SE 4C03 Winter 2001

08. Common Network Services

Instructor: W. M. Farmer

Revised: 6 March 2001

Client-Server Model

- **Servers** provide services over a network
  - Usually listen at a particular **reserved** TCP and/or UDP port
  - May participate in more than one TCP connection at the same time
  - Servers are often called **daemons** and are then given a name that ends with `d` (e.g., `httpd`)
  - A server may be organized as a group of processes or threads
- **Clients** utilize services provided by servers
  - Clients initiate connections to servers
  - Usually are assigned an **ephemeral** port by the OS
- Servers are usually more complex than clients
  - The burden of security falls mostly on the server

Components of a Client-Server Application

1. Communication protocol
2. Server running a server program
3. One or more clients running client programs
4. Communication channels via TCP or UDP

Simple Mail Transport Protocol (SMTP)

- SMTP enables mail to be transferred between hosts
  - SMTP servers listen at TCP port 25
  - The source of mail delivered using SMTP may be spoofed
- Internet mail is usually an essential service
  - Mail should be kept private
  - An organization's mail should be handled by a central mail server for better security and easier administration
  - A database of mail names should be protected
- **Sendmail** is a common Unix implementation of SMTP
  - Large C program that usually runs as root
  - Many exploitable security bugs in past versions

Telnet

- Allows one to remotely log into a host
  - Telnet servers listen at TCP port 23
  - Telnet clients can connect to other TCP ports
- Telnet is often open to attack and exploitation
  - The telnet server is not given special protection
  - Passwords and the telnet session itself are usually transmitted as clear text
  - A telnet session my be hijacked by corrupting the telnet command or sniffing a telnet session
- A telnet session can be made very secure
  - The telnet server is strongly protected from tampering
  - Authentication is performed using one-time passwords
  - The whole telnet session is encrypted (see Secure Shell)

File Transfer Protocol (FTP)

- FTP is the principal protocol for transferring files over the Internet
  - Can handle large files of all types
- Two kinds of FTP:
  - Authenticated
  - Unauthenticated (i.e., anonymous)
- Involves multiple TCP connections
  - One **control connection**
  - Several **data connections**, one for each file transferred
- FTP servers listen at TCP port 21

FTP Operation Modes

FTP can operate in two different modes:

1. **Normal mode**
   - Data connections initiated by FTP server (from TCP port 20)
   - Opens FTP client to attack
2. **Passive mode**
   - Data connections initiated by FTP client (from and to TCP ports ≥ 1024)
   - Opens FTP server to attack

Anonymous FTP

- The user can log in as `anonymous`, `guest`, or `ftp` and does not have to provide a password
- An FTP server offering anonymous FTP must be carefully configured so not to open up the host to attack
  - Read privileges should be limited to the public contents of the FTP repository
  - Write privileges should not be granted if possible
- An anonymous FTP server runs under the `ftp` account and should perform a `chroot` command to set the root directory of the process to a safe, restricted part of the file system (e.g., the home directory of the `ftp` user)
- Access control to the FTP server on Unix is done via the `/etc/ftpusers` file which contains the names of the users who are **denied** access
  - `ftp` can be put here to turn off anonymous FTP

Trivial File Transfer Protocol (TFTP)

- TFTP is a simple protocol for transferring files
- An implementation of TFTP can be much smaller than an implementation of FTP
- There is **no** user authentication!
  - A TFTP server on an Internet host opens ups the host to attack
- Implemented on top of UDP
  - TFTP servers listen at UDP port 69
  - The TFTP protocol ensures some reliability (server acknowledges data packets received)

Domain Name System (DNS)

- Purposes:
  - Provides a hierarchical scheme for naming hosts and collections of hosts
  - Maps host names to IP addresses
  - Maps IP addresses to host names
  - Host names may be assigned aliases
  - Stores information about hosts and collections of hosts
- DNS is managed by a collection of cooperating **name servers**
  - Each server is responsible for part of the name space
  - Name servers communicate with each other using both TCP and UDP
  - Name servers listen on TCP and UDP ports 53
  - Answers to name lookups are cached by name servers to optimize lookup costs

Domain Names

- A **domain name** consists of a sequence of **labels** separated by dots
  - Each suffix of a domain name is also a domain name
  - A domain name denotes a set of one or more hosts
  - A domain name denoting an individual host (called a **host name**) looks no different than a domain name denoting a collection of hosts
  - The domain denoted by $d_1.d_2 \ldots d_n$ is a subdomain of $d_m \ldots d_n$ where $1 \leq m$
  - The syntax of a domain name has nothing to do with the IP addresses or network structure of the hosts in its denotation

Domain Names (cont.)

- DNS may be used with any set of domain names but the Internet's DNS currently uses a set of official top-level domain names of two kinds:
  - Organizational domains (`edu`, `com`, `gov`, `int`, `net`, `mil`, `org`)
  - Country domains (e.g., `ca`, `de`, `uk`, and `us`)
- Additional top-level domain names are in the process of being adopted

DNS Security Concerns

- Authentication based on the domain name of the source host alone is much weaker than authentication based on the IP address alone
  - Host names are easily spoofed
- Attackers may try to corrupt or replace part of the DNS system
  - Packets may be misdirected
  - Name authentication may be thwarted
- DNS is a very effective tool for probing an organization's network
  - Consequently, DNS information concerning the internal network of an organization should be hidden from the Internet

