Chapter 28: User Security

- Policy
- Access
- Files, devices
- Processes
- Electronic communications

Policy

- Assume user is on Drib development network
  - Policy usually highly informal and in the mind of the user
- Our users' policy:

U1 Only users have access to their accounts

U2 No other user can read, change file without owner's permission

U3 Users shall protect integrity, confidentiality, availability of their files

U4 Users shall be aware of all commands that they enter or that are entered on their behalf

Access

- U1: users must protect access to their accounts
  - Consider points of entry to accounts
- Passwords
- Login procedure
- Leaving system

Passwords

- Theory: writing down passwords is ***BAD***!
- Reality: choosing passwords randomly makes them hard to remember
  - If you need passwords for many systems, assigning random passwords and *not* writing something down won't work
- Problem: Someone can read the written password
- Reality: degree of danger depends on environment, how you record password

Isolated System

- System used to create boot CD-ROM
  - In locked room; system can *only* be accessed from within that room
    - No networks, modems, etc.
  - Only authorized users have keys
- Write password on whiteboard in room
  - Only people who will see it are authorized to see it

Multiple Systems

- Non-infrastructure systems: have users use same password
  - Done via centralized user database shared by all non-infrastructure systems
- Infrastructure systems: users may have multiple accounts on single system, or may not use centralized database
  - Write down transformations of passwords

Infrastructure Passwords

- Drib devnet has 10 infrastructure systems, 2 lead admins (Anne, Paul)
  - Both require privileged access to all systems
  - root, Administrator passwords chosen randomly
- How to remember? Memorize an algorithm!
  - Anne: “change case of 3rd letter, delete last char”
  - Paul: “add 2 mod 10 to first digit, delete first letter”
- Each gets printout of transformed password

Papers for Anne and Paul

Actual password	Anne's version	Paul's version
C04cEJxX	C04ceJxX5	RC84cEJxX
4VX9q3GA	4VX9Q3GA2	a2VX9q3GA
8798Qqdt	8798QqDt$	67f98Qqdt
3WXYwgnw	3WXywgnwS	Z1WXYwgnw
feOioC4f	feoioC4f9	YfeOioC2f
VRd0Hj9E	VRD0Hj9Eq	pVRd8Hj9E
e7Bukcba	e7BUkcbaX	Xe5Bukcba
ywyj5cVw	ywYj5cVw*	rywyj3cVw
5iUikLB4	5iUIkLB4m	3JiUikLB4
af4hC2kg	af4HC2kg+	daf2hC2kg

Non-Infrastructure Passwords

- Users can pick
  - Proactive password checker vets proposed password
- Recommended method: passwords based on obscure poems or sayings
  - Example: “ttrs&vmbi” from first letter of second, fourth words of each line, putting “&” between them:

    He took his vorpal sword in hand:
    Long time the manxome foe he sought—
    So rested he by the Tumtum tree,
    And stood awhile in thought.

  Third verse of *Jabberwocky*, from *Alice in Wonderland*

Analysis

- Isolated system meets U1
  - Only authorized users can enter room, read password, access system
- Infrastructure systems meet U1
  - Actual passwords not written down
  - Anne, Paul don't write down algorithms
  - Stealing papers does not reveal passwords
- Non-infrastructure systems meet U1
  - Proactive password checker rejects easy to guess passwords

Login Procedure

- User obtains a prompt at which to enter name
- Then comes password prompt
- Attacks:
  - Lack of mutual authentication
  - Reading password as it is entered
  - Untrustworthy trusted hosts

Lack of Mutual Authentication

- How does user know she is interacting with legitimate login procedure?
  – Attacker can have Trojan horse emulate login procedure and record name, password, then print error message and spawn real login
- Simple approach: if name, password entered incorrectly, prompt for retry differed
  – In UNIX V6, it said "Name" rather than "login"

More Complicated

- Attack program feeds name, password to legitimate login program on behalf of user, so user logged in without realizing attack program is an intermediary
- Approach: trusted path
  - Example: to log in, user hits specified sequence of keys; this traps to kernel, which then performs login procedure; key is that no application program can disable this feature, or intercept or modify data sent along this path

Reading Password As Entered

- Attacker remembers it, uses it later
  - Sometimes called “shoulder surfing”
  - Can also read chars from kernel tables, passive wiretapping, etc.
- Approach: encipher all network traffic to defeat passive wiretapping
  - Drib: firewalls block traffic to and from Internet, internal hosts trusted not to capture network traffic
  - Elsewhere: use SSH, SSL, TLS to provide encrypted tunnels for other protocols or to provide encrypted login facilities

Noticing Previous Logins

- Many systems print time, location (terminal) of last login
  - If either is wrong, probably someone has unauthorized access to account; needs to be investigated
- Requires user to be somewhat alert during login

