Chapter 20
Silberschatz and Galvin

Security

Protection and Security

- Protection is a strictly *internal* problem
  - Protection: mechanism for controlling the access of programs, processes, or users to the resources defined by a computer system. Must be able to specify controls to be imposed and must provide some form of enforcement.
- Security also must deal with *external* problems
  - Security: a measure of confidence that the integrity of a system and its data will be preserved.

Security

- Security must consider external environment of the system, and protect it from:
  - unauthorized access.
  - malicious modification or destruction.
  - accidental introduction of inconsistency.
- Easier to protect against accidental than malicious misuse.

Security

- Security measures must be taken at two levels
  - Physical: site (or sites) containing the computer systems must be physically secured against intruders
  - Human: Users must be screened carefully to reduce the chance of a user authorizing access for an intruder

Physical security

- Some things to consider:
  - – securing computer sites
  - – securing communication links
    - listening to network traffic (net snoopers)
    - protecting microwave links
  - – electromagnetic radiation from computer sites

Security facets

- Security facets: major ones are data loss and intruders
  - – Data loss (relatively manageable by backups)
  - – Acts of God (fire, floods, earthquakes, wars, etc.)
  - – Hardware or software errors: Unreadable disks and tapes, program bugs, etc.
  - – Human errors: Incorrect data entry, wrong tape or disk mounted, etc.
  - – Intruders: passive (just looking around) and active (modifying data)

Security facets

- Some categories include:
  - Casual prying, snooping by insiders
  - Determined attempts to make money (stealing the rounded interest; siphoning off unused accounts; blackmail)
  - Espionage, commercial and governmental
  - Malicious abuse (consume system resources; destroy or alter data; etc.)

Security flaws

- Errors in system program designs: suid version of lpr that removed files without checking privileges
- Insufficient legality checking: (general problem with suid shell script)
  - Process composed of two actions. User interrupts suid process after first action, modifies environment, continues with second
  - concrete example: mkdir used to (1) create directory with mknod and then (2) chown owner from root to user.
  - mkdir foo
  - mknod (associates name foo with new inode)
    - pause mkdir
    - rm foo and link some system file to it
    - resume mkdir
    - chown (user now owns the system file)

Authentication

- User identity most often established through passwords,can be considered a special case of either keys or capabilities.
- Passwords must be kept secret.
  - Frequent change of passwords.
  - Use of “non-guessable” passwords (user-chosen passwords are often easy to guess)
  - Log all invalid access attempts.

Authentication

- Password stealing
  - Easiest way is through social means (see following)
  - Technological approaches also
    - simple one: leave program running on a terminal that fakes the login sequence. Capture user name and password to a file and then exit with a fake error message, returning control to the real login process
  - Unix password files used to be openly available (encrypted password). Lends itself to brute-force cracking. Unfortunately some programs require access to the password file to run (e.g., mail)
    - also unfortunately Unix only uses first eight characters of password

Authentication

- Note that many of the most effective breaches of security are not technological
  - fake deposit slips
  - easily guessable passwords
  - calling people on the phone and asking for passwords (or Credit Card numbers, for that matter)

Authentication

- Passwords
  - Some Unix encryption algorithms incorporate a wait before returning values
  - Some versions of login process insert longer and longer waits after incorrect password specification
  - Others just simply return fake prompts after some number of failures

Authentication

- TENEX Password problem
  - TENEX (PDP 10) included way of invoking user function on each page fault (to permit user monitoring of program's behavior)
  - Also required passwords to access file
    - Scenario: arrange password so it falls across page boundaries: A/AAAAA. Attempt to access file. If page fault before "Illegal access"then first character right. If not, change character and try again. Only 128 characters so you'll get it right eventually! Then move on to character two of the password: VA/AAAA, and so on. 128*n versus 128^n different trials.

Authentication

- Consider effect of password lookup routine that operates faster for correct passwords
  - Hence guessing program can determine quickly if password is valid; if no answer within a specified time, assume you have guessed the wrong password.

Authentication

- One-time passwords
  - paired passwords
    - user is challenged and must respond with correct answer
  - passwords that are different times used
    - SecureID
      - hardware calculator
      - current time is used as a random seed
      - user enters a personal identification number
      - display shows the one-time password

SecureID

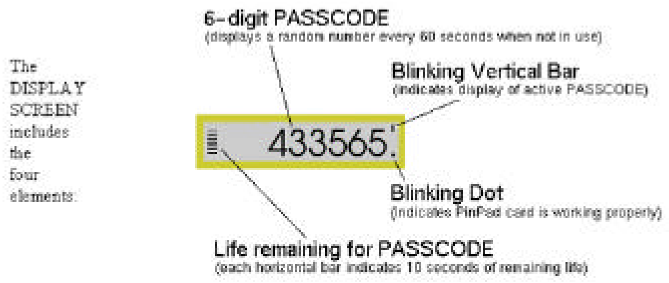
← DISPLAY SCREEN in upper right hand corner

← numeric KEYPAD below the word SECURID
← P key (protect/clear key) below the number 9
← ♦ key (enter key) below the number 7

