System Administration

Security

UNIX

Security

- **a pervasive problem**
  - *"A surprisingly large portion of the entire infrastructure must be trustworthy, including pieces you might not have realized were critical."*
  - *(see p.200, Frisch)*
- **threats may come from a variety of sources**
  - hacking, cracking, a curious/misguided employee, a sysadmin's mistake, …
- **effect of a loss of security may vary**
  - *"Effective thinking about security begins by considering potential losses rather than potential threats, …"*
- **how to handle security issues will vary**
  - *"Plan and prepare…never is almost always shorter than you think."*
  - *"Security begins and ends with people."*
- **you can never automate security 100%**
  - any security system has to be a judicious mix of policy, procedure and technology

Unix Security

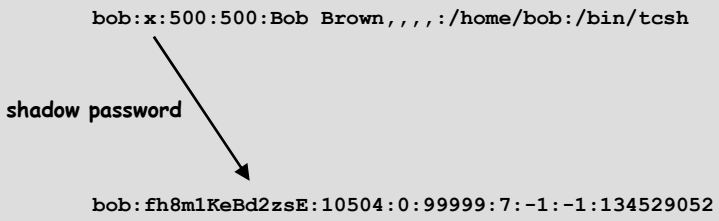
- Unix provides three major lines of defense
  - login authentication
  - filesystem security mechanisms
  - network monitoring and restriction

Login Authorization

- **first line of defense**
  - the login program looks up a supplied username and password in /etc/passwd
  - Unix is unusual in that it's password file is world-readable
    - *but the actual passwords in the file are encoded using a one-way encoding function*
      - easy to encode, VERY difficult to decode
    - *leaves things open to a dictionary-based attack method*
      - especially if poor passwords are chosen by the users
  - the shadow package now optional on most Unix systems aims to prevent this
    - */etc/passwd is split*
      - /etc/passwd holds everything it did before *except for* the encoded password
        - must remain world readable; many programs use it (incl. ls, for example)
      - /etc/shadow holds all password data
        - protected so that only root can read it
    - *also provides additional mechanisms*
      - configurable
      - password aging and expiration
      - control over a user's password selection

Shadow File

- contents of the /etc/shadow file:
  - login name
  - encrypted password
  - days since Jan 1, 1970 that password was last changed
  - days before password may be changed
  - days after which password must be changed
  - days before password is to expire that user is warned
  - days after password expires that account is disabled
  - days since Jan 1, 1970 that account is disabled
  - a reserved field
- defaults are set in /etc/login.defs
  - along with more obscure info



More Shadow Password Stuff

- shadow file can be edited directly
  - can also use various tools
    - *usermod*

```
# usermod -e 6/6/1999 bob
```

      - sets the account expiration date
    - *passwd*

```
# passwd -n 2 -x 365 -w 3 -i 60 bob
```

      - minimum password lifetime = 2 days; maximum password lifetime = 365 days; warning period = 3 days; inactive period = 60 days
    - *chage*

```
# chage -d bob
```

      - sets date of last password change; can be used to force a user to change his/her password at next login

```
#!/bin/csh
# force_passwd_change <username> -- run as root
chage -l $1 >& /dev/null
if ($status == 1) then
  echo force_passwd_change: invalid user \'$1\'
  exit 1
endif
set max=`grep ^$1\: /etc/shadow | awk -F: '{print $5}'`
chage -d `date +%D` $1
set today=`grep ^$1\: /etc/shadow | awk -F: '{print $3}'`
set yest=`expr $today - 1`
if ($max >= $yest) set max=`expr $yest - 1`
set date=`expr $yest - $max`
chage -M $max -I 2 -W 7 -d $date $1
```

Checking Password Security

- **structural verification: pwck & grpck**
  - verify the integrity of the system authentication information. Entries in /etc/passwd, /etc/group and /etc/shadow are checked to see that they have the proper format and valid data in each field.
- **miscellany**
  - npasswd & passwd+
    - *both are alternative passwd commands that do stricter (and configurable) checking on the proposed passwords to ensure that they can't be guessed easily*
      - superceded by the more general facility offered by PAM
  - chfn—change finger information in the GCOS field
  - chsh—change login shell
  - /etc/securetty
  - /etc/shells
  - COPS, Crack & SATAN
    - *third-party utilities*

