689 10.19.49.130 10.19.49.131 TCP 60 9088 → 51915 [PSH, ACK] Seq=295 Ack=55 Win=254 Len=2
11 2.489865 10.19.49.130 10.19.49.131 TCP 1514 9088 → 51915 [ACK] Seq=297 Ack=69 Win=254 Len=1460
12 2.489866 10.19.49.130 10.19.49.131 TCP 646 9088 → 51915 [PSH, ACK] Seq=1757 Ack=69 Win=254 Len=592
16 5.302883 10.19.49.130 10.19.49.131 TCP 60 9088 → 51915 [PSH, ACK] Seq=2349 Ack=77 Win=254 Len=2
18 5.303067 10.19.49.130 10.19.49.131 TCP 60 9088 → 51915 [PSH, ACK] Seq=2351 Ack=85 Win=254 Len=2
2 1.780802 10.19.49.131 172.217.7.202 TLSv1.2 100 Application Data
4 1.843088 10.19.49.131 172.217.7.202 TCP 54 51947 → 443 [ACK] Seq=47 Ack=47 Win=258 Len=0
6 2.488617 10.19.49.131 10.19.49.130 TCP 88 51915 → 9088 [PSH, ACK] Seq=1 Ack=1 Win=2053 Len=34
[SEQ/ACK analysis]
[Timestamps]
TCP payload (592 bytes)
Data (592 bytes)
Data: 20202020202020206e444d333534333331202000009ad9c1…
0000 00 0c 29 c8 ab 9f 00 0c 29 f1 69 94 08 00 45 00 ··)·····)·i···E·
0010 02 78 79 72 40 00 40 06 47 e3 0a 13 31 82 0a 13 ·xyr@·@· G···1···
0020 31 83 23 80 ca cb 70 9d d2 9d 78 96 10 ee 50 18 1·#···p· ··x···P·
0030 00 fe 6b d2 00 00 20 20 20 20 20 20 20 6e 44 ··k··· nD
0040 4d 33 35 34 33 33 31 20 20 00 00 9a d9 c1 3c 00 M354331 ·····<·
0050 00 00 c1 12 00 00 80 00 00 00 00 0e 00 00 00 00 ···· ····
0060 00 50 00 00 03 fa 00 00 9a d6 00 00 00 79 53 57 ·P····· ·····ySW
0070 20 63 6f 72 6e 65 72 20 6f 66 20 42 69 6c 74 6d corner of Biltm
0080 6f 72 65 20 4d 61 6c 6c 20 20 20 20 20 20 20 20 ore Mall
0090 20 20 20 20 20 20 6e 53 32 32 39 34 32 20 20 20 nS 22942
00a0 20 00 00 9a d9 c1 46 32 00 00 c1 14 00 00 00 00 ·····F2 ····
00b0 9a f1 00 0e 00 00 00 00 00 50 00 00 03 fb 00 00 ···· ·P····
00c0 9a d7 00 00 00 7a 63 6c 6f 73 65 64 20 74 69 6c ·····zcl osed til
00d0 20 6e 6f 6f 6e 20 4d 6f 6e 64 61 79 73 20 20 20 noon Mo ndays
00e0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 6e 5a nZ
00f0 35 35 37 30 39 20 20 20 20 00 00 9a dc c1 5a 00 55709 ·····Z·
0100 00 00 c1 17 00 00 00 00 9a f1 00 0e 00 00 00 00 ···· ····
0110 00 50 00 00 03 fc 00 00 9a d7 00 00 00 7b 65 78 ·P····· ·····{ex
0120 70 72 65 73 73 20 20 20 20 20 20 20 20 20 20 20 press
0130 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
Data (data.data), 592 bytes
Packets: 32 · Displayed: 32 (100.0%) · Dropped: 0 (0.0%)
Profile: Default

Agenda

- What is encryption?
- Why do we want to use it for Informix?
- Considerations
- Configuring the Server with GSKit
- Configuring the Client
- Putting it all together
- Setting up Informix Clients with OpenSSL (14.10xC4)
- Using an external Certificate Authority

What is Encryption?