(NOT SHOWN: serial number of card on reverse)

SecurID® PinPad™ card[2]

SecureID


The DISPLAY SCREEN includes the four elements:
6-digit PASSCODE
(displays a random number every 60 seconds when not in use)
Blinking Vertical Bar
(indicates display of active PASSCODE)
433565
Blinking Dot
(indicates PinPad card is working properly)
Life remaining for PASSCODE
(each horizontal bar indicates 10 seconds of remaining life)

Program threats

- Trojan horse
  - Code segment that misuses its environment.
  - Exploits mechanisms for allowing programs written by users to be executed by other users.
  - example already given in fake login process
  - another example--unexpected effects of including “.” early on a Unix path
    - Why . is no longer first on many people's path (e.g., executable file named "ls" in /tmp). If behaves as real ls (but not showing itself) many people will never notice!

Program threats

- Trap doors left by some previously trusted person
  - Doomsday machine scenarios of “trash the system if I am not in the payroll file”
  - Insert special code in system programs that give special privileges to specific users or on specific password
  - But this is easy to detect---how about putting code in the compiler to generate such instructions for any program compiled with the compiler?
    - People rarely compare object to source!
    - Still possible to detect if someone else starts maintaining compiler. But what if we are bootstrapping and the code is in an early version of the compiler but then taken out? Object code retains the functions but source doesn't show them anymore!

System threats

- Worms – use spawn mechanism; standalone program.
  - Internet worm (more later)
    - Exploited UNIX networking features (remote access) and bugs in finger and sendmail programs.
    - Grappling hook program uploaded main worm program.
  - Viruses: fragment of code embedded in a legitimate program; spread into other programs.

Internet worm (November 2, 1988)

- Caused by Robert Tappan Morris, Jr.
  - Sr. is NSA expert, previously at Bellcore
- Consisted of a bootstrap and the worm proper
- Bootstrap, 99 lines of C named l1.c, tried to install itself and if successful acquired worm from the machine it came from and started it
- Worm tried to hide its existence, checked to see if it was already running on the machine and in 6/7 cases terminated if so. If not terminated, it tried to (1) break user passwords and (2) spread to other machines

Internet worm

- Primary effect of worm was to occupy most of machines' cycles (1/7 not enough to keep it from taking over). This is main reason why it was noticed.
- Spreading to other machines
  - (1) use broken passwords to get to other machines user has accounts on
  - (2) attempt to find "trusted hosts" and rsh to them

Internet worm

- Spreading to other machines (continued)
  - (3) exploit bug in finger program
    - finger / fingerd relationship
    - fingerd failed to check for buffer size
    - finger a specially crafted 536-byte long string
    - on fingerd side, this overflows allocated buffer, overwriting parts of data stack (e.g., return vector)
    - fingerd thus caused to returns control to this corrupted area code here attempts to create a shell (execute /bin/sh). If successful then successful penetration of system

Internet worm

- Spreading to other machines (continued)
  - (4) exploit system administration sloppiness
    - sendmail has a "debug" option that is supposed to be disabled if not disabled permits execution of commands
    - worm checked to see if destination hosts had debug disabled and took advantage if not

Threat monitoring

- Check for suspicious patterns of activity – i.e., several incorrect password attempts may signal password guessing.
- Audit log – records the time, user, and type of all accesses to an object; useful for recovery from a violation and developing better security measures.
- Scan the system periodically for security holes; done when the computer is relatively unused.

Threat monitoring

- Check for:
  - Short or easy-to-guess passwords
  - Unauthorized set-uid programs
  - Unauthorized programs in system directories
  - Unexpected long-running processes
  - Improper directory protections
  - Improper protections on system data files
  - Dangerous entries in the program search path (Trojan horse)
  - Changes to system programs; monitor checksum values

Threat monitoring

- Firewall: separates trusted and untrusted systems
  - limits network access between two security domains

Encryption

- Encrypt clear text into cipher text.
- Properties of good encryption technique:
  - Relatively simple for authorized users to encrypt and decrypt data.
  - Encryption scheme depends not on the secrecy of the algorithm but on a parameter of the algorithm called the encryption key.
  - Extremely difficult for an intruder to determine the encryption key.

Encryption

- *Data Encryption Standard* substitutes characters and rearranges their order on the basis of an encryption key provided to authorized users via a secure mechanism.
- Scheme only as secure as the mechanism.

Encryption

- Public-key encryption based on each user having two keys:
  - public key – published key used to encrypt data.
  - private key – key known only to individual user used to decrypt data.
- Must be an encryption scheme that can be made public without making it easy to figure out the decryption scheme.
  - Efficient algorithm for testing whether or not a number is prime.
  - No efficient algorithm is known for finding the prime factors of a number.

Encryption

- Properties of good encryption technique:
  - Relatively simple for authorized users to encrypt and decrypt data.
  - Encryption scheme depends not on the secrecy of the algorithm but on a parameter of the algorithm called the encryption key.
  - Extremely difficult for an intruder to determine the encryption key.

Encryption

- Must select an encryption scheme that can be made public without making it easy to figure out the decryption scheme.