Protection and Security

Protection: ensure controlled access to resources *internally* within a system

- OS provides mechanisms for policy enforcement
- Principle of least privilege: grant only enough privileges to complete task

Security: need to have adequate protection and consider *external* environment

- Security is hard because so many people try to break it.

Protection Domains

- Process should have access to specific objects and have the right to do specific things with each
- Rights should change to reflect what is needed at the time
- At any time, a process is operating in a protection domain that determines its access rights
- The domains should change to reflect actual needs
(principle of least privilege)
- In most systems
  - domain changes are rare
  - more rights are granted than are needed

Protection Domains: Examples

- When you compile, should the compiler have access to all of your files? E.g., mail, mp3 files, video
  - what is stopping it from transferring these files to another host
  - what is stopping it from deleting these files
- Often protection domain == user

  All processes belonging to a user have the same rights
- Changing protection domains

  UNIX setuid, effective user id becomes the same as file owner

  Windows Server "execute as"
- Grant additional rights to specific programs, but not a solution to the first problem above

Protection: The Access Control Matrix

		Objects				
		1	2	3	4	5
	1		R		R	
	2			R,W		R,W
Subjects	3	R,X	R,W	R,W		
	4			R		

objects: the things to be protected, e.g., files, printers, etc.

subjects: users, groups, roles

matrix entries: access rights, i.e., operations allowed by a subject on an object

A common implementation is an *access control list* for each object.

Protection: Access Control Administration

- there must be a mechanism for changing the access rights describe in the access control matrix
  - set of subjects is dynamic
  - set of objects is dynamic
  - access rights may need to change
- some approaches
  - encode access control change rights in the access control matrix
    * add “owner” as a possible access right. Subject with owner rights on object x can change access rights in x’s column.
  - new users/subjects can inherit rights from others

Protection: Example – Access Rights in Unix

- subjects are users and groups (group membership is maintained separately)
- each object has an owner and a group
- access rights are specified for the owner, for the group, and for everyone else
- object access rights can be modified by the object owner
- major access rights are read, write, and execute
- access controls can be applied to files, devices, shared memory segments, and more.

Protection: Authentication

- object access is performed by processes
- to apply access controls, it is necessary to associate processes with users
- this requires user *authentication*
- some authentication techniques:
  - passwords
  - cryptographic (e.g., public key methods)
  - physical tokens (e.g., smart cards)
  - biometrics

Security

OS / Network threats:

- Trojan Horses
- Trap Doors
- Buffer Overflows
- Worms
- Viruses
- Other (specific examples)

Even Worse:

- Physical
- Human

Trojan Horses

- Two different meanings:
  1. A programs that intends to run and do something undesirable
     E.g., download a program/game that also erases all of your files
  2. User tricked into running it
     Unix PATH variable includes "." as first entry
     `% cd /home/username; ls` BUT
     `/home/username/ls` is actually
     `cd $HOME; /bin/rm -rf *`

     Fundamental problem: privilege of command determined by user

Trap Doors / Back Doors

Lets perpetrator do things they wouldn't normally be allowed to do.
For example,

- when run by root
- surprise/undocumented response to a special input
  game cheats (type sequence of characters that make you invincible)
- special/alternate login program allows author access via special user name

Usually obscure, casual examination of source would miss it

Buffer Overflows

Send more data than is expected by executing program

Many programs have fixed-sized input buffers but don't limit input length

Data contains executable code

Consequences:

- Crash
- Much worse when buffer is on the stack (execute arbitrary code)

Very machine dependent but the X86 family is very wide-spread

Worms

- Program that replicates itself from one system to another (usually via Internet)
- Tend to consume resources leading to a Denial of Service (DOS) attack
- Arbitrarily bad behaviour
- Often use buffer overflows to compromise systems/users

  Morris Worm:
  - Spread using legitimate access of compromised users (e.g., .rhosts)
  - 1988 – 3 yrs probation, 400 hrs community service, $10K

  Sobig:
  - Mail to all users in the address book
  - Modifies system parameters to restart worm when rebooted (registry)

Viruses

- Not a free standing program, but a fragment attached to a legitimate program
- Essentially a dynamic Trojan horse or trap door.
- Especially a problem on single user systems with weak or non-existent protection

  Makes it easy to infect file systems
- Microsoft Office Macros and/or Active X lead to problems with Word files, email and web content
  - no need to reboot
  - execution not even expected
- Denial of Service often occurs at the same time (as a result of rapid spreading)

Specific Examples / Hacks

- Allocate virtual memory and look for data
- Illegal system calls or calls with invalid number of parameters or types
- Modify any OS data structures stored in user data e.g., `open()` call
- Look for don't do XXX in the documentation
- Social Engineering