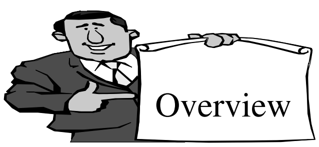
Network Security

Shivkumar Kalyanaraman
Rensselaer Polytechnic Institute
shivkuma@ecse.rpi.edu
http://www.ecse.rpi.edu/Homepages/shivkuma



- Common Network Attacks
- Security techniques: passwords, hash functions, one-time passwords, digital signatures, symmetric/asymmetric key cryptography
- IPSec, SSL, Kerberos, S/Key, (+ mention of PAP, CHAP, RADIUS, TACACS)
- Firewalls

Common Network Attacks

- ***Sniffing/Snooping*** **- Monitoring the network for sensitive data and passwords**
- ***Message Replays*** **- Sending a message repeatedly to a receiver ("replay attack")**
- ***Message Alteration*** **- Modifying a message and sending**
- ***Message Delay and Denial*** **- Lowering or removing quality of service in a network (AKA Denial-of-service)**
- ***Spoofing*** **- Making a packet appear to come from a location other than the one from which it was sent**

Common Network Attacks

- *SYN flooding*:

  1 Z(x) ---SYN---> A ...

  2 X<---SYN/ACK--- A ...

  3 X <---RST--- A

- 1) Attacking host sends a multitude of SYN requests to fill it's backlog queue with pending connections.
- 2) The target responds with SYN/ACKs to what it believes is the source of the incoming SYNs. All further requests to this TCP port will be ignored. The target port is flooded.

Common Network Attacks

- *Avarice* - a SYN,RST generator designed to disallow any TCP traffic on an Ethernet segment.
  - 1) Listen for the 3-way handshake procedure to begin
  - 2) When one is detected, immediately generate a forged RST packet and sends it back to the client
- The result is that no TCP based connections can be negotiated, and therefore no TCP traffic can flow.

Common Network Attacks

- *Sloth* - a zero TCP window generator
- 1) Detect a connection
- 2) Transmits a spoofed TCP zero-size window advertisement,
- 3) Host stops sending data, and start sending window probes
- 3) Constantly return zero-size windows

Common Network Attacks

- *Land Attack* - **sends a spoofed packet with the SYN flag from the same IP and port number as the destination**
- *La Tierra* - **Sends the same packet used in a land attack but to more than one port and it doesn't matter (on some systems, esp. NT) if the port is opened or closed**

Security Requirements

- **Authentication - establishing proof of identity**
- **Access Control - regulating access to some object (also called "authorization")**
- **Integrity - detecting that the data is not tampered with.**
- **Confidentiality - maintaining the privacy of sensitive data**
- **Non-repudiation -** ability to prove that the sender actually sent the data

Authentication techniques

- **Weak: clear-text password**
- **Strong: don't send secrets on the wire**
  - **One-time password (Eg: S/Key. RFC 2289)**
    - User remembers secret pass-phrase.
    - Server issues a challenge (random #)
    - User applies hash function to it multiple times to generate a new password.
  - Simple challenge-response: (Eg: CHAP):
    - Server encrypts a random number based upon the user's password ("challenge")
    - User decrypts & returns result ("response")

Authentication & authorization

- CHAP: also allows server-controlled "re-authentication"
- Kerberos: single sign-on to multiple servers
- Alt: digital signatures (discussed later): authenticates every message.
- Authorization:
  - Which resources can this user access ?
  - Achieved using "access control lists" (ACLs) stored in database or directory
- Client-server rather than peer-peer for better manageability (eg: RADIUS vs CHAP/PAP)

Encryption techniques

- Symmetric encryption: (Eg: DES, RC-4)
  - Share a secret ("key")
  - Encrypt text based upon the shared secret
  - Longer key (eg: 128-bits) => more secure
  - **Advantages:**
    - Less CPU intensive
    - Provides integrity verification and privacy
  - **Disadvantages**:
    - Keys have to somehow reach receivers
    - Need one key for every receiver
    - Need separate authentication infrastructure

Public Key Encryption (PKE)

- Asymmetric ("Public-key Encryptoin")
  - Eg: RSA, Diffie-Hellman
  - Public key; Private key
  - Data -> Public key -> private key -> Data
  - Use ***receiver's public*** key to encrypt and send data to receiver ("body")
  - Authentication => verify ownership of private key. Encrypt message with ***sender's private*** key ("signature")
  - Problems:
    - Extremely CPU intensive, and slow
    - Need to secure private keys

Hash Functions, Message Digests, Digital Signatures

- Problem: The private-key based "signature" is too slow to generate & is a lot of overhead
- Solution:
  - 1. Convert the message into a smaller-sized, tough-to-guess numeric value using a "one-way hash function" (eg: MD5, SHA)
  - 2. This numeric value (16-32 bytes) is called a "message digest" or a Message Authentication Code (MAC)
  - 3. Encrypt the MAC with the private key to create a "digital signature"
  - 4. Receiver generates MAC, decrypts digital signature and compares to authenticate

PKE (contd)

- PKE slow => the text of the message is encrypted using symmetric encryption (eg: DES): integrity and confidentiality
  - Append digital signature for authentication & non-repudiation
- Another problem with PKE:
  - Anyone can create a new public key and advertise it as belonging to a third-party.
  - Need to authenticate advertiser of public key, and later verify that the sender indeed has the corresponding private key

X.509 and Certificate Authorities (CAs)

- Solution:
  - Have a trusted third-party ("certificate authority" (CA)) authenticate the advertisement of a public key. Eg: Verisign
  - The CA digitally signs the public key advertisement: creates a "X.509 certificate"
  - Issues: ("Public Key Infrastructure (PKI)")
    - CA should guard its private key closely
    - CA does background checks on customers.
    - CA can provide several "grades" of certificates.
    - Certificate registration (CA's public key) security
    - Scalability: need multiple, distributed CAs !

