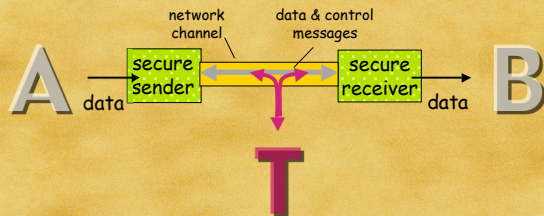
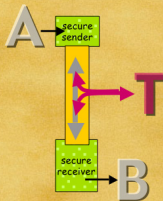


Security Issues in a Nutshell



Based on a diagram by Kurose & Ross

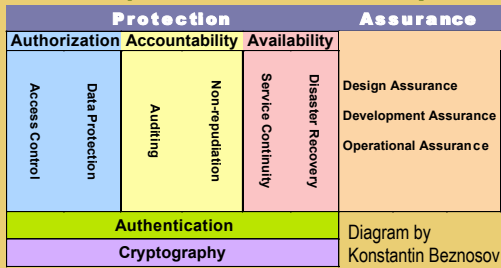
What can 'the intruder' do?*



- ⊗ *eavesdrop*: intercept messages
- ⊗ *insert* messages into connection
- ⊗ *impersonate*: can fake (spoof) source address in packet (or any field in packet)
- ⊗ *hijack*: 'take over' ongoing connection by removing sender or receiver, inserting himself in place
- ⊗ *denial of service*: prevent service from being used by others (e.g., by overloading resources)



Components of Security



Important Concepts (1 of 2)★

Confidentiality: only sender, intended receiver should "understand" message contents

- ☞ sender encrypts message
- ☞ receiver decrypts message

Authentication: sender and receiver want to confirm identity of each other

Trust: sender and receiver must have confidence in system and its maintainers

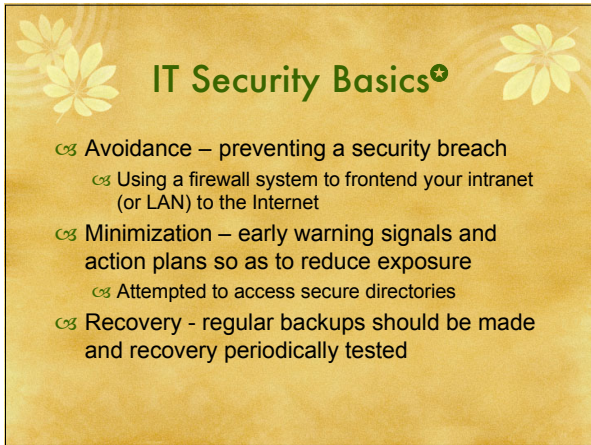
Important Concepts (2 of 2)★

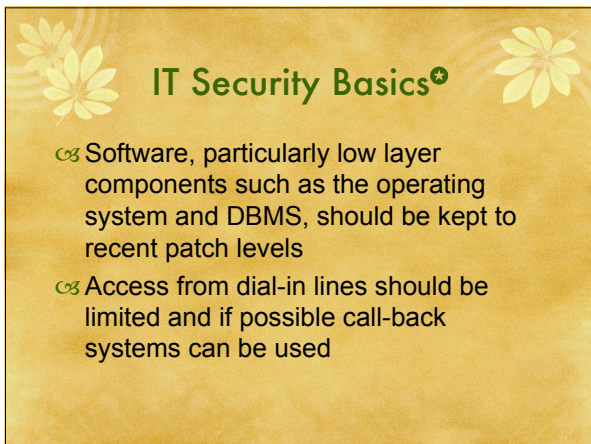
Message Integrity / Data Protection: sender and receiver want to ensure message not altered (in transit, or afterwards) at least **without detection**

Access and Availability: services must be available to users (through the system) and users must be able to use the system

Access Control / Authorization: only specific people (or agents) can use the system







IT Security Basics*

- ☞ Passwords (and potentially User Ids) should be forced to change periodically
- ☞ Passwords should be difficult to guess
 - ☞ Try to create passwords such as:
To Be or Not To Be ➔ 2bon2b
- ☞ Databases should be secured in terms of access rights to data (usually by individual or group)

Physical Security*

- ☞ Large mainframe systems have always had adequate physical security
- ☞ The transition from LAN to WAN to Internet has caused new interest in these methods
- ☞ Physical security means locked doors and security personnel
- ☞ Options are to host on a secure ISP/ASP (InternetHosting.com)

Secure sockets layer (SSL)*

<ul style="list-style-type: none"> ☞ transport layer security to any TCP-based app using SSL services. ☞ used between Web browsers, servers for e-commerce (shttp). ☞ security services: <ul style="list-style-type: none"> ☞ server authentication ☞ data encryption ☞ client authentication (optional) 	<ul style="list-style-type: none"> ☞ server authentication: <ul style="list-style-type: none"> ☞ SSL-enabled browser includes public keys for trusted CAs. ☞ Browser requests server certificate, issued by trusted CA. ☞ Browser uses CA's public key to extract server's public key from certificate. ☞ check your browser's security menu to see its trusted CAs.
---	--

SSL (continued)★

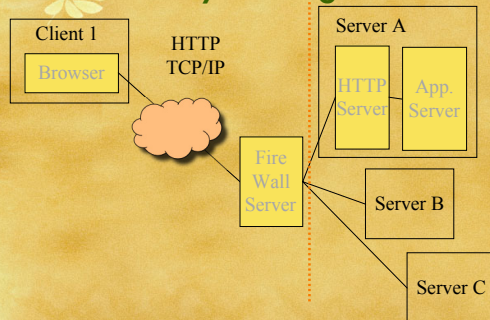
Encrypted SSL session:

- Browser generates *symmetric session key*, encrypts it with server's public key, sends encrypted key to server.
 - Using private key, server decrypts session key.
 - Browser, server know session key
 - All data sent into TCP socket (by client or server) encrypted with session key.
- SSL: basis of IETF Transport Layer Security (TLS).
 - SSL can be used for non-Web applications, e.g., IMAP.
 - Client authentication can be done with client certificates.

