Overview

- Last Lecture
  - System installation
- This Lecture
  - Post installation
- Next Lecture
  - Wireless networking

Software Installation

- GNU software structure
  - lib, bin, sbin, etc, src
- GNU software installation
  - **./configure**
  - **make**
  - **make -n install:** before real installation.
  - **make install**
- Package management
  - **apt**, **rpm, yum, dpkg (the basic mechanism of apt)**

Post-configuration

- Create user accounts and environments
  - Sort out the access rights of different user groups
- Configure syslogd and klogd for log messages
  - Important for monitoring the system status and security
- Automate administrative tasks
  - Check and filter logs, clean disk space, intrusion detection
- Security of the system
  - Is my system protected from potential risks?
  - Hardware, data, and services are well protected?
  - Privacy is protected?

Unix maze

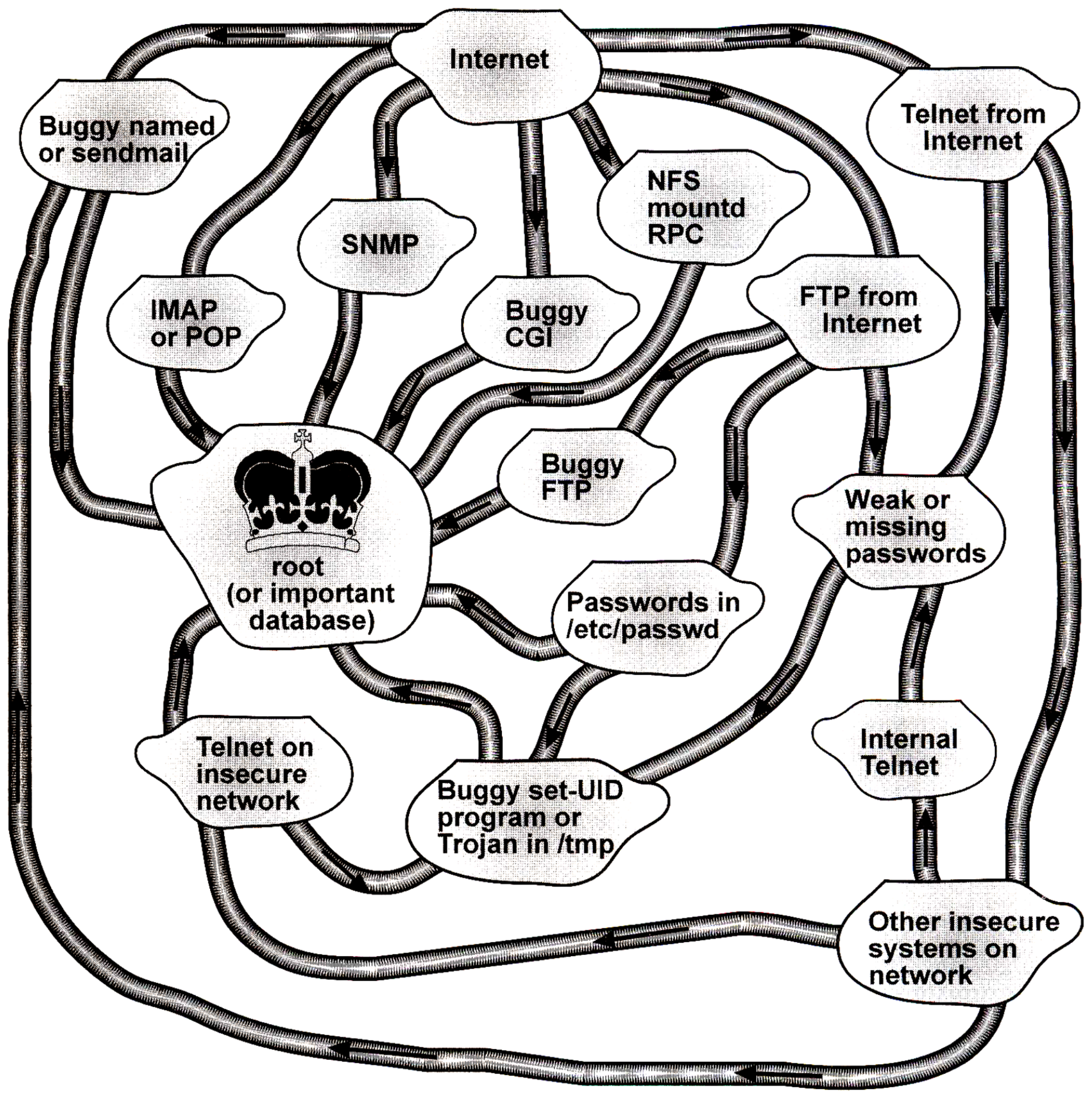
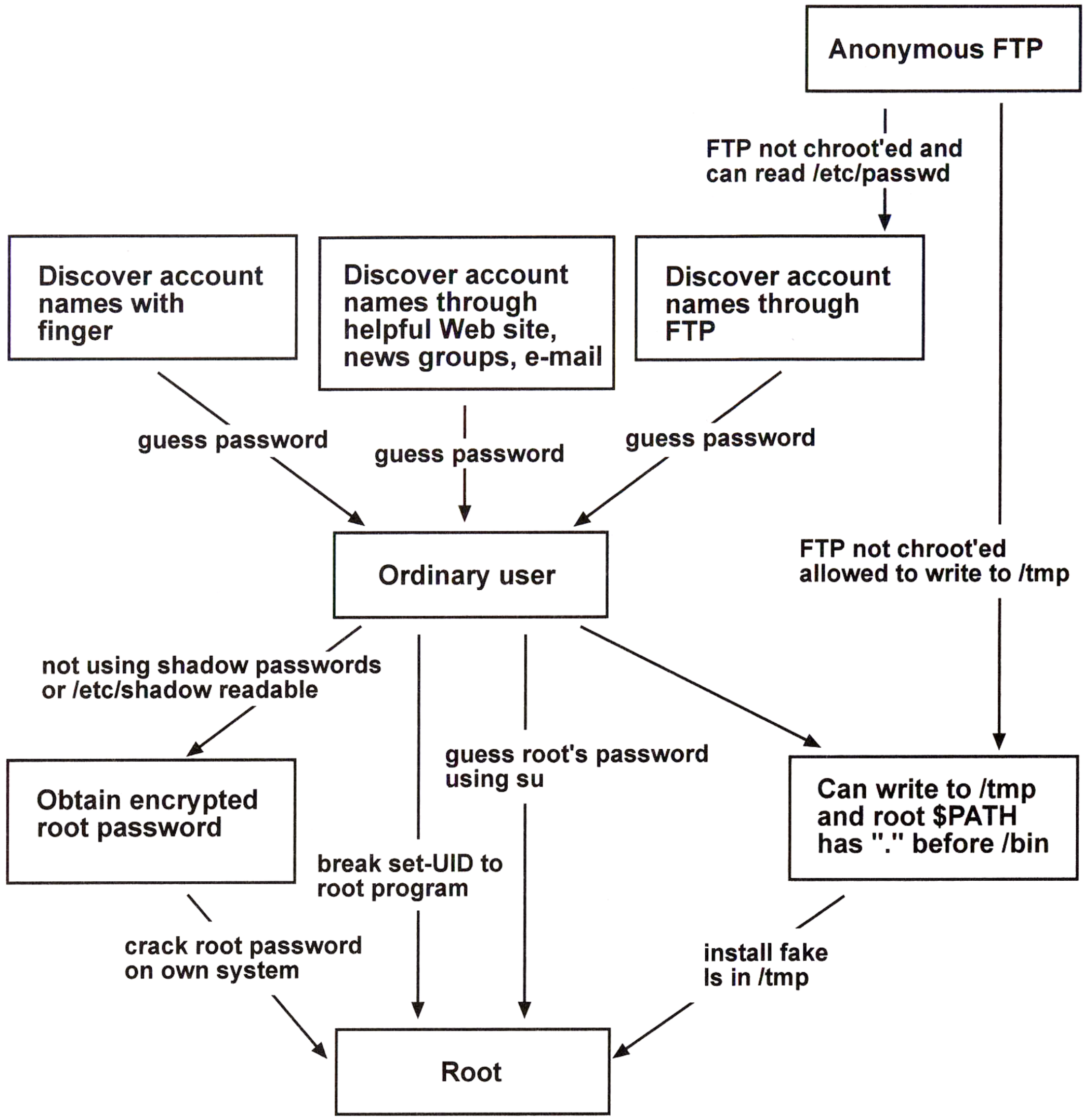


Figure 2.1 You are in a maze of twisty little passages.

Attack paths



Anonymous FTP
FTP not chroot'ed and can read /etc/passwd
Discover account names with finger
Discover account names through helpful Web site, news groups, e-mail
Discover account names through FTP
guess password
guess password
guess password
Ordinary user
FTP not chroot'ed allowed to write to /tmp
not using shadow passwords or /etc/shadow readable
guess root's password using su
Obtain encrypted root password
Can write to /tmp and root $PATH has "." before /bin
break set-UID to root program
crack root password on own system
install fake ls in /tmp
Root

Figure 2.2 Attack paths to root.

Why not set PATH=.:/bin:/sbin

- PATH variable decide search order of commands
- The above setting can find a command under the current directory first.
  - E.g. if you have a command named ls under the current directory, the local ls is executed, not /bin/ls
- If such ls (bash script) is put in /tmp
  - /bin/cp /etc/shadow ~
  - /bin/ls
- When the file system is full, with root privilege, the system admin can
  - cd /tmp; ls

Security issues

- Protect your physical equipment
  - Locked in a secure room?
  - Keyboard and power and reset buttons are accessible to attackers?
  - Floppy drive and CD-ROM are accessible to attackers?
  - Password is set for BIOS/EFI?
  - Is it possible for a Trojan horse to be installed?
- Detect potential attacks from Internet
  - Check log files and alert the SA by email
  - Remove unnecessary and insecure services

Security issues (cont.)

- Protect your system from attacks
  - Don’t set: Path= (.:/bin:/sbin:…
  - Avoid weak passwords: use a password suite to enforce certain rules
  - Shadow passwords: /etc/shadow
  - Close unneeded network ports
  - Check file system ownership and permissions: find permission problems
  - Use the least privilege rule for any user/program
  - Stale and unnecessary accounts
  - Avoid dangerous software with root privilege
  - Tools for automatically checking file system changes
  - Update old software versions

Services

- Different services from Internet
  - SSH, SMTP, NFS, WWW, DNS, DHCP, . . .
- Client/server model
  - Request and respond
- Proxies and agents
  - Services can be offered by proxies
  - Reasons: security, caching, censorship
- Use **nmap** to find insecure services or ports

Services

- Server programs (for services) are called daemons
  - Daemons have no terminal for standard input, output, error. They have to use log files for error reporting.
  - They are ever running once started and should not be controlled by terminals like ctrl+c.
- Two ways to start up daemons
  - Directly
  - Indirectly by **inetd** when needed
- Need to configure **/etc/services** for new services/daemons
- Normally daemons are started up when the system boots up (from the startup scripts **/etc/rc.d**)
- Each service uses a socket bound to a unique port number (well-known ports for well-known services)

Network daemons

- Daemons provide application services via the network
  - A daemon binds to a port, most commonly a well-known TCP/UDP port, and waits for incoming connections on it. If one occurs, the daemon accepts the connection, creates a child process that services the connection, while the parent continues to listen for further requests
- Normally one service needs one daemon and at least one instance of every possible service you wish to provide must be active in memory at all times
- Problems with many daemons
  - Most daemons are not frequently used but occupy memory space
  - To overcome these inefficiencies, most UNIX systems run a special network daemon **inetd**

The *inetd* super server

- Super Server **inetd**
  - Internet Daemon (port number of the daemon?)
  - It is started at system boot time and takes the list of services it is to manage from a startup file named **/etc/inetd.conf**
- The configuration file inetd.conf
  - An entry in the file **/etc/inetd.conf** consists of a single line made up of the following fields
    - *service type protocol wait user server cmdline*
  - If you want to start up some service, you should add a line in the file for that service
    - Use **kill -hup** to inform **inetd** to re-read the configuration file

The *inetd* super server (cont.)

- Fields in each line of ***inetd.conf***
  - service: gives the service name. The name is translated to a port number by looking it up in the **/etc/services** file
  - type: specifies a socket type, either stream or dgram (connection-oriented or connection-less)
  - protocol: gives the name of the transport protocol used by the service, e.g. tcp or udp. The names have to be valid protocols listed in **/etc/protocols**
  - user: owner of the server when it is running
    - Normally root, but sometimes may be nobody. Use the principle of least privilege.

The *inetd* super server (cont.)

- wait: It can be either wait or nowait. If wait is specified, **inetd** executes only one server for the specified port at any time; otherwise, it immediately continues to listen on the port after starting the server and may start multiple instances of servers at the same time
  - For most RPC (remote procedure call) servers specify wait; for multi-threaded servers specify nowait
- server: full path of the server program
- cmdline: the command line arguments to be passed to the server

The *inetd* super server (cont.)

- How inetd works?
  - Listens for connections on certain internet sockets (depending on the content of ***inetd.conf***)
  - When a connection is found on one of its sockets, it decides what service the socket corresponds to, and invokes a program to service the request
  - The daemon repeats the above steps.
- Insecure servers should be removed from *inetd.conf*
  - finger, tftp, ftp, telnet, rsh, rlogin, rexec, etc.
  - Use **netstat -a** to show all listened ports
- For more information about **inetd**, ***man inetd***

TCP wrapper

- TCP wrapper: tcpd
  - Allow access control to network services using **hosts.allow** and **hosts.deny**
  - Tcpd monitors incoming requests for telnet, finger, ftp, exec, rsh, rlogin, tftp, and other services.
  - Operation is as follows: whenever a request for a service arrives, the inetd daemon is directed to run the tcpd program instead of the desired server. tcpd logs the request and does some additional checks. If all is well, tcpd runs the appropriate server program and goes away.
  - Another way to use tcpd is to call tcpd library in programs

TCP wrapper (cont.)

- Checks in tcpd
  - Pattern access control: use **hosts.allow** and **hosts.deny** files in **/etc**. **hosts.allow** is checked first. If the incoming request matches one of the entries, the connection is allowed; otherwise, check the **hosts.deny**. If one of the entries matches in it, the connection is rejected. If none matches, the connection is allowed
  - User name can be checked with the RFC 931 protocol
  - Host name and address are checked with DNS service
    - Protects from IP spoofing
- Shell commands can be executed
  - When some suspicious connections are found, put in a log file the attacker's info. and send email to root
- For more information about **tcpd**, ***man 8 tcpd***
- For more information about access pattern in hosts.allow and hosts.deny, ***man 5 hosts_access***

Summary

- Attack case with $PATH=.:/bin:/sbin:…
  - How could it happen?
- Why do we use a super server like inetd?
- What is the difference between a daemon process and a normal process?
- How are the files **hosts.allow** and **hosts.deny** used for access control?