SSL for VSE

Connectivity Systems
Product Development
Don Stoever

Agenda

- The Need
  - Secure communications with applications on VSE
- The Solution
  - The SSL protocol
- The Tools
  - Standard cryptographic algorithms
- The Implementation
  - Installing SSL on VSE
- The Benefits
  - Creating secure applications for VSE

The Need

- E-Business
- E-Commerce
- Secret web sites and ports
- Viruses
- Hackers
- Denial of Service attacks
- Authentication
- Confidentiality
- Data Integrity

The Need: IP problems

- IP packets have no inherent security
  - Relatively easy to forge the addresses
  - Modify the contents
  - Replay old packets
  - Contents easy to inspect
- No guarantee that IP packets are:
  - From the claimed sender
  - Contain the original data set by sender
  - Not inspected by a third party

The Need: TCP problems

- TCP provides a reliable connection
  - Lost packets are retransmitted
  - But no:
    - Authentication
    - Confidentiality
    - Integrity
    - Repudiation

Application Message Integrity

- Messages
  - contain sensitive data
  - travel a complex path
  - must be authenticated
  - must be kept confidential
  - must not be altered
- Why not AMI for VSE ???

Why not just front end VSE ?

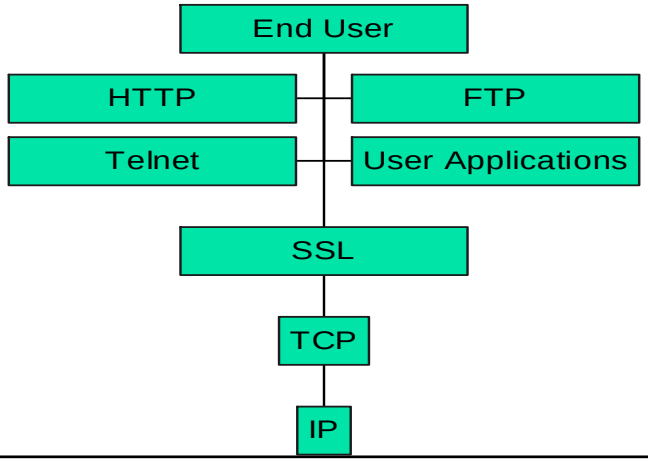
- Native solution is:
  - More secure
  - Efficient
  - Cheaper
  - Easy to maintain
  - Less complicated
- VSE can now do it all too...
- So, why not have secure messaging applications on VSE ?

The Solution: SSL for VSE

- SSL provides secure messaging for TCP/IP applications on VSE by using:
  - Public Key Infrastructure for server and client authentication
  - Data Encryption for confidentiality
  - One-way keyed hash functions for message integrity
  - Digital Signatures for proof of authorship