Rlogin and Rsh

- Allows one to remotely log into another host and remotely execute a command, respectively, without providing a password
  - Useful for implementing universal login privileges
  - Servers listen at TCP ports 513 and 514, respectively
- Uses two different password-less authentication mechanisms:
  - Access is allowed to user U on the destination host from user U on the source host if the source host name is in the destination host's `/etc/hosts.equiv` file of **trusted partners** or in U's `.rhosts` file on the destination host
  - Access is allowed to a user U on the destination host if the source host/user name pair is in U's `.rhosts` file

Rlogin and Rsh Security Concerns

- Access rights can be granted by any user
- Trust is transitive
  - Access to one host gives access to all of its trusted partners
  - Attackers will try to deposit an appropriate entry into `/etc/hosts.equiv` or some user's `.rhosts` file
- Host names are authenticated via IP addresses
  - IP spoofing and DNS attacks are possible
- Rlogin and rsh should usually not be made available to hosts on the Internet

X Windows

- X Windows is the major windowing system used on Unix hosts
  - Allows applications to transparently reside on other hosts
  - The X server runs on the user's host and receives information from X clients running on other hosts as well as the user's host
  - X servers listen at TCP ports 6000, 6001, 6002, ...
- X clients can do almost anything: capture screen contents, record key strokes, generate key strokes, etc.
  - X provides a powerful way of attacking a host

X Windows Security Mechanisms

- **xhost mechanism**: The source IP address of a remote X client is compared to the host names authorized using the `xhost` command
  - Authorization is offered to all or no users on a host
  - IP spoofing and DNS attacks are possible
- **Xauthority magic cookie mechanism**: An X client must send the "magic cookie" for the user's current X session to the X server
  - The magic cookie is stored in the user's `.Xauthority` file
  - The magic cookie must be exported to remote hosts

Secure Shell (SSH)

- SSH provides a secure remote shell
  - Secure communication
  - Strong authentication
  - TCP forwarding
  - Secure X communications
- Protects against:
  - Source address, route, and DNS spoofing
  - Password interception
  - Session hijacking
  - Disclosure and modification of transmitted data
- SSH servers listen at TCP port 22
- Intended as a complete replacement of telnet, ftp, rlogin, rsh, rcp, and rdist

Establishing an SSH Connection

1. Client and server establish a TCP connection
2. Client and server exchange protocol identification
3. Server sends host and server public keys to client
4. Client generates session key, encrypts it with both public keys, and sends it back to server with selected cipher type
5. Server sends an encrypted confirmation to client
6. Client authenticates server
   (a) Client checks to see if server's host key is in the user's known hosts file
   (b) If no host key for the server is present, the user is given the opportunity to add it to the known hosts file

Establishing an SSH Connection (cont.)

   (c) If the server's host key has been changed, the user is warned that the server may have been compromised

7. Client authenticates itself to server using public key (or another weaker) authentication method
   (a) Server sends a challenge to client encrypted with the user's public key stored on the server
   (b) Client decrypts the challenge with the user's private key, which is decrypted using the passphrase supplied by the user
   (c) Client sends the required response signed using the user's private key to the server
   (d) Server verifies the response using the user's public key
8. Client makes several requests to finish setting up the secure channel

RPC-Based Services

- Remote Procedure Call (RPC)
  - Protocol is used by a variety of services
  - There are more possible services than port numbers
  - Works with either TCP or UDP
- Authentication can be a problem
  - There are various forms of **secure RPC** that use cryptographic authentication
- A **portmapper** keeps track of what ports the individual RPC servers are listening at
  - Most RPC servers do not listen at fixed ports
  - Clients query the portmapper for the port number of the RPC server they want to contact
  - The portmapper listens at TCP and UDP ports 111
  - The portmapper can be asked to forward a packet and thereby cover up the original source of the request

Network File System (NFS)

- Provides access to a central file repository
  - Based on RPC and UDP
  - Robustness is of prime importance
  - Servers are stateless: clients keep state information
- NSF servers normally listen at UDP port 2049, but may listen at other UDP ports instead
- Access to a file requires a identification string for the file called the **file handle**
  - A file handle provides permanent access to a file or directory
  - File handles may be passed around

NFS Security Concerns

- A client receives a handle to the root directory at mount time
  - The root file handle gives access to the whole file system
- The client host is authenticated by its IP address
- Client authentication is only checked once

Web Service

- Utilizes multiple protocols
  - Principle protocol: Hypertext Transfer Protocol (HTTP)
  - Other protocols: FTP, Gopher, NNTP, WAIS, telnet, SMTP
- Web servers usually listen at TCP port 80, but may listen at many other TCP ports (e.g., 81, 8000, 8080, 8888, etc.)

Web Service Security Concerns

- Web server security concerns
  - A Web site is usually open to the whole Internet
  - Attackers can damage the Web server and modify or steal the information on a Web site
  - Attackers can use a Web site as an entrance to other services and resources on the server's host and network
  - Unauthorized information may be obtained via document requests
  - Unauthorized commands and programs may be executed via script requests
- Web browser security concerns
  - Web sites may be spoofed
  - Web documents may contain dangerous JavaScript or Java applets

Network Time Protocol (NTP)

- Synchronizes a host's clock with the rest of the Internet
  - NTP servers communicate with each other using both TCP and UDP
  - NTP servers listen at TCP and UDP ports 37
- Time should be synchronized across a network so that time-based protection mechanisms work correctly
  - Unsynchronized time may allow **replay attacks**
- NTP servers can be a target of attack
  - An attacker can only change a host's clock a small bit at a time
  - Over a period of time, an attacker could change the clock enough to surmount time-based protection mechanisms

Finger

- Provides personal information about an individual user on a host
  - Finger servers listen at TCP port 79
- Is very useful to someone probing your network
- A very dangerous service, finger should either be disabled or carefully managed