1 SECURITY

Security is the set of mechanisms and techniques which prevent unauthorised access to resources or disclosure of information

- *Multi-access* systems must prevent unauthorised access of resources belonging to one user by another user
- Authorised *sharing* is ok
- *Security Policy* specifies *what* security is provided :
  what is protected
  who is allowed what access
  what protection mechanisms should be used
- *Security Mechanisms* : how is security policy implemented
  the same mechanism can however support different policies
- Security Includes : *Threat Analysis*, *Access control*, *Confidentiality and Encryption* and *Security management*

Will primarily look at *access control* and *protection mechanisms*

2 Access Control

- **Identification** : Identify the user - *who are you ?*
  General *login id* (name or number) supplied by user
  what other methods exist ?

- **Authentication** : Verifying identity of user - *Are you who you say you are* - Use of passwords (generally)
  Check if user supplied password matches that stored in system
  Check at login time / access to a resource (file)
  stored passwords MUST be protected : encryption
  Unix uses a *one-way* function for encoding passwords :
  x $\equiv$ F(x) (easy to compute)
  F(x) $\equiv$ x (difficult or impossible to compute)
  store F(x) NOT x (need to be protected)
  Must prevent *guessing passwords* - max number of tries (3 attempts at login)

- **Authorisation** : Specify who can access what - what operations are permissible (based on policy decisions)
  Default authorisation : *no access ?*, *all access ?*

- *Principle of Least Privilege* : Give user process the minimum access rights required to carry out assigned task (-need to know basis!)

3 Protection Domain

Protection domain is a set of access rights defined as a set of objects and operations permissible on them

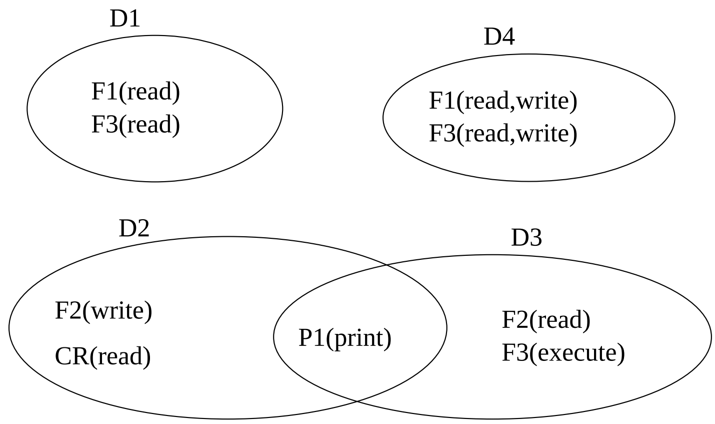

Figure 1: Access Domains

User process executing in domain D has access rights specified by D

Domain \ Object	F1	F2	F3	CR	P1
D1	read		read		
D2	write			read	print
D3		read	execute		print
D4	read write		read write		

Figure 2: Access Matrix

Specifies authorisation policy (Rows = domain, columns = objects)

4 Dynamic Protection

- Process may need to dynamically change access rights while executing (i.e. switch from one domain to another c.f. OS mode with access to privileged instructions) - Generally needed for OS procedures
- Use access matrix with domains as objects and enter rights to indicate domain switch can take place
- Another means of dynamically changing protection is to change access matrix itself : *Owner* : can add/remove any access right within the column applying to the owned object
  Copy : can copy rights marked with * within the column
  Transfer : can transfer rights marked with + within column
  Control : applies only to domains - A process in domain D2 can add or remove access rights within the row applying to D4

Object / Domain	F1	F2	F3	CR	P1	D1	D2	D3	
D1	read	owner	read*				enter		
D2	write+			read	print			enter	co
D3		read	execute		print				
D4	read write		read write			enter			

Figure 3: Access Matrix

5 Access Control Mechanisms

Mechanisms for controlling access to specific resources (like files or directories)

5.1 Global Table

- Global table of ordered triples
  (domain, object, permitted operations)
- User in domain **D** performs operation **O** on object **F** hence, search table for entry *(D,F,O)*
- If entry found, operation permitted otherwise raise exception
- *Problems* : Could lead to a large table
  Many entries if object accessible from many domains

5.2 Access Control List (ACL)

- Each column stored as an access list of
  (domain, operation) stored with each object
- In Unix, for instance, only 3 user domains (owner/user (u) , group (g) and others (o)) - with operations being read (r), write (w) and execute (x)
- A file can be set to switch userid to owner when it executes i.e. allows increase of privilege when using system programs
- Unix approach gives very *coarse* ACL
- Explicitly storing domains or individual users is more flexible

- Systems with more explicit ACLs store default domains plus individual entries (VAX/VMS)

System	r	w	-
Owner	r	w	e
Group	r	-	e
World	-	-	-
joe	r	-	-
tom	r	-	e

6 Capabilities

- A *row of the access matrix* can be *associated with a domain* to give a **capability list**
- A capability is a *protected pointer* to an object which specifies the permitted operations on the object
- Possession of capability gives *right to perform operations specified by it* (c.f. possession of a key)
- Capabilities are NOT directly accessed by users, but are *protected objects* - maintained by the OS and accessed *indirectly* by the user
- The OS provides procedures to *create, delete and modify* capabilities
- *Rights Amplification* : Trust worthy procedure can act on object passed as parameter of a specified type, if a calling process has capability to procedure
  Procedure rights *may be more than* rights of calling process

OBJECT TYPE	OPERATIONS	OBJECT POINTER

Figure 4: Capability

Hydra : Basic system objects are segments (with read, write and execute operations)
Application specific protection based on access procedures (operations) on user defined objects
Auxiliary rights : names of operations on a type
Capability give access to a specific object instance