The Solution: TCP with SSL Protocol

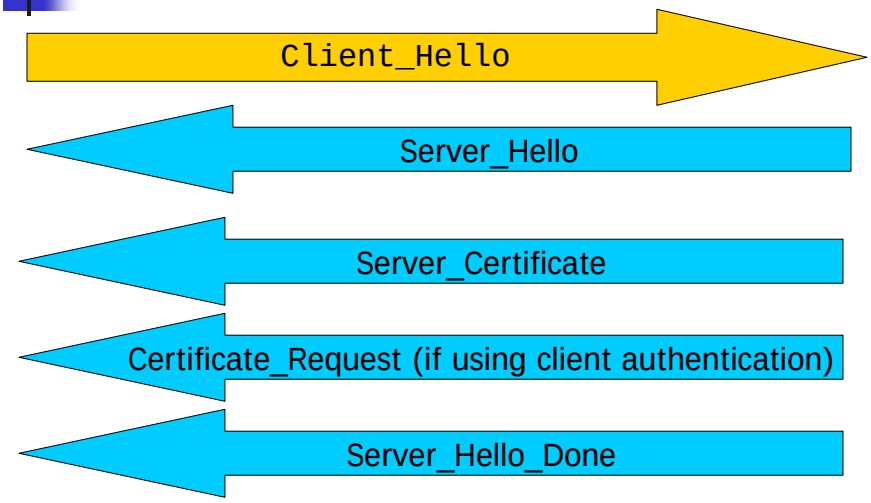
Secure



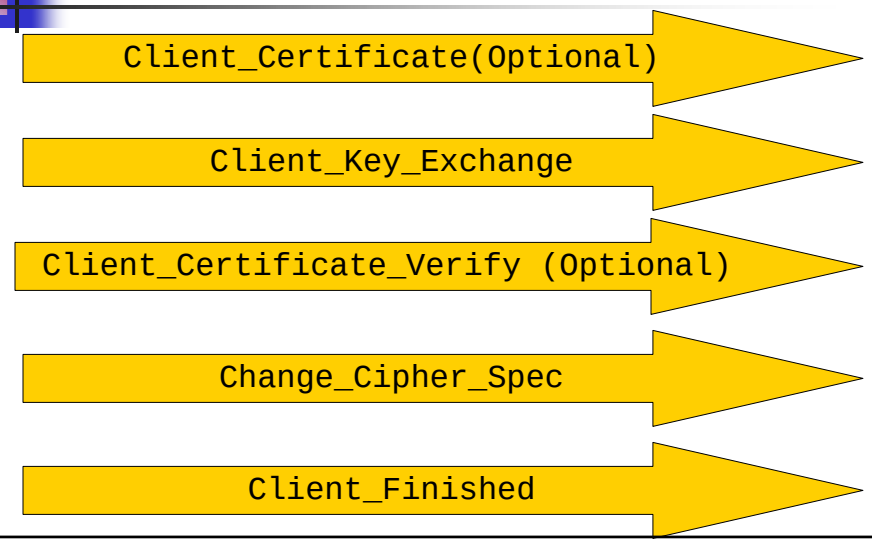
SSL Overview

- Two sockets connected
  - One must be a Server the other a Client
- Server always authenticated
- Client authentication optional
- Client and Server must:
  - Agree on cipher algorithms
  - Establish crypto keys

