

## Security

Terminology  
Traditional Unix Security  
TCP Wrapper  
Cryptography  
Kerberos

---

---

---

---

---

---

---

---

## Terminology

- ◆ Authentication: identifying someone (or something) reliably. *Proving you are who you say you are.*
- ◆ Authorization: permission to access a resource.

---

---

---

---

---

---

---

---

## Terminology

- ◆ Encryption: Scramble data so that only someone with a secret can make sense of the data.
- ◆ Decryption: Descrambling encrypted data.
- ◆ DES: Data Encryption Standard: secret key cryptographic function standardized by NBS (NIST).

---

---

---

---

---

---

---

---

## Terminology (cont.)

- ◆ Secret Key Cryptography: a cryptographic scheme where the same key is used to encrypt and decrypt.
- ◆ Public Key Cryptography: a cryptographic scheme where different keys are used for encryption and decryption.

---

---

---

---

---

---

---

---

## Terminology (more!)

- ◆ Firewall: a network component that separates two networks and (typically) operates in the upper layers of the OSI reference model (Application layer).
- ◆ Screening Router: a discriminating router that filters packets based on network layer (and sometimes transport layer) protocols and addresses.

---

---

---

---

---

---

---

---

## Unix Network Security

Some basic approaches:

1. Do nothing and assume requesting system is secure.
2. Require host to identify itself and trust users on known hosts.
3. Require a password (*authentication*) every time a service is requested.

---

---

---

---

---

---

---

---

## Traditional Unix Security (BSD)

- ◆ Based on option 2 – trust users on trusted hosts.
  - if the user has been authenticated by a trusted host, we will trust the user.
- ◆ Authentication of hosts based on IP address! (doesn't deal with IP spoofing)

---

---

---

---

---

---

---

---

## Reserved Ports

- ◆ Trust only clients coming from trusted hosts with source port less than 1024.
  - Only *root* can bind to these ports.
- ◆ We trust the host. The request is coming via a trusted service (a reserved port) on the host.

---

---

---

---

---

---

---

---

## Potential Problem

- ◆ Anyone who knows the root password can replace trusted services.
- ◆ Not all Operating Systems have a notion of *root* or reserved ports!
- ◆ It's easy to impersonate a host that is down.

---

---

---

---

---

---

---

---

## Services that use the BSD security model

- ◆ lpd – line printing daemon.
- ◆ rshd – remote execution.
- ◆ rexec – another remote execution.
- ◆ rlogin – remote login.

---

---

---

---

---

---

---

---

## BSD Config Files

- ◆ /etc/hosts.equiv – list of trusted hosts.
- ◆ /etc/hosts.lpd – trusted printing clients.
- ◆ ~/.rusers – user defined trusted hosts and users.

---

---

---

---

---

---

---

---

## lpd security

check client's address for reserved port  
and

check /etc/hosts.equiv for client IP  
OR  
check /etc/hosts.lpd for client IP

---

---

---

---

---

---

---

---

## rshd, rexecd, rlogind security

- ◆ As part of a request for service a username is sent by the client.
- ◆ The username must be valid on the server!

---

---

---

---

---

---

---

---

## rshd security

1. check client's address for reserved port  
if not a reserved port – reject request.
2. check for password entry on server for specified user.  
if not a valid username – reject request.

---

---

---

---

---

---

---

---

## rshd security (cont.)

3. check /etc/hosts.equiv for client's IP address.  
if found – process request.
4. check users ~/.rhosts for client's IP address.  
if found – process request, otherwise reject.

---

---

---

---

---

---

---

---

## rexecd security

client sends username and password to server as part of the request (plaintext).

1. check for password entry on server for user name.
2. encrypt password and check for match.

rexecd is rarely used!

---

---

---

---

---

---

---

---

## rlogind security

- ◆ Just like rshd.
- ◆ If trusted host (user) not found – prompts for a password.

---

---

---

---

---

---

---

---

## Special Cases

- ◆ If username is *root* requests are treated as a special case:
  - look at */.rhosts*
  - often disabled completely.

---

---

---

---

---

---

---

---

## TCP Wrapper

- ◆ TCP wrapper is a simple system that provides some firewall-like functionality.
- ◆ A single host (really just a few services) is isolated from the rest of the world.
- ◆ Functionality includes logging of requests for service and access control.

---

---

---

---

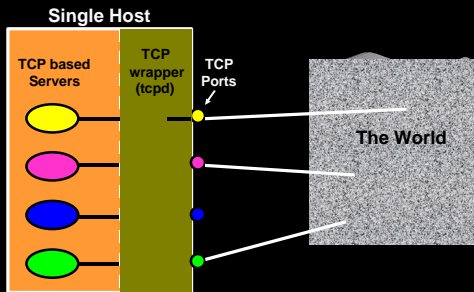
---

---

---

---

## TCP Wrapper Picture



---

---

---

---

---

---

---

---

## tcpd

- ◆ The `tcpd` daemon checks out incoming TCP connections before the real server gets the connection.
- ◆ `tcpd` can find out source IP address and port number (*authentication*).

---

---

---

---

---

---

---

---

## tcpd (cont.)

- ◆ A log message can be generated indicating the service name, client address and time of connection.
- ◆ `tcpd` can use client addresses to *authorize* each service request.

---

---

---

---


---

---

---

---

## Typical tcpd setup

- ◆ `inetd` (the ) is told to start `tcpd` instead of the real server.
- ◆ `tcpd` checks out the client by calling `getpeername` on descriptor 0.
- ◆ `tcpd` decides whether or not to start the real server (by calling `exec`).

---

---

---

---

---

---

---

---

## tcpd configuration

- ◆ The configuration files for `tcpd` specify which hosts are allowed/denied which services.
- ◆ Entire *domains* or IP networks can be permitted or denied easily.
- ◆ `tcpd` can be told to perform RFC931 lookup to get a username.

---

---

---

---

---

---

---

---