Basic network security threats

- **Packet sniffing**
- **Packet forgery (spoofed from address)**
- **DNS spoofing – wrong IP address for hostname**
- **Assume "bad guy" controls network**
  - Can read all your packets
  - Can tamper with your packets
  - Can inject arbitrary new packets

Old authentication systems

- **Send password**
  - Ethernet sniffer collects everyone's password
- **Use IP address (`.rhosts`, NFS)**
  - Assume traffic from "privileged port" is root on host
  - Attacker can still forge packets
- **Use host name**
  - Worse than IP address (DNS insecurity)
- **One-time passwords**
  - Attacker can hijack TCP connection
  - If OTP derived from password, attacker can guess off-line

Keeping communications secret

- **Encryption guarantees secrecy**
  - Block ciphers (like AES)
  - Stream ciphers – block stream XORed with plaintext
  - Attacker cannot recover plaintext from ciphertext w/o K
- **Problem: Attacker can tamper with messages**
  - Stream ciphers – flip any bit
  - Block ciphers in CBC mode – corrupt a block, flip bit in next

Message authentication codes

- **Message authentication codes (MACs)**
  - Sender & receiver share secret key K
  - On message m, $\mathrm{MAC}(K, m) \to v$
  - Attacker cannot produce valid $\langle m, v \rangle$ without K
- **To send message securely, append MAC**
  - Send $\{m, \mathrm{MAC}(K, m)\}$, or encrypt $\{m, \mathrm{MAC}(K, m)\}_{K'}$
  - Receiver of $\{m, v\}$ checks $v \stackrel{?}{=} \mathrm{MAC}(K, m)\}$
- **Problem: Replay – don't believe previous** $\{m, v\}$

The Kerberos authentication system

- **Goal: Authentication in "open environment"**
  - Not all hardware under centralized control (e.g., users have "root" on their workstations)
  - Users require services from many different computers (mail, printing, file service, etc.)
- **Model: Central authority manages all resources**
  - Effectivaly manages human-readable names
  - User names: dm, waldman, . . .
  - Machine names: class1, class2, . . .
  - Must be assigned a name to use the system

Kerberos principals

- ***Principal*: Any entity that can make a statement**
  - Users and servers sending messages on network
  - "Services" that might run on multiple servers
- **Every kerberos principal has a key (password)**
- **Central key distribution server (KDC) knows all keys**
  - Coordinates authentication between other principals

Kerberos protocol

- **Goal: Mutually authenticated communication**
  - Two principals wish to communicate
  - Principals know each other by KDC-assigned name
  - Kerberos establishes shared secret between the two
  - Can use shared secret to encrypt or MAC communication (but most services don't encrypt, none MAC)
- **Approach: Leverage keys shared with KDC**
  - KDC has keys to communicate with any principal

Protocol detail

- **To talk to server s, client c needs key & ticket:**
  - Session key: $K_{s,c}$ (randomly generated key KDC)
  - Ticket: $T = \{s, c, \mathrm{addr}, \mathrm{expire}, K_{s,c}\}_{K_s}$
    (K_S is key s shares with KDC)
  - Only server can decrypt T
- **Given ticket, client creates authenticator:**
  - Authenticator: $T, \{c, \mathrm{addr}, \mathrm{time}\}_{K_{s,c}}$
  - Client must know $K_{s,c}$ to create authenticator
  - T convinces server that $K_{s,c}$ was given to c
- **"Kerberized" protocols begin with authenticator**
  - Replaces passwords, etc.

Getting tickets in Kerberos

- **Upon login, user fetches "ticket-granting ticket"**
  - $c \rightarrow t$: c, t (t is name of TG service)
  - $t \rightarrow c$: $\{K_{c,t}, T_{c,t} = \{s, t, \mathrm{addr}, \mathrm{expire}, K_{s,c}\}_{K_t}\}_{K_c}$
  - Client decrypts with password ($K_c = \mathrm{SHA\text{-}1(pwd)}$)
- **To fetch ticket for server s**
  - $c \rightarrow t$: $s, T_{c,t}, \{c, \mathrm{addr}, \mathrm{time}\}_{K_{c,t}}$
  - $t \rightarrow c$: $\{T_{s,c}, K_{s,c}\}_{K_{c,t}}$
- **To achieve mutual authentication with server:**
  - $c \rightarrow s$: $T_{s,c}, \{c, \mathrm{addr}, \mathrm{time}\}_{K_{s,c}}$
  - $s \rightarrow c$: $\{\mathrm{time} + 1\}_{K_{s,c}}$

Authentication in AFS

- **User logs in, fetches kerberos ticket for AFS server**
- **Hands ticket and session key to file system**
- **Requests/replies accompanied by an authenticator**
  - Authenticator includes CRC checksum of packets
  - Note: CRC is not a valid MAC!
- **What about anonymous access to AFS servers?**
  - User w/o account may want universe-readable files

AFS permissions

- **Each directory has ACL for all its files**
  - Precludes cross-directory links
- **ACL lists principals and permissions**
  - Both "positive" and "negative" access lists
- **Principals: Just kerberos names**
  - Extra principles, system:anyuser, system:authuser
- **Permissions: rwlidak**
  - read, write, lookup, insert, delete, administer, lock

Kerberos inconvenience

- **Large (e.g., university-wide) administrative realms**
  - University-wide administrators often on the critical path
  - Departments can’t add users or set up new servers
  - Can’t develop new services without central admins
  - Can’t upgrade software/protocols without central admins
  - Central admins have monopoly servers/services
    (Can’t set up your own without a principal)
- **Crossing administrative realms a pain**
- **Ticket expirations**
  - Must renew tickets every 12–23 hours
  - Hard to have long-running backround jobs

Security issues with kerberos

- Spoofing local login
- KDC vulnerability
- Kinit could act as oracle
- Replay attacks
- Off-line password guessing
- Can't securely change compromised password