- Protecting data by encapsulating it in a way that only trusted parties can read it
- Many different forms and implementations
- Uses CPU to encrypt to send and more CPU to decrypt when received
- Can be configured for desired strength

PKI (Public Key Infrastructure)

- Common method of encryption (see HTTPS)
- Two keys are needed, a public key that is okay to share and a private key that only the server should know
- A CA (certificate authority) issues the keys - for the first exercise we will ‘self-sign’ and act as our own CA

How Informix Uses PKI

1. Server sends digital certificate to client
2. Client verifies the digital certificate
3. If validated the client creates a limited use key, encrypts it using the server's public key and sends it to the server
4. Server gets the key, decrypts it, and will use the limited use key as long as the session is active
5. All further data on that connection is now encrypted, and only known to the two sides

Why Is Encryption Important For Informix?

- Avoid someone sniffing passwords
- Verify that the server you connect to is valid
- Avoid having someone stand between you and your server (man in the middle)
- Avoid having your data watched by a third party, by default data is sent in clear text
- Comply with many regulations that require it
- Have data encrypted the entire way to and from the client and server

Why It Is Important

Encryption Considerations

- Requires setup on both the server and clients
- Additional CPU overhead on both sides
- May not support old connections (CSDK prior to version 3.x, ADODB, etc)
- Certificates expire
- Larger key size is more secure but higher CPU usage

What Supports SSL?

- ODBC, JDBC, and SQLJ connections
- DRDA and SQLI
- ESQLC
- dbaccess
- ER connections
- HDR connections
- Informix standard utilities
- Connection Manager
- Distributed queries
- PAM SSO

IBM GSKit

- Primary utility to setup and manage encryption keys for Informix
- Ships with Informix CSDK and Server
- Provides libraries and utilities for SSL and TLS communications
- Used by Informix, DB2, and other IBM products
- Primary utility is gsk8capicmd (or gsk7capicmd if on a legacy system)

Terminology

- Keystore– Small file-based database of certificates (public and/or private keys)
- Stash file – Small local protected file that contains password information to open password protected keystores
- Certificate – File that holds public key information

Putting It All Together

1. Configure Server
2. Add SSL listener to sqlhosts
3. Create Server Keystore
4. Export Server Certificate
5. Create conssl.cfg file in $INFORMIXDIR/etc
6. Create Client Keystore
7. Add Server Cert to Client Keystore
8. Test

Configuring The Server

- Verify gskit is installed
- Adjust onconfig values
- Update sqlhosts
- Create the keystore and stash files
- Bring the listener online
- Keystore files need to be in $INFORMIXDIR/ssl

onconfig Changes

- SSL_KEYSTORE_LABEL ifx_encrypt
- NETTYPE socssl,1,50,NET
- VPCLASS encrypt,num=1
- Add a new DBSERVERALIAS for ssl

  ifx_server_ssl

Update sqlhosts

```
ifx_server_ssl    onsocssl server   port
```

Setting Up Server Keystore

- All commands will be run in $INFORMIXDIR/ssl
- If on a legacy system use gsk7capicmd in place of gsk8capicmd
- If on a 64-bit system, the command will have _64 at the end (gsk8capicmd_64)
- In this example the DBSERVERNAME is ifx_server
- The keystore label here is “ifx_encrypt”
- The DBSERVERNAME must match the keystore and stash filename

Create The Keystore

```
gsk8capicmd_64 -keydb -create -db \
ifx_server.kdb -pw password -type cms -stash
```

Flags:

- **-keydb -create** : Create a new keystore
- **-db** :use the local database ifx_server.kdb
- **-pw** : set the password to the value
- **-type cms** : Certificate type - we use cms for Informix
- **-stash** : Stash the passwords with the files

Create Server Cert

Create server cert, the label needs to match the SSL_KEYSTORE_LABEL value

```
gsk8capicmd_64 -cert -create -db ifx_server.kdb \
-stashed -label ifx_encrypt -size 2048 \
-default_cert yes -expire 365 -dn "CN=ifx_server_ssl"
```

-cert -create : create a new certificate
-db ifx_server.kdb : use that file for the database
-stashed : read the stash file to get the password
-label ifx_encrypt : the label we defined in $ONCONFIG
-size 2048 : the size of the key pair

Create Server Cert (cont.)

```
gsk8capicmd_64 -cert -create -db ifx_server.kdb \
-stashed -label ifx_encrypt -size 2048 \
-default_cert yes -expire 365 -dn "CN=ifx_server_ssl"
```

-default_cert yes : this will be the default certificate

-expire 365 : number of days for the certificate to be valid

-dn "CN=ifx_server_ssl" : Unique name for this certificate, only CN= is required

Configure Server (cont.)