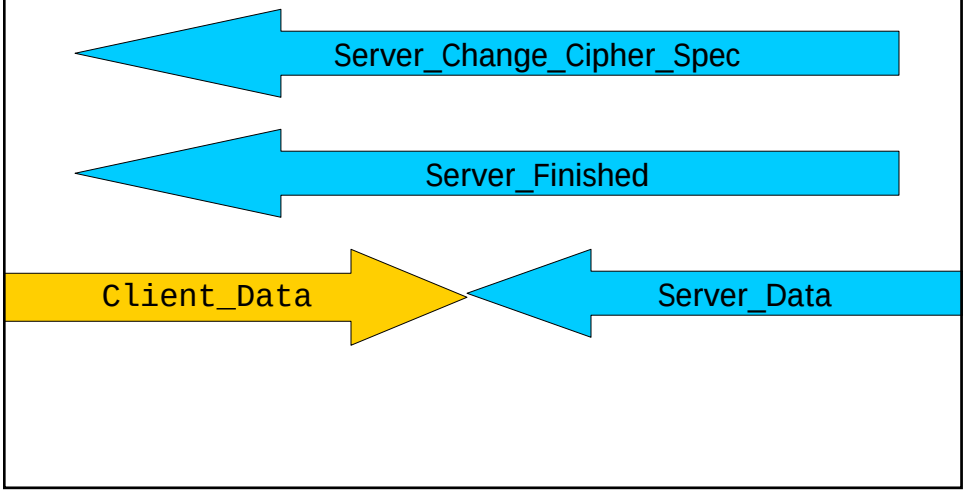
SSL Handshake Hello's



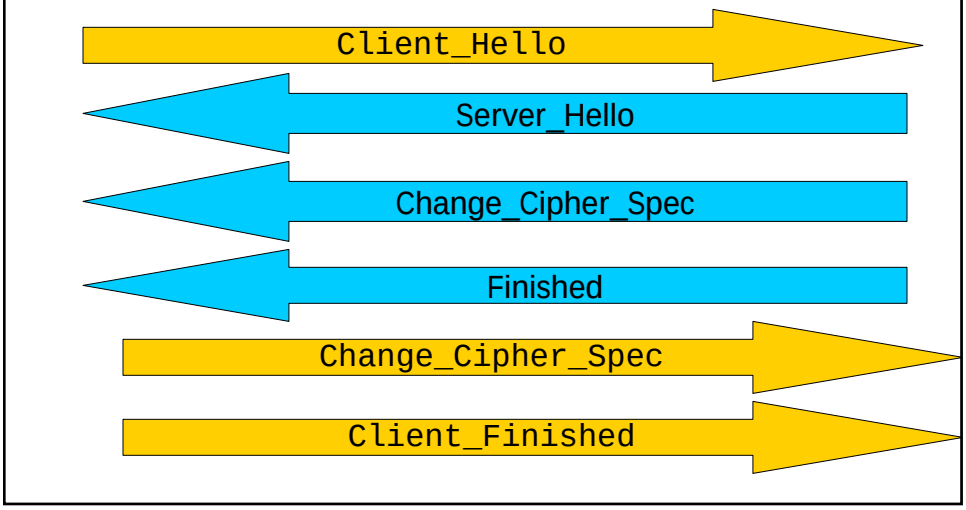
SSL Handshake Client Messages



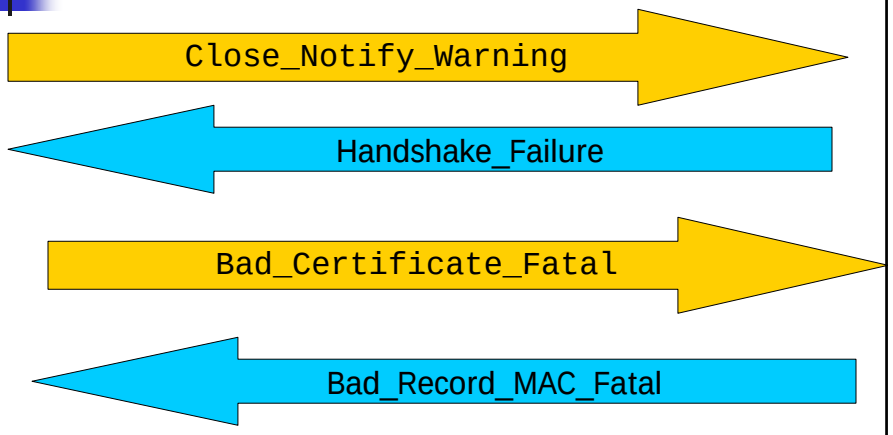
SSL Handshake Server Messages



SSL Handshake Resuming a Previous Session



SSL Alerts



SSL Enabled Server on VSE

- Server allocates a socket binds to a port, listens, and issues a accept.
- Client connects to the VSE server and sends a “client hello”.
- Server passes control to the SSL4VSE secure socket initialization routine which performs the actual SSL handshake.
- Server responds to the “client hello” by choosing the cipher algorithms that will be used during the session and sending the clients its x.509v3 PKI certificate.

SSL Enabled Server on VSE

- Key material is generated that will be used for encryption, decryption, and message authentication.
- Once the handshake is completed a secure connection is ready, and the server and client can then use secure socket read and write functions of the SSL4VSE API.