Untrustworthy Trusted Hosts

- Idea: if two hosts under same administrative control, each can rely on authentication from other
- Drib does this for backups
  - Backup system logs into workstation as user “backup”
    - If password required, administrator password needs to be on backup system; considered unacceptable risk
    - Solution: all systems trust backup server
- Requires accurate identification of remote host
  - Usually IP address
  - Drib uses challenge-response based on cryptography

Analysis

- Mutual authentication meets U1
  – Trusted path used when available; other times, system prints time, place of last login
- Protecting passwords meets U1
  – Unencrypted passwords only placed on trusted network; also, system prints time, place of last login
- Trusted hosts meets U1
  – Based on cryptography, not IP addresses; number of trusted systems minimal (backup system only)

Leaving the System

- People not authorized to use systems have access to rooms where systems are
  - Custodians, maintenance workers, etc.
- Once authenticated, users must control access to their session until it ends
  - What to do when one goes to bathroom?
- Procedures used here

Walking Away

- Procedures require user to lock monitor
  - Example: X window system: *xlock*
    - Only user, system administrator can unlock monitor
  - Note: be sure locking program does not have master override
    - Example: one version of lock program allowed anyone to enter "Hasta la vista!" to unlock monitor

Modems

- Terminates sessions when remote user hangs up
  - Problem: this is configurable; may have to set physical switch
    - If not done, next to call in connects to previous user's session
  - Problem: older telephone systems may mishandle propagation of call termination
    - New connection arrives at telco switch and is forwarded before termination signal arrives at modem
    - Same effect as above
- Drib: no modems connected to development systems

Analysis

- Procedures about walking away meet U1
  - Screen locking programs required, as is locking doors when leaving office; failure to do so involves disciplinary action
  - If screen locking password forgotten, system administrators can remotely access system and terminate program
- Procedures about modems meet U1
  - No modems allowed; hooking one up means getting fired (or similar nasty action)

Files and Devices

- File protection allows users to refine protection afforded their data
  - Policy component U2 requires this
- Users manipulate system through devices, so their protection affects user protection as well
  - Policy components U1, U4 require this

Files

- Often different ways to do one thing
  - UNIX systems: Pete wants to allow Deb to read file *design*, but no-one else to do so
    - If Pete, Deb have their own group, make file owned by that group and group readable but not readable by others
    - If Deb only member of a group, Pete can give group ownership of file to Deb and set permissions appropriately
    - Pete can set permissions of containing directory to allow himself, Deb's group search permission
  - Windows NT: same problem
    - Use ACL with entries for Pete, Deb only:

      { (Pete, full control), (Deb, read) }

File Permission on Creation

- Use template to set or modify permissions when file created
  - Windows NT: new directory inherits parent's ACL
  - UNIX systems: identify permissions to be denied
    - *umask* contains permissions to be disabled, so can say "always turn off write permission for everyone but owner when file created"

Group Access

- Provides set of users with same rights
- Advantage: use group as role
  - All folks working on Widget-NG product in group *widgetng*
  - All files for that product group readable, writable by *widgetng*
  - Membership changes require adding users to, dropping users from group
    - No changes to file permissions required

Group Access

- Disadvantage: use group as abbreviation for set of users; changes to group may allow unauthorized access or deny authorized access
  - Maria wants Anne, Joan to be able to read *movie*
  - System administrator puts all in group *maj*
  - Later: sysadmin needs to create group with Maria, Anne, Joan, and Lorraine
    - Adds Lorraine to group *maj*
    - Now Lorraine can read *movie* even though Maria didn't want her to be able to do so

File Deletion

- Is the *name* or the *object* deleted?
- Terms
  - File attribute table: contains information about file
  - File mapping table: contains information allowing OS to access disk blocks belonging to file
  - Direct alias: directory entry naming file
  - Indirect alias: directory entry naming special file containing name of target file
- Each direct alias is alternative name for same file

Rights and Aliases

- Each direct alias can have different permissions
  - Owner must change access modes of each alias in order to control access
- Generally false
  - File attribute table contains access permissions for each file
    - So users can use any alias; rights the same

Deletion

- Removes directory entry of file
  - If no more directory entries, data blocks and table entries released too
  - Note: deleting directory entry does *not* mean file is deleted!