- At this point you will have two files
  - ifx_server.sth (stash)
  - ifx_server.kdb (keystore database)
- Both need to have owner/permissions informix:informix 600

Set Up Local Clients GSKit

Create client keystore

```
gsk8capicmd_64 -keydb -create -db clikeydb.kdb \
 -pw password -type cms –stash
```

Extract the public cert from the server keystore, write it to ifx_server.cert (plain text)

```
gsk8capicmd_64 -cert -extract -db ifx_server.kdb \
-format ascii -label ifx_encrypt -pw password \
-target ifx_server.cert
```

Add Certificate To Keystore GSKit

Add server cert to the keystore:

```
gsk8capicmd_64 -cert -add -db clikeydb.kdb \
-stashed -label ifx_encrypt -file ifx_server.cert \
-format ascii
```

Make sure the clikeydb.* files are 664 for other clients to use

Client conssl.cfg

$INFORMIXDIR/etc/conssl.cfg

```
SSL_KEYSTORE_FILE       /opt/informix/ssl/clikeydb.kdb
SSL_KEYSTORE_STH        /opt/informix/ssl/clikeydb.sth
```

Server Final Steps

- Bring the engine up with the SSL listener enabled
- Cannot use onmode –P for SSL listeners.
- Verify you can connect to the TCP ports via dbaccess
- Verify you can connect to the SSL ports via dbaccess
- Repeat the client keystore creation and cert import for any other UNIX clients

Check The Server

- Look for the startup message that the SSL listener is up:

```
Forking 1 'socssl' listener threads...succeeded
```

onstat –g ntt:

```
45b1c1c0 socssllst        9 15:40:06 15:40:33  informix14_1|9089|socssl
```

How you will sleep after it is in place

Configuring Windows Client

- Create a directory where the keystore and the stash files can live
- This can be $INFORMIXDIR/ssl
- For this example, we will use c:\ssl

Set Up conssl.cfg

- Enter the Informix Client SDK directory/etc
- If using more than one version of CSDK it needs to be in all of the etc directories
- Needs to contain

```
SSL_KEYSTORE_FILE C:\ssl\clikeydb.kdb
SSL_KEYSTORE_STH C:\ssl\clikeydb.sth
```

- If using a directory with spaces it needs to use DOS formatting

```
C:\progra~1\inform~1\etc
```

Generating Keydb (Windows)

- Copy the ifx_server.cert file from the server to c:\ssl
- Run a command window as administrator
  - Open start menu
  - Type cmd
  - Right click on ‘Command Prompt’ choose ‘Run as administrator’

Generating Keydb (cont.)

Add your gsk8\bin directory to your path if it is not already there

set PATH=%PATH%;c:\progra~1\ibm\gsk8\bin

cd c:\ssl

Create a new client keydb as on UNIX

gsk8capicmd_64.exe -keydb -create -db clikeydb.kdb \

-pw password -type cms –stash

gsk8capicmd_64.exe -cert -add -db clikeydb.kdb \

-label ifx_encrypt -file ifx_server.cert –stashed \

-format ascii

Windows Connection

- Set up your ODBC connection as normal
- Make sure to use onsocssl and the DBSERVERALIASES value of the ssl listener
- Test your connection
- You can use a client keystore generated on Linux as long as they are on the same GSKit version

OpenSSL Integration

- Integrated in Client SDK 4.50xc4 and IDS 14.1xc4
- By far the most used SSL system
- Widely supported
- Better certificate management
- Does not require GSKit to be installed
- You have to select it on the CSDK install

Enabling Encryption With OpenSSL

- Create a private key pair
- Create a self signed certificate
- Create a .pem file using both
- Create the keystore to contain both files
- Create the stash file
- Set SSL_KEYSTORE_LABEL
- Bring the engine up