Putting it all together ...

- Server:
  - Securely register with CA
  - Distribute X.509 certificates I.e. public key
  - To send to receiver, use public key of receiver
  - For body: use symmetric encryption using shared secret (aka "cookie") which itself is exchanged using PKE
  - Append signature: Apply hash function to text to generate a MAC, and apply my private key

Putting it all together (contd) ...

- Client:
  - Verify X.509 certificates (public key) for CA signature and certification; store if ok
  - Use private key to decrypt remote's password, and use this to decode the text portion. This may involve matching a result with a crypto-checksum
    - If ok, then integrity, confidentiality guaranteed
  - Use standard hash function on text to get a MAC
  - Apply sender's public key to digital signature to get a MAC value.
  - Compare the two MACs. If equal, then authenticated, non-repudiable.

SSL

- Session oriented, stateful. Integrated w/ HTTPS
- Client may optionally have a X.509 certificate.
- Server required to have an X.509 certificate
- Client verifies server certificate; server performs optional client authentication
- Server private key verified w/ a "challenge".
- Agree to a shared secret for symmetric encryption
  - Session ID is agreed upon -- stored in server cache => not necessary to re-authenticate.
  - Data transfer using 128 bit keys

SSL (contd)

- IETF standard = TLS: transport layer security
- LDAP combined with X.509 certificates, presented through SSL can achieve single “sign-on” access like Kerberos
- Problem: firewalls cant peek in (no “escrow”)
  - Need proxy server terminates SSL sessions at the firewall and no SSL within enterprise.
    - => client authentication cannot be done (proxy server can’t have client’s private key)

Kerberos

- Single sign-on authentication/authorization for enterprise
- Kerberos V5 in Microsoft Win 2000
- Avoids hassles of CAs, and PKI, securing private keys, and private key portability
- Concepts:
  - Realms: Each realm has a master Key Distribution Center (KDC): trusted third party
  - 3 components:
    - Authentication server (AS): responsible for authenticating user

Kerberos (contd)

  - Ticket granting server (TGS): gives access to specific servers to authenticated users
  - Secret key database
- Strong authentication:
  - User sends login name; AS sends TGT (w/ secret key based upon user’s password)
  - User enters password; and workstation attempts to decrypt TGT using this password. After decryption, user gets also a session key
  - Send an “authenticator” to TGS. Encrypted w/ session key (a shared secret w/ TGS), plus name of server, TGT, and timestamps.

Kerberos (contd)

- TGS decrypts authenticator and gives a "service ticket"
  - Gets new session key to be shared between user and server
- Need to access more servers => connect w/ TGS to get service ticket until TGT does not expire

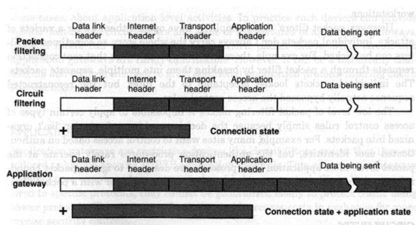
IPSec

- **IPSEC: IP-level Security Protocol**
  - **Encryption takes place between the transport and internet layers**
  - **Designed to provide privacy, forgery detection, or both for IP packets**
  - **Uses a security parameter index (SPI) to negotiate cryptographic and authentication algorithms**
  - **Authentication header (AH) and encapsulating security payload (ESP)**
- **RFC 1825, 1826, 1827 and work in IPSec working group Internet drafts**

Methods: Firewalls

- **Security objectives:**
  - **Ensure controlled access to Internet services**
  - **Protect a distributed enterprise network from outsiders**
  - **Protect the whole range of Internet protocols currently in use**
  - **Public Internet access is available and cost is very important**

Methods: Firewalls

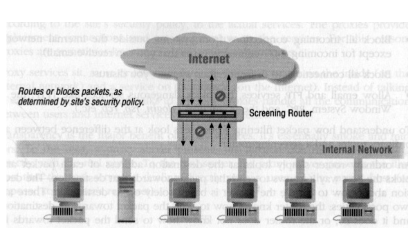


Methods: Firewalls

- Firewall control mechanisms:
  - Packet filtering - Based on the contents of individual packets
  - Circuit filtering - Controls data by controlling the flow of data and blocking if not permitted
  - Application gateway - Processes and forwards messages specific to particular TCP/IP application protocols (AKA proxy)

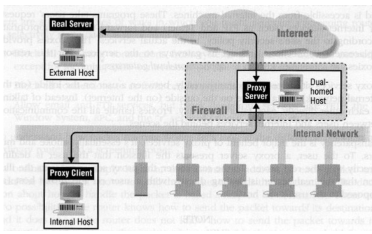
Methods: Firewalls

- Simple firewall router:



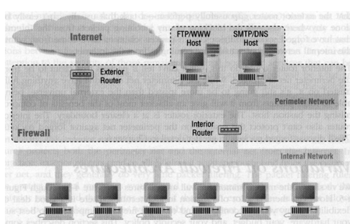
Methods: Firewalls

- Proxy:



Methods: Firewalls

- Perimeter network with bastion hosts



Summary

- **Common Network Attacks**
- **Security techniques: passwords, hash functions, one-time passwords, digital signatures, symmetric/asymmetric key cryptography**
- **IPSec, SSL, Kerberos, S/Key, (+ mention of PAP, CHAP, RADIUS, TACACS)**
- **Firewalls**