6.1 Capabilities in Distributed Systems

A server or manager issues capabilities to access the objects it controls, such as file servers managing object files

OBJECT REFERENCE			OPERATIONS	CHECK FIELD
Server Address	Object Type	Object Instance		
64	16	16	32	64 bits

Figure 5: Capability in distributed systems

1. Check field encrypted version of permitted operations field and a random constant
2. Protects capability and prevents it being forged
3. When object created, a random number generated and stored in an internal reference table to the object
4. When an access request received, the object field used as an index into the table
5. Decrypting check field must yield the correct constant for the request to be permitted

- *Creation, copying or modifying* capabilities are *protected OS operations*
- *Operations field* indicates which of these operations are permitted on the capability itself
- Modifying *operations field* (usually to reduce access rights before passing on capability) performed by requesting the originator to issue a new one (by a special *commutative ONE WAY function*

- Capabilities permit VERY LIMITED access rights to be passed to other users
- Check capabilities can be more efficient than ACLs - so are used when checking is needed for each operation

7 Revocation of Access Rights

Are these to be performed

- Immediately or after a delay
- For all users or a selective group
- All rights or partial rights
- Temporary or permanent

Remove access rights for an object - given to a user/domain

Easy with global table or access list - *search list for object and remove entry*

Capabilities are distributed throughout the system - must be found and destroyed - *difficult*

- **Expiry Time** : capabilities expire after a time and new must be requested - *this is refused if rights have been revoked*
- **Back pointers** : objects maintain pointers to all capabilities issued - costly to implement, particularly if capabilities are passed around as parameters
- **Indirect Capabilities** : capability points to table entry which points to object - *Invalidate entry to revoke capability* - No selective revocation
- **Keys** : capability contains encrypted key checked by object - *change key in object to revoke capability* - No selective revocation

8 Security in Multics

Ring structured hierarchical domains (0-7)
The lower the ring number, the higher the privileges - with the operating system at ring 0 or D0 (domain 0)

- Segmented address space, with segment = file
- Segment has ring number (j) + R, W, X access bits
- Process has current ring number (i) to indicate domain in which it is executing
- Process can access any segment for which j $\geq$i, and the relevant access bit is set
- Process can perform a *controlled switch* to another domain by calling a procedure in the other domain
- The Segment descriptor includes :
  Access bracket : upper (b2) and lower (b1) limits from which call can take place to any procedure
  Gate List - list of entries which may be called
  Limit for calls to gates (b3)

A procedure in ring (i) can call procedure in segment (j) if :

1. (i) is less than (j) - *calling process transferring to less privileged ring.* May have to copy segments to area which can be accessed by less privileged procedure
2. (i) lies between b1 and b2 - *calling process within access bracket* - call any procedure in segment
3. (i) is greater than b2, but (i) $\leq$ b3 - call to a gate - *controlled entry* (to a more privileged OS procedure)
   This violates principle of least privilege.
   Process in ring (i) can access any segment with (j) greater than (i)

9 Security in WinNT

Aims : Meet requirements of the DoD *Orange Book* for the C-2 level of security "assurance" for commercial computer products.

The Structure

- Portable OS with minimal dependence on hardware features
- All computer resource - files, memory, processes (and special synchronisation primitives like semaphores) and ports are treated as objects - via *Object Manager*
- WinNT consists of an executive + several protected subsystems (servers) - executive atop HAL (hardware abstraction layer) and operates in *kernel* mode
- Security component consists of two parts : *WinLogon* and *protected server - Local Security Authority (LSA)* (both execute in user mode)
- LSA relies on the *Security Accounts Manager* and two executive components *Object Manager* and *Security reference monitor* to determine access privileges and obtain system resources
- Each user has a security ID and object security via access control lists (which users/groups are allowed access to what)

Authentication

Used to identify and log onto NT (via Ctrl-Alt-Del)

- Implements the *Orange Book* - Trusted path *requirement to assure user that when the Secure Attention Sequence is pressed, the resulting prompt for username and password is from OS and not from a spoofing program of a malicious user*
- Every user belongs to a particular *group* and a few special groups are built in.
- Each group has a name and a set of user rights (operations permitted)
- A user has the rights of all the groups he/she belongs to, plus any *special* rights granted to that user by the administrator
- Roughly two dozen user rights : *define printers*, *add hardware* (such as ethernet cards), *shut down*.

Auditing

User Manager controls auditing policy (i.e. keeping a record of who accesses the system and when (and possibly also from where)).

- Auditing can slow system - an overhead - use *User Manager* to specify level of auditing required
- Audited user actions fall into 7 categories - such as *file and object access*, *login and logouts*, *exercise of user rights* - where actions within each category can be audited for success or failure
- View audit records with the *Events Viewer* - can view *system events*, *application events* or *security events* - with each record containing a time-stamp

- Log also an object - and has an associated access control list - mainly for *Administrators group* - but also authorised user
- Possible to *force system shutdown* when log is full!!

Object Reuse

NT's logical objects translated to physical RAM and disk space - since these contain and recycle objects (swapping etc), it is necessary to ensure that a user does not access the remains of another user's work

- Erasing a DOS disk - DOES NOT really clear files contents (simply mark region as being unused) - FAT table analysis
- Try to prevent this with NT - when NT creates a new object - empty of data - clears previous object and assigns it zero length
- Prohibits from reading past a file's logical EOF marker
- NT clears RAM (when allocating memory) and prevents probing random locations (processes allowed to access only blocks of memory granted to them)

Network Security

Still a major topic for NT - may come to this later