



6G7Z1009: Introduction to Computer Forensics and Security

Access Control and Covert Channel



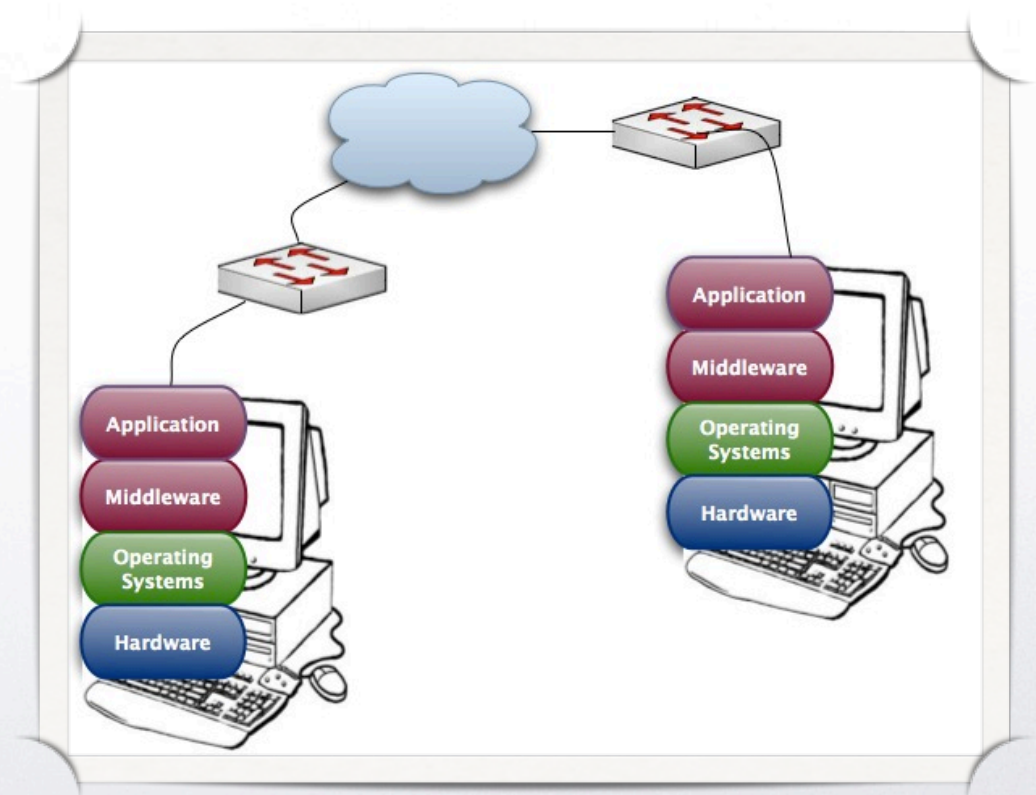
Outline of This Lecture

- What Is Access Control?
- Access Control Model
- Access Control Lists vs. Capability
- Unix Access Control



What is Access

- A traditional security mechanism
- Dealing with whether a user can have permission to access system resources and network resources, e.g., which programs the user can use, or share, etc.
- Access control can take place at different levels: File ACL, Network ACL





