Security

Protection mechanisms

At least three different categories:

• Data integrity: Backup etc., taken care by normal OS functions

- Protection against user errors: done by separating users and processes
- Protection against malicious users: more complicated, involves trade-offs between ease of use and level of protection

Major problem: *Identification of users* Most common scheme: Passwords

+ : Easy to use and understand

- : All too often too easily guessable

- : Exposure problem

- : Where to store passwords

Other problems arise from vicious programs. Most famous example so far : *Internet Worm* (1988)

Very clever use of weaknesses in OS design/program bugs

66

Access to shared resources must be controlled

Two aims:

- Protection against users' mistakes
- Increase in reliability

## Principles:

 separate policy (what) from mechanism (how)
Important for flexibility

 Users should have as much privilege as necessary to get job done

Standard way: each user separate domain of protection

Need *trusted way* of making system privileges available

Disadvantage: primary target for breakins (setuid-programs in UNIX)

67

## Access matrix

Implementation Issues

abstract view of protections have row for each domain and column for object

Entry indicates access right Example:

	$F_1$	$F_2$	$F_3$	Printer
$D_1$	read		write	
$D_2$				print

Have capabilities like read, write, copy, owner, control

Unsuitable for implementation because matrix far too large

- Access lists for objects (store columns)
- Capability lists for domains (store rows)

Capabilities allow great flexibility Example: Hydra

- Auxiliary rights: each process can pass access to procedure to other processes
  ⇒ dynamic change of access rights
- Rights amplifications: procedure can act on specified type on behalf of any process which is allowed to execute it Rights cannot be passed on
  ⇒ flexible and more secure mechanism for granting higher privileges temporarily

coarser than access lists

for each file, have three categories of possible users

- owner
- group (pre-defined set of users)
- all others

owner can grant permission to

- read
- write
- execute program/find files in directory

70

- Newer Alternative: PGP (Pretty Good Privacy)

Relies on so-called Public Key Cryptography:

Have two different keys, public key  $K_p$  and secret key  $K_s$  clear text encrypted via  $K_p$  can be decrypted only via  $K_s$  and vice versa

Assume every host in network has public and secret key

Now communication can take place by sending message from A to B as  $(m_{A_s})_{B_p}$ . Only B can decode the message, and it can verify that it came from A.

Certification of keys still necessary.

Two possible solutions:

- Central authentication server (Kerberos):

Aim: authentication in insecure networks

Have following protocol:

- Client → Auth Server: Credentials, please?
- Auth Server  $\rightarrow$  Client: ((Client Id, Session Key) $_S$ , Session Key) $_C$
- Client decrypts message and keeps Session Key
- Client → Server: (Client Id, Session Key)<sub>S</sub>
- Server decrypts message and obtains Client Id and Session Key

Authentication Server must be trusted Protocol vulnerable to replay attacks

71