Example

- Anna on UNIX wants to delete file *x*, setuid to herself
  - `rm x` works if no-one else has a direct alias to it
  - Sandra has one, so file not deleted (but Anna's directory entry is deleted)
    - File still is setuid to Anna
- How to do this right:
  - Turn off all permissions on file
  - *Then* delete it
    - Even if others have direct links, they are not the owners and so can't change permissions or access file

Persistence

- Disk blocks of deleted file returned to pool of unused disk blocks
- When reassigned, new process may be able to read previous contents of disk blocks
  - Most systems offer a “wipe” or “cleaning” procedure that overwrites disk blocks with zeros or random bit patterns as part of file deletion
  - Useful when files being deleted contain sensitive data

Direct, Indirect Aliases

- Some commands act differently on these
  - Angie executes command to add permission to file to let Lucy read it
  - If file name direct alias, works
  - If file name indirect alias, does it add permission to the indirect alias or the file itself?
- Semantics of systems, commands on systems differ
  - Example: on RedHat Linux 7.1, when given indirect alias of file, *chmod* changes permissions of actual file, *rm* deletes indirect alias

Analysis

- Use of ACLs, *umask* meet U2
  - Both set to deny permission to”other” and “group” by default; user can add permissions back
- Group access controls meet U2
  - Membership in groups tightly controlled, based on least privilege
- Deletion meets U2
  - Procedures require sensitive files be wiped when deleted

Devices

- Must be protected so user can control commands sent, others cannot see interactions
- Writable devices
- Smart terminals
- Monitors and window systems

Writable Devices

- Restrict access to these as much as possible
- Example: tapes
  - When process begins writing, ACL of device changes to prevent other processes from writing
  - Between mounting of media, process execution, another process can begin writing
  - Moral: write protect all mounted media unless it is to be written to
- Example: terminals
  - Write control sequence to erase screen—send repeatedly

Smart Terminals

- Has built-in mechanism for performing special functions
  - Most important one: block send
  - The sequence of chars initiating block send do *not* appear on screen
- Write Trojan horse to send command from user's terminal
- Next slide: example in mail message sent to Craig
  - When Craig reads letter, his startup file becomes world writable

Trojan Horse Letter

```
Dear Craig,
Please be careful. Someone may ask you to execute
chmod 666 .profile
You shouldn't do it!
Your friend,
Robert
<BLOCK SEND (-2,18), (-2,18)><BLOCK SEND
(-3,0),(3,18)><CLEAR>
```

Why So Dangerous?

- With writable terminal, someone must trick user of that terminal into executing command; both attacker *and user* must enter commands
- With smart terminal, only attacker need enter command; if user merely reads the wrong thing, the attacker's compromise occurs

Monitors and Window Systems

- Window manager controls what is displayed
  - Input from input devices
  - Clients register with manager, can then receive input, send output through manager
- How does manager determine client to get input?
  - Usually client in whose window input occurs
- Attack: overlay transparent window on screen
  - Now all input goes through this window
  - So attacker sees all input to monitor, including passwords, cryptographic keys

Access Control

- Use ACLs, C-Lists, etc.
- Granularity varies by windowing system
- X window system: host name or token
  - Host name, called *xhost* method
  - Manager determines host on which client runs
  - Checks ACL to see if host allowed to connect

X Windows Tokens

- Called *xauth* method
  - X window manager given random number (*magic cookie*)
    - Stored in file ".Xauthority" in user's home directory
  - Any client trying to connect to manager must supply this magic cookie to succeed
    - Local processes run by user can access this file
    - Remote processes require special set-up by user to work

Analysis

- Writable devices meet U1, U4
  – Devnet users have default settings denying all write access to devices except the user
- Smart terminals meet U1, U4
  – Drib does not allow use of smart terminals except on systems where *all* control sequences (such as BLOCK SEND) are shown as printable chars
- Window managers meet U1, U4
  – Drib uses either xhost or token (xhost by default) on a trusted network, so IP spoofing not an issue

Process

- Manipulate objects, including files
  - Policy component U3 requires users to be aware of how
- Copying, moving files
- Accidentally overwriting or erasing files
- Encryption, keys, passwords
- Start-up settings
- Limiting privileges
- Malicious logic

Copying Files

- Duplicates contents
- Semantics determines whether attributes duplicated
  - If not, may need to set them to prevent compromise
- Example: Mona Anne copies *xyzzy* on UNIX system to *plugh:*

```
cp xyzzy plugh
```

  - If *plugh* doesn't exist, created with attributes of *xyzzy* except any setuid, setgid discarded; contents copied
  - If *plugh* exists, attributes not altered; contents copied

Moving Files

- Semantics determines attributes
- Example: Mona Anne moves *xyzzy* to /*tmp*/*plugh*
  - If both on same file system, attributes unchanged
  - If on different file systems, semantically equivalent to:

```
cp xyzzy /tmp/plugh
rm xyzzy
```

    Permissions may change …

Accidentally Overwriting Files

- Protect users from themselves
- Example: deleting by accident
  - Intends to delete all files ending in ".o"; pattern is "*.o", "*" matching any string
  - Should type `rm *.o`
  - Instead types `rm * .o`
  - All files in directory disappear!
- Use modes to protect yourself
  - Give –i option to rm to prevent this