OpenSSL Commands

openssl genrsa –out=server_key.pem

openssl req –new –x509 –key server_key.pem \
-subj "/CN=<servername>" –days 3000 -out server_cert.pem

cat server_key.pem server_cert.pem > server_import.pem

openssl pkcs12 –export –in server_import.pem \
-name server_ssl_label –passout pass:abc12345 –out <servername>.p12

onkstash <servername>.p12 abc123

SSL_KEYSTORE_LABEL server_ssl_label

OpenSSL Config

```
informix@llama:/opt/informix/ssl$ openssl genrsa -out=server_key.pem
Generating RSA private key, 2048 bit long modulus (2 primes)
.....................................................................+++++
.............+++++
e is 65537 (0x010001)
informix@llama:/opt/informix/ssl$ openssl req -new -x509 -key server_key.pem -subj "/CN=llama" -days 3000 -out server_cert.pem
informix@llama:/opt/informix/ssl$ cat server_key.pem server_cert.pem > server_import.pem
informix@llama:/opt/informix/ssl$ openssl pkcs12 -export -in server_import.pem -name server_ssl_label -passout pass:abc12345 -out llama_tcp.p12
informix@llama:/opt/informix/ssl$ onkstash llama_tcp.p12 abc12345
informix@llama:/opt/informix/ssl$ ls
llama_tcp.p12  llama_tcp.sth  server_cert.pem  server_import.pem  server_key.pem
informix@llama:/opt/informix/ssl$ oninit
informix@llama:/opt/informix/ssl$ onstat -g ntt

IBM Informix Dynamic Server Version 14.10.FC7DE -- On-Line -- Up 00:00:26 -- 181252 Kbytes
2021-11-16 08:01:33

global network information:
  #netscb connects        read       write    q-free  q-limits  q-exceed alloc/max  slow_DNS
   5/   5       0          0           0    0/   0  170/  10    0/   0    0/  -1        0

Individual thread network information (times):
         netscb thread name    sid     open     read    write address
       45ec0600 socssllst        7 08:01:12                 10.19.39.97|9098|socssl
       4634fc90 soctcplst        5 08:01:12                 llama|10000|soctcp
       45eb0b88 soctcplst        4 08:01:12                 10.19.39.97|9088|soctcp
       45eacb88 socsslpoll       3 08:01:12
       45ea6ac0 soctcppoll       2 08:01:12
```

```
write address
      10.19.39.97|9098|socssl
      llama|10000|soctcp
      10.19.39.97|9088|soctcp
```

OpenSSL Client

(Server)

openssl pkcs12 –in <INFORMIXSERVER>.p12 \
-passin pass:abc12345 \
–out KEYSTORE_LABEL.cert.pem –nokeys

scp server_name.cert.pem \ target_system:$INFORMIXDIR/ssl

(Client)

openssl pkcs12 –export –out client.p12 \
–passout pass:xyz56789 \
-in server_name.cert.pem –caname server_name –nokeys

onkstash client.p12 xyz56789
set up conssl.cfg
chmod 644 client.*

Using An External CA

1. Configure Server
2. Add SSL listener to sqlhosts
3. Create Server Keystore
4. Create Server CSR (certificate signing request)
5. Send the CSR to the CA (Certificate Authority)
6. Get back a new key and the CA's own Certificate
7. Import both into your keystore
8. Create conssl.cfg file in $INFORMIXDIR/etc
9. Create Client Keystore
10. Add CA certificate to Client Keystore
11. Test

Using An External Certificate Authority

- Settings for this example:

INFORMIXSERVER=	informix4
INFORMIXDIR=	/opt/informix
Primary listener:	informix4 onsoctcp 9088
SSL Listener:	informix4_ssl onsocssl 9089
SSL_KEYSTORE_LABEL:	test_ssl_label
Keystore Password:	my_password

Set Up The Server

Create the server keystore

gsk8capicmd_64 -keydb -create -db informix4.kdb -pw my_password -type cms -stash

Use GSKit to generate a CSR (certificate request file)

gsk8capicmd_64 -certreq -create -db informix4.kdb -stashed -label test_ssl_label -dn "CN=xdbsystems.com" -size 2048 -sigalg SHA256_WITH_RSA -target informix4.csr

Importing The Keys

Add the management chain certificates to the keystore

You will get this from the CA server

gsk8capicmd_64 -cert -add -db informix4.kdb -pw my_password -file ManagementCA-chain.pem