The Tools: Cryptography Algorithms

- SSL requires cryptography functions
  - X509v3 PKI certificates for identification
  - RSA for key exchange
  - DES for data encryption
  - MD5 and SHA-1 for message hashing
  - HMAC for message authentication

Crypto Toolkit for VSE

- API for cryptography standards
  - Message Digest algorithms
    - MD5 RFC1321
    - SHA-1 FIPS Pub 180-1
  - Bulk Data encryption
    - DES FIPS Pub 46-3
    - Triple DES Ansi x9.52 Triple DES
  - Message authentication
    - HMAC RFC2104
  - Digital Signatures
    - RSA PKCS#1

RSA Public Key encryption

- Used by SSL for initial key exchange and digital signatures
- Separate keys used for encrypt and decrypt
- Public key shared with others in signed certificate
- Private key used to decrypt and for creating digital signatures
- RSA patent expired in September, 2000

Installing SSL on VSE

- Install a SSL enabled client
  - MS-IE, QWS3270 Secure, Zephyr Passport, etc
- Create a RSA private key file
- Submit a CSR request to a Certificate Authority.
  - It will contain your public key and is digitally signed with your private key

Installing SSL on VSE

- Install the CA signed certificate
- Install the CA root certificate
- Configure the SSL daemon on VSE

Example SSL enabled client

- Microsoft Internet Explorer
  - Must be version 5 or higher
  - Under the the "Tools - Internet Options - Advanced - Security“ the "Use SSL/TLS“ checkbox must be checked
  - Under the the "Tools - Internet Options – Content - Certificates the CA root certificate must be installed

Creating a RSA Key file

- Based on RSA PKCS#1

```
// EXEC CIALSRVR
SETPORT 5622
/*
* *  RSA private key created on PC and sent to VSE
/&
```

Creating a CSR Request

- Based on RFC2314

```
// EXEC CIALCREQ
Webmaster: dstoever@tcpip4vse.com
Phone: xxx-xxx-xxxx
Server: TCP/IP for VSE 1.4
Common-name: www.dstoever.com
Organization Unit: Development
Organization: Connectivity Systems
Locality: Columbus
State: Ohio
Country: US
/*
```

Install Certificate Authority Root Certificate

```
// EXEC CIALROOT
-----BEGIN CERTIFICATE-----
MIICpDCCAg2gAwIBAgIDPltCMA0GCSqGSIb3DQEBBAUAMIGHMQswCQYDVQQGEwJa
QTEiMCAGA1UECBMZRk9SIFRFU1RJTkcgUFVSUE9TRVMgT05MWTEdMBsGA1UEChMU
VGhhd3RIIENlcnRpZmljYXRpb24xFzAVBgNVBAsTDlRFU1QgVEVTVCBURVNUMRww
...
IyqW1vNOcNo=
-----END CERTIFICATE-----
/*
```

Install CA Signed certificate

```
// EXEC CIALCERT
-----BEGIN CERTIFICATE-----
MIICpDCCAg2gAwIBAgIDPltCMA0GCSqGSIb3DQEBBAUAMIGHMQswCQYDVQQGEwJa
QTEiMCAGA1UECBMZRk9SIFRFU1RJTkcgUFVSUE9TRVMgT05MWTEdMBsGA1UEChMU
VGhhd3RIIENlcnRpZmljYXRpb24xFzAVBgNVBAsTDlRFU1QgVEVTVCBURVNUMRww
...
IyqW1vNOcNo=
-----END CERTIFICATE-----
/*
```

IETF Standards Implemented

http://www.ietf.org/html.charters/tls-charter.html

- RFC2246 The TLS Protocol
  - Handshake requires server certificate from VSE
  - RSA used for generating key material
  - DES used for application data encryption
  - HMAC-SHA1 used for message authentication

IETF Standards Implemented

- RFC1321 The MD5 Message-Digest Algorithm
- RFC2104 HMAC: Keyed hashing for message authentication
- RFC2202 Test Cases for HMAC-MD5 and HMAC-SHA-1
- RFC1113 Universal Printable Character encoding
- RFC2459 Internet x509v3 PKI certificates
- Internet draft HTTP over TLS

FIPS Standards

- PUB 46-3 Data Encryption Standard (DES)
- PUB 81 DES Modes of Operation
- Cipher Block Chaining mode
- PUB 180-1 Secure Hash Standard (SHA-1)
- http://www-08.nist.gov/cryptval/des.htm
- http://csrc.nist.gov/pki/nist_crypto/welcome.html

SSL Implementation on VSE

- Using the SSL Pass-Through server
  - Defining the SSL pass-through daemon on VSE
- Using the SSL API on VSE
  - Client Server application without SSL
  - Client Server application with SSL
- Using Cryptography APIs on VSE
  - Using DES to encrypt data
  - Using SHA-1 to create a message fingerprint
  - Using RSA to create a digital signature
  - Using BASE64 encoding to transmit binary data

Using the SSL Pass-Through Server

- Allows quick and easy implementation of SSL
- No application modifications
- SSLD on VSE performs handshake with SSL enabled client
  - Encrypts outbound data to SSL client
  - Decrypts inbound data from SSL client
- Currently used by TelnetD

Defining the SSL daemon on VSE

- DEFINE TLSD,
  - ID=TLSD01, Identifier
  - PORT=443, We listen here
  - PASSPORT=23 Pass to real daemon
  - CIPHER=0A096208 Allowed ciphers
  - CERTLIB=KEYLIB Library name
  - CERTSUB=SSLKEYS Sublibrary name
  - CERTMEM=SSL4VSE Member name
  - TYPE=1 Server application
  - MINVERS=0300 Protocol version

Implementing SSL into Applications on VSE

- Two sockets connected
  - One must be a Server the other a Client
- Server always authenticated
- Client authentication optional
- Client and Server must:
  - Agree on cipher algorithms
  - Establish crypto keys

SSL for VSE API

- Based on IBM OS/390 SSL Programming Guide and Reference, manual number SC24-5877
  - Easy porting for OS/390 SSL applications
  - Callable from either C or BAL

Client Server without SSL

- Client
  - Allocate socket
  - Connect to server
  - Read/Write socket
  - Close socket
- Server
  - Allocate socket
  - Bind socket to a port
  - Listen on port
  - Accept client connection
  - Read/Write socket
  - Close socket

Client Server with SSL

- Client
  - SSL setup environment
  - Allocate socket
  - Connect to server
  - SSL socket initialization
  - SSL read/write socket
  - SSL Close socket
  - Close socket
- Server
  - SSL setup environment
  - Allocate socket
  - Bind socket to a port
  - Listen on port
  - Accept client connection
  - SSL socket initialization
  - SSL read/write socket
  - SSL Close socket
  - Close socket

SSL for VSE API

- Based on IBM OS/390 SSL Programming Guide and Reference, manual number SC24-5877
  - Easy porting for OS/390 SSL applications
  - Callable from either C or BAL

Setup SSL environment

- C function = gsk_initialize()
  - C header file = SSLVSE.H
- BAL vcon = IPCRINIT
  - BAL macro = SSLVSE.A
- Standard linkage
  - R13 = save area
  - R14 = return address
  - R15 = entry point
  - R1 = parameter list
- On return R15 = return code
  - Negative = failed, R1=@reason

Setup SSL environment

- Minimum acceptable protocol version
- Identify lib.sublib containing the private key and certificates
- Session timeout value for fast client reconnect
- Specifies the method for verifying client certificates

Initialize SSL socket

- C function = gsk_secure_soc_init()
  - C header file = SSLVSE.H
- BAL vcon = IPCRSINI
  - BAL macro = SSLVSE.A
- Standard linkage
  - R13 = save area
  - R14 = return address
  - R15 = entry point
  - R1 = parameter list
- On return R15 = return code
  - Negative = failed, R1=@reason

Initialize SSL socket

- Type of handshake
  - Server/Client without client authentication
  - Server/Client with client authentication
- List of acceptable cipher suites
  - RSA512-Null-MD5, RSA512-Null-SHA
  - RSA512-DES40-SHA
  - RSA1024-DES-SHA
  - RSA1024-TripleDES-SHA

Initialize SSL socket

- @ of read socket routine
- @ of write socket routine
- Calls back into your code for reading and writing to the actual socket
- Parmlist passed contains:
  - Fullword handle for use by application
  - @ of data receive/send area
  - Length of data receive/send area

Using cryptography APIs on VSE

- SSL requires cryptography functions
  - X509v3 PKI certificates for identification
  - RSA for key exchange
  - DES for data encryption
  - MD5 and SHA-1 for message hashing
  - HMAC for message authentication

CryptoVSE API algorithms

- API for cryptography standards
  - Message Digest algorithms
    - MD5 RFC1321
    - SHA-1 FIPS Pub 180-1
  - Bulk Data encryption
    - DES FIPS Pub 46-3
    - Triple DES Ansi x9.52 Triple DES
  - Message authentication
    - HMAC RFC2104
  - Digital Signatures
    - RSA PKCS#1

Using DES to encrypt data

- C function = cry_des_encrypt()
  - C header file = SSLVSE.H
- BAL vcon = CRYDESEC
  - BAL macro = SSLVSE.A
- Standard linkage
  - R13 = save area
  - R14 = return address
  - R15 = entry point
  - R1 = parameter list
- On return R15 = return code
  - Negative = failed, R1=@reason

Using DES to encrypt data

- Parameters addresses off R1
  - 0(R1) = address of data to encrypt
  - 4(R1) = length of data to encrypt
  - 8(R1) = address of key
  - 12(R1) = length of key
  - 16(R1) = address of work area
  - 20(R1) = length of work area

Using SHA-1 to create a message fingerprint

- C function = cry_sha_hash()
  - C header file = SSLVSE.H
- BAL vcon = CRYSHAHA
  - BAL macro = SSLVSE.A
- Standard linkage
  - R13 = save area
  - R14 = return address
  - R15 = entry point
  - R1 = parameter list
- On return R15 = return code
  - Negative = failed, R1=@reason

Using SHA-1 to create a message fingerprint

- Parameters addresses off R1
  - 0(R1) = address of input data for hash
  - 4(R1) = length of input data
  - 8(R1) = not used
  - 12(R1) = not used
  - 16(R1) = address of work area
  - 20(R1) = length of work area
- 20-byte SHA-1 hash will be returned in the supplied work area

Using RSA to create a digital signature

- C function = cry_rsa_signature_create()
  - C header file = SSLVSE.H
- BAL vcon = CRYRSASI
  - BAL macro = SSLVSE.A
- Standard linkage
  - R13 = save area
  - R14 = return address
  - R15 = entry point
  - R1 = parameter list
- On return R15 = return code
  - Negative = failed, R1=@reason

Using RSA to create a digital signature

- Parameters addresses off R1
  - 0(R1) = address of input data
  - 4(R1) = length of input data
  - 8(R1) = address of RSA private key
  - 12(R1) = length of RSA private key
  - 16(R1) = address of work area
  - 20(R1) = length of work area
- 64 or 128 byte RSA PKCS#1 digital signature will be returned in the supplied work area

Using BASE64 encoding to transmit binary data

- C function = cry_universal_print_encode()
  - C header file = SSLVSE.H
- BAL vcon = CRYUPENC
  - BAL macro = SSLVSE.A
- Standard linkage
  - R13 = save area
  - R14 = return address
  - R15 = entry point
  - R1 = parameter list
- On return R15 = return code
  - Negative = failed, R1=@reason

Using BASE64 encoding to transmit binary data

- Parameters addresses off R1
  - 0(R1) = address of input data
  - 4(R1) = length of input data (48)
  - 8(R1) = not used
  - 12(R1) = not used
  - 16(R1) = address of work area
  - 20(R1) = length of work area
- 64 bytes of universally printable characters will be returned in the supplied work area

Questions ?