Encryption

- Must trust system
  - Cryptographic keys visible in kernel buffers, swap space, and/or memory
  - Anyone who can alter programs used to encrypt, decrypt can acquire keys and/or contents of encrypted files
- Example: PGP, a public key encryption program
  - Protects private key with an enciphering key ("pass-phrase"), which user supplies to authenticate file
  - If keystroke monitor installed on system, attacker gets pass-phrase, then private key, then message

Saving Passwords

- Some systems allow users to put passwords for programs in files
  - May require file be read-protected but *not* use encryption
- Example: UNIX *ftp* clients
  - Users can store account names, host names, passwords in *.netrc*
  - Kathy did so but *ftp* ignored it
  - She found file was readable by anyone, meaning her passwords stored in it were now compromised

Start-Up Settings

- When programs start, often take state info, commands from environment or start-up files
  - Order of access affects execution
- Example: UNIX command interpreter *sh*
  - When it starts, it does the following:
    - Read start-up file /*etc*/*profile*
    - Read start-up file .*profile* in user's home directory
    - Read start-up file named in environment variable **ENV**
  - Problem: if any of these files can be altered by untrusted user, *sh* may execute undesirable commands or enter undesirable state on start

Limiting Privileges

- Users should know which of their programs grant privileges to others
  - Also the implications of granting these
- Example: Toni reads email for her boss, Fran
  - Fran knew not to share passwords, so she made a setuid-to-Fran shell that Toni could use
    - Bad idea; gave Toni too much power
  - On Toni's suggestion, Fran began to forward to Toni a copy of every letter
    - Toni no longer needed access to Fran's account

Malicious Logic

- Watch out for search paths
- Example: Paula wants to see John's confidential designs
  - Paula creates a Trojan horse that copies design files to /tmp; calls it *ls*
  - Paula places copies of this in all directories she can write to
  - John changes to one of these directories, executes *ls*
    - John's search path begins with current working directory
  - Paula gets her information

Search Paths

- Search path to locate program to execute
- Search path to locate libraries to be dynamically loaded when program executes
- Search path for configuration files
- …

Analysis

- Copying, moving files meets U3
  - Procedures are to warn users about potential problems
- Protections against accidental overwriting and erasing meet U3
  - Users' startup files set protective modes on login
- Passwords not being stored unencrypted meets U3
  - In addition to policy, Drib modified programs that accept passwords from disk files to ignore those files

Analysis (*con't*)

- Publicizing start up procedures of programs meets U3
  - Startup files created when account created have restrictive permissions
- Publicizing dangers of setuid, giving extra privileges meets U3
  - When account created, no setuid/setgid programs
- Default search paths meet U4
  - None include world writable directories; this includes symbol for current working directory

Electronic Communications

- Checking for malicious content at firewall can make mistakes
  - Perfect detectors require solving undecidable problem
  - Users may unintentionally send out material they should not
- Automated e-mail processing
- Failing to check certificates
- Sending unexpected content

Automated E-mail Processing

- Be careful it does not automatically execute commands or programs on behalf of other users
- Example: NIMDA worm, embedded in email
  - When user opens letter, default configuration of mail passed NIMDA attachment to another program to be displayed
  - This executes code comprising worm, thereby infecting system

Failure to Check Certificates

- If certificate invalid or expired, email signed by that certificate may be untrustworthy
  - Mail readers must check that certificates are valid, or enable user to determine whether to trust certificate of questionable validity
- Example: Someone obtained certificates under the name of Microsoft
  - When discovered, issuer *immediately* revoked both
  - Had anyone obtained ActiveX applets signed by those certificates, would have been trusted

Sending Unexpected Content

- Arises when data sent in one format is viewed in another
- Example: sales director sent sales team chart showing effects of proposed reorganization
  - Spreadsheet also contained confidential information deleted from spreadsheet but still in the file
  - Employees used different system to read file, seeing the spreadsheet data—and also the "deleted" date
- Rapid saves often do not delete information, but rearrange pointers so information appears deleted

Analysis

- Automated e-mail processing meets U4
  - All programs configured not to execute attachments, contents of letters
- Certificate handling procedures meet U4
  - Drib enhanced all mail reading programs to validate certificates as far as possible, and display certificates it could not validate so user can decide how to proceed
- Publicizing problems with risk of “deleted” data meets U4
  - Also, progams have “rapid saves” disabled by default

Key Points

- Users have policies, although usually informal ones
- Aspects of system use affect security even at the user level
  - System access issues
  - File and device issues
  - Process management issues
  - Electronic communications issues