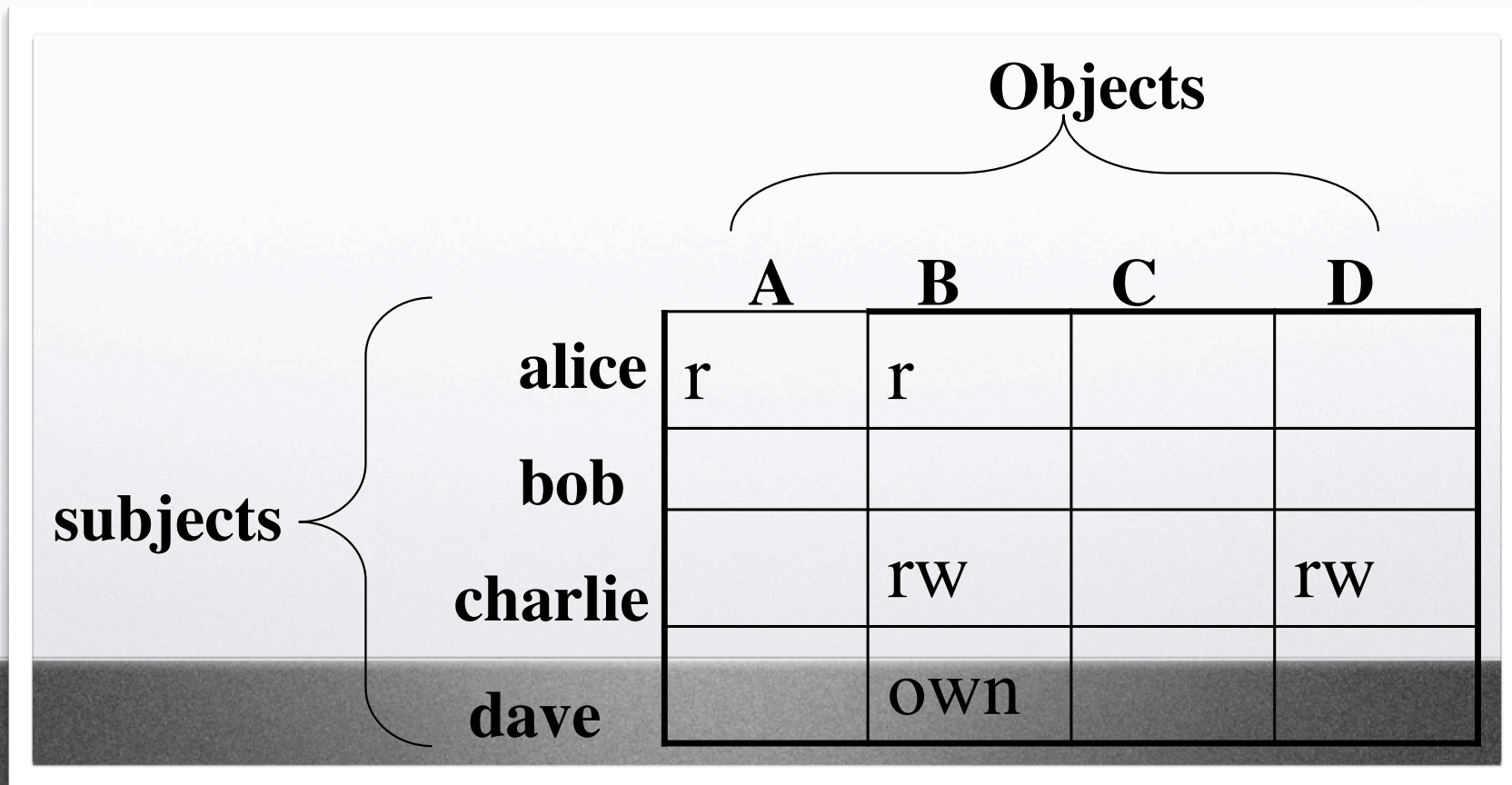
Access Control Abstract Model -I

- Basic abstraction: Subjects, Objects and Rights
 - Subjects: can be processes, modules, and roles
 - Objects: can be files, processes, etc.
 - Rights: means the access of the subject to the object



Access Control Abstract Model -II

- Access control matrix



The diagram illustrates an access control matrix. A bracket on the left groups the subjects (alice, bob, charlie, dave) under the label 'subjects'. A bracket on top groups the objects (A, B, C, D) under the label 'Objects'. The matrix is a 4x4 grid where each cell contains a specific access right or is empty.

		Objects			
		A	B	C	D
subjects	alice	r	r		
	bob				
	charlie		rw		rw
	dave		own		



Access Control Abstract Model -III

- **Users and principals**
 - Users: real world users
 - Principals: unit of access control and authorization
 - The system authenticates the user in context of a particular principal