Using a Firewall★

- A firewall server or router acts as an electronic security cop
- No machine other than firewall is directly accessible from Internet
- May also function as a "proxy" server allowing intranet systems to access only portions of the Internet
- Internet security methods are focused at the firewall reducing cost and admin overhead

Security through HTTPS★



Cryptography

- ☞ Cryptography or ciphering is an ancient method of encoding a message — only a receiver with a key can decipher the content
- ☞ A single (symmetric) secret key is used to encrypt and decrypt
- ☞ Requires the communication of the key between sender and receiver!
- ☞ Basis of nuclear war-head command and control security

Public Key Cryptography

- ☞ In 1976 Diffie & Hellman at Stanford U. developed *public-key cryptography*
- ☞ Asymmetric:
 - ☞ Private key – kept secret by owner
 - ☞ Public key – distributed freely to all who wish to send
 - ☞ Generated by computer algorithm, so a mathematical relation exists between them ... however ...
 - ☞ It is computationally difficult to determine the private key from the public key, even with knowledge of the encryption algorithm

Public Key Cryptography

- ☞ The keys come in the form of tightly coupled pairs which anyone can generate using methods such as RSA, SHA-1, DSA (RSA is most common)
 - ☞ Javascript demo: <http://shop-js.sourceforge.net/crypto2.htm>
- ☞ There is only one public key corresponding to any one private key and vice versa
- ☞ Sender encodes data using public key of receiver
- ☞ Receiver decodes data using unique private key, no one else can do the same
- ☞ This ensures integrity of the data

Authentication*

- ☞ How can you be sure that the person sending the encrypted data is who they say they are
- ☞ This requires some method of authenticating the identity of the sender
- ☞ The solution is for the sender to "sign" the data using his/her private key – the data is encrypted using the sender's private key
- ☞ The receiver validates (decrypts the data) the "signature" using the sender's public key
- ☞ This will work as long as receiver can be sure the sender's public key belongs to the sender and not an imposter ... enter PKI

Integrity and Authentication*

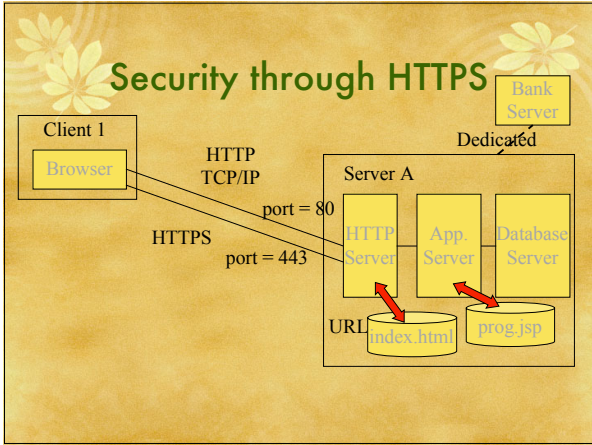
- ☞ Example: Consider a merchant wants to send a secure message to a customer:
 - ☞ Merchant encrypts message using customer's public key
 - ☞ Merchant then signs message by encrypting with their private key
 - ☞ Customer decrypts using the merchants public key to prove authenticity of sender
 - ☞ Customer decrypts using their private key to ensure integrity of message

PKI – Public Key Infrastructure*

- ☞ Integrates PK cryptography with digital certificates and certificate authorities (CA)
- ☞ Digital certificate = issued by a CA, includes user name, public key, serial number, expiration date, signature of trusted CA (message encrypted by CA's private key)
- ☞ Receipt of a valid certificate is proof of identity – can be checked at CAs sight
- ☞ www.verisign.com is major player

Security and HTTPS

- ☞ Certificate is an entity's public key plus other identification (name, CA signature)
- ☞ SSL – Secure Socket Layer
 - ☞ Lies between TCP/IP and HTTP and performs encryption
- ☞ HTTPS is the HTTP protocol that employs SSL – it uses a separate server port (default = 443)



SSL – Secure Socket Layer

1. Client makes HTTPS connection to server
2. Server sends back SSL version and certificate
3. Client checks if certificate from CA
4. Client creates session "premaster secret", encrypts it and sends it to server and creates "master secret"
5. Server uses its private key to decrypt "premaster secret" and create the same "master secret"
6. The master secret is used by both to create session keys for encryption and decryption

Consider Your Rôle

- ☞ Surfer
- ☞ Information provider
- ☞ Shopper
- ☞ Seller
- ☞ Game player
- ☞ ...

Consider Network Rôles

☞ Surfer	☞ Client
☞ Information provider	☞ Server
☞ Shopper	☞ Browser
☞ Seller	☞ Web Server
☞ Game player	☞ User
☞ ...	☞ Interpreting browser display
	☞ Issuing commands

Privacy