Import the new server certificate you received from the certificate request

gsk8capicmd_64 -cert -receive -db informix4.kdb -stashed -file informix4.pem

Validate The Keys

Check the list of certificates, note the long CN name created by the CA, you will need it

gsk8capicmd_64 -cert -list -db informix4.kdb -stashed

Certificates found

* default, - personal, ! trusted, # secret key

! "O=EJBCA,CN=ManagementCA,UID=c-0zno0f82c4y4p6umb"

- test_ssl_label

The CN

O=EJBCA,CN=ManagementCA,UID=c-0zno0f82c4y4p6umb

Create The Client Keystore

Create the keystore, importing the CA Management Chain. Note that we are importing the CN of the server certificate from the prior step

openssl pkcs12 -export -nokeys -in /opt/informix/ssl/ManagementCA-chain.pem -caname "O=EJBCA,CN=ManagementCA,UID=c-0fno0f82c4y4p6umb" -passout pass:my_password -out client1.p12

Stash the file

onkstash client1.p12 my_password

Note that no certificate file for the server was needed, just the CA chain. The CA will handle all of the certificates

OpenSSL With A CA

Create the keystore based on the Management CA pem file
Need to provide the full caname

```
openssl pkcs12 -export -nokeys
-in /opt/informix/ssl/ManagementCA-chain.pem \
-caname "O=EJBCA,CN=ManagementCA,UID=c-
0fno0f82c4y4p6umb" \
-passout pass:my_password -out client1.p12
```

Stash the file as before

```
onkstash client1.p12 my_password
```

That’s It

- At this point your client keystore only has a single entry in for it:

*default, - personal, ! trusted, # secret key

! "O=EJBCA,CN=ManagementCA,UID=c-0zno0f82c4y4p6umb“

With trusting the CA Chain certificate, all server keys that were validated by the CA will be trusted, so you do not need specific server certificates installed on the client keystore.

New keys will be validated automatically.

Other Notes

- You can have multiple servers' certificates imported into in a client keyring allowing it to SSL access many systems
- You can reuse a client keyring between multiple client systems
- Make sure any users that need to connect can read from the keystore and stash files
- If doing server to server communication you need all server certificates in each server's server keystore

Alternative Options

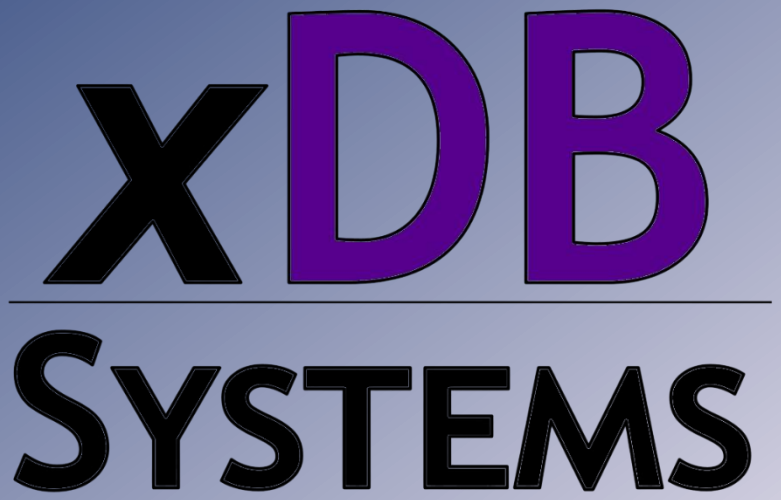
- Permanent or on-demand VPNs
- SSH Tunnels
- SPWDCSM (simple password communication support module)
- Mixed environment
- Hope the auditors don’t notice

Converting from GSKit to OpenSSL

- https://www.iiug.org/en/2020/10/27/replay-for-the-ibm-informix-from-gskit-to-openssl-webinar-is-now-avaiable/

Advanced Informix Consulting and Support

- **Informix Remote DBA 24/7** Peace of mind for your systems
- **Expert consultants** for any Informix problem
- Support for **Informix Upgrades** from and to any version
- **Migrations** to new hardware, let us help virtualize your systems
- Get help **configuring** and **managing** UNIX systems
- Informix **cloud** migrations
- **IBM Informix sales**
- Let us **tune your system**, we can maximize the potential of your database
- *What can we do for you today?*



Thank You

Thomas Beebe

tom@xdbsystems.com

For more information:

https://www.xdbsystems.com