PAM

- **Pluggable Authentication Modules**
  - passwd/shadow facility is system-wide and a bit crude
    - *need to have finer granularity*
      - may want to protect sensitive applications, etc.
    - *need an extensible and configurable system*
      - so that biometrics, etc. can be used if so desired
- **PAM**
  - a set of modules that can be configured to provide various authentication features to any PAM-aware application
    - *in Linux, the system login/passwd applications are PAM-aware*
    - *also samba, ppp, rsh, ftp, etc.*
  - /etc/pam.d
    - *configuration directory*
      - contains a file for each service making use of PAM
      - file specifies how to do authentication
        - what steps are required, what code module to call at each step and what to do based on the return code from each module

The /etc/pam.d/rlogin File

- a good representative example
  - specifies a number of 'stacked' tests that may be applied by PAM on behalf of the service. The module-type field specifies whether success is absolutely required, is optional or "no need to try anything else" (sufficient).
  - also specifies arguments, such as "use_authtok", (ensures that the pam_pwdb module does not prompt for a password, but instead uses the one provided by pam_cracklib)
  - ***"Most Red Hat Linux users will never need to touch this…When you use RPM to install programs that need to do authentication, they automatically make the changes that are needed to do normal password authentication."***

```
#%PAM-1.0
# service-name module-type control-flag module-path arguments
auth     required  /lib/security/pam_securetty.so
auth     required  /lib/security/pam_pwdb.so shadow nullok
auth     required  /lib/security/pam_nologin.so
account  required  /lib/security/pam_pwdb.so
password required  /lib/security/pam_cracklib.so
password required  /lib/security/pam_pwdb.so shadow nullok use_authtok
session  required  /lib/security/pam_pwdb.so
```

/etc/pam.d/rlogin Explanation

- the first 2 lines are comments
- the next three lines stack up three modules to use for login authorization
  - check that /etc/securetty allows the user to log in on the tty being used
  - causes the user to be asked for a password and the supplied password checked
  - checks to see if the file /etc/nologin exists
  - all three modules are checked, even if one fails
- line 6 does any necessary login accounting
- line 7 checks the password against a set of rules
- line 8 specifies that if the login program changes the user's password, it should use the module in pam_pwdb.so to do so
- the final line specifies that the pam_pwdb.so module should be used to manage the login session
- the order of the lines within each file matters

Filesystem Security

- revolves around the built-in filesystem protection mechanisms

```
# ls -lga /tmp
total 90
drwsrwxrwx   4  root root    1024  Dec   4 07:30  ./
drwxr-xr-x 18  root root    1024  Sep  27 08:07  ../
-rwxr-xr-x   1  root root    4094  Oct  30 21:38  a.out*
-rwxrwxr-x   1  bob  bob     8363  Oct  30 21:32  conftest*
-rw-------   1  bob  bob    56770  Dec   1 18:31  fvwmrca00704
-rw-r--r--   1  root root     381  Nov   8 15:12  glint.gif
-rw-rw-r--   1  root root    4724  Nov  18 21:17  hw_survey.txt
drwxr-xr-x   2  root root   12288  Sep  27 08:06  lost+found/
-rw-r--r--   1  root bob       11  Nov   2 20:35  lpq.00002b98
-rw-rw-r--   1  bob  bob       54  Oct  30 21:22  x.c
```

- also around the user/group facility

Things To Consider

- correct ownerships & protections for system files and directories
  - generally owned by root or a pseudo-user
- *each* SETUID and SETGID file is a potential security hole
  - *(see p.227 & p.231, Frisch)*
  - can be found by

```
# find / -type f \( -perm -2000 -o -perm -4000 \) -ls
```

- users home directory trees shouldn't have world-writeable subdirectories
- user's '.' files shouldn't be writeable by anyone else
  - backdoor used by the earlier example
- groups need to be thought about
  - all-or -nothing approach, so not a perfect mechanism
    - *ACL mechanism provided by AIX and HP/UX (& Windows NT...)*
- need to be able to detect changes

tripwire

- "unquestionably among the finest freely-available software packages in existence."
  - *(see p.256, Frisch)*
  - compares the current state of important files and directories with their stored correct attributes
    - *according to criteria selected by the system administrator*
      - can take a long time to prepare the initial database
        - it can only detect changes if it originally looked at a property
    - *ASCii database of attributes*
      - should be stored offline to prevent compromise
    - *reports changes*
      - clear format but voluminous
        - may make it difficult to decide what change was legal, what was not
  - may be invaluable for a relatively static filesystem, probably not good for user/project home areas
    - *also good for detecting tampering in a firewall or some other dedicated server*
- ***(see also COPS, p.257, Frisch)***

Network Security

- a very big area!
  - very difficult to guard against ALL forms of network threat
    - (see p.619, Frisch)
  - common attacks
    - *readable password files that contains poorly-chosen passwords*
    - *services that do insufficient authentication*
      - NFS, X
    - *buggy services*
      - sendmail is notorious
    - *misuse of standard facilities*
      - .rhosts
    - *IP spoofing*
    - *denial of service*
      - "ping of death"
  - often it is a combination of 'cracks' that lead to a big security hole

Trust

- Unix contains two mechanisms for trusting users and systems
  - both can cause security problems
    - *host-level equivalence: /etc/hosts.equiv*
      - list of hostnames
        - listed hosts are treated as being 'equivalent' to the host that owns the file
      - if a user from a listed host attempts to login, and also has a login on the owner host he/she will *not* be asked for a password
      - makes sense in an ideal world; problem arises when a user account on an equivalent host has been compromised…
      - use of /etc/hosts.equiv is *discouraged*

```
server
ftp
www
admin
```

    - *user-level equivalence: ~/.rhosts*
      - list of hostnames (may also give user names)
      - allows password-less access to an account from a remote machine
        - from a user with the same name
        - from a user with a different name, if that name has been associated with the host
      - same problems as above
      - root should ***never*** have a .rhosts file
        - don't want anyone masquerading as root. EVER!
        - best to never allow remote root access: login as a normal user and use sudo or su
        - *(see p.624, Frisch)*

```
host1
host2 fred
host.other.dom harry
```

TCP Wrappers

- monitors and filters incoming requests for the SYSTAT, FINGER, FTP, TELNET, RLOGIN, RSH, EXEC, TFTP, TALK, and other network services
- the access control software consults two files:
  - access will be granted when a (daemon, client) pair matches an entry in the /etc/hosts.allow file
  - otherwise, access will be denied when a (daemon, client) pair matches an entry in the /etc/hosts.deny file
  - otherwise, access will be granted
- a non-existent access control file is treated as if it were an empty file: access control is turned off if there are no access control files

Configuring TCP Wrappers

- protected services are listed in /etc/inetd.conf
  - specify the use of /usr/sbin/tcpd

```
ftp      stream tcp nowait root /usr/sbin/tcpd in.ftpd -l -a
telnet   stream tcp nowait root /usr/sbin/tcpd in.telnetd
gopher   stream tcp nowait root /usr/sbin/tcpd gn
```

    - *tcpd can allow or deny access to a service based on the origin of the request, and the configuration in /etc/hosts.allow and /etc/hosts.deny*

```
# /etc/hosts.allow
ALL: redhat.com .redhat.com
in.talkd: ALL
in.ntalkd: ALL
in.fingerd: ALL
in.ftpd: ALL
```

```
# /etc/hosts.deny
in.tftpd: ALL: (/some/where/safe_finger -l @%h | \
  /usr/ucb/mail -s %d-%h root) &
```

  - tcpd and hosts_access (5) manual pages give details
  - events are logged via the standard syslog facility
  - *(see p.625-6, Frisch)*

TCP Wrappers Utilities

- **tcpdchk**
  - checks the configuration files for potential problems
    - *needed because it can be difficult to define correct settings*
- **tcpdmatch**
  - checks the configuration against a *virtual* request from the network—lets you know whether a request would be denied or accepted
- ***"Programs like tcpdchk and tcpdmatch are excellent complements to the security program tcpwrapper because they let you head off security problems before they happen."***

SATAN

- **Security Administrator Tool for Analyzing Neworks**
  - probes a network for security vulnerabilities
  - does many security-related tests
    - *configurable*
    - (p.628, Frisch)
  - *"one good thing about SATAN is that its documentation tells you how to fix the vulnerabilities that it finds."*
  - *SATAN is now a favorite tool of crackers*
    - *caused a furore when introduced*
      - Dan Farmer (the writer) eventually lost his job at SGI!
    - *often banned by nervous organizations*
      - futile: *"obscurity is not security"*
    - *"...SATAN was written to improve internet security. Don't put our work to shame."*

swatch

- helps to filter all the copious error messages that a busy system generates
  - may difficult to find messages of real importance
    - *swatch helps with this*
  - *(p.627, Frisch)*

Other

- **other services are also configurable**
  - samba (extensively!), apache, etc.
  - example: ftp
    - */etc/ftpaccess*
      - defines most of the access control for the ftp server. Can: set up logical groups to control access from different sites, limit the number of simultaneous FTP connections, configure transfer logging, etc.
    - */etc/ftphosts*
      - used to allow or deny access to certain accounts from various hosts
    - */etc/ftpusers*
      - lists all the users that are not allowed to ftp into your machine. For example, root is listed in /etc/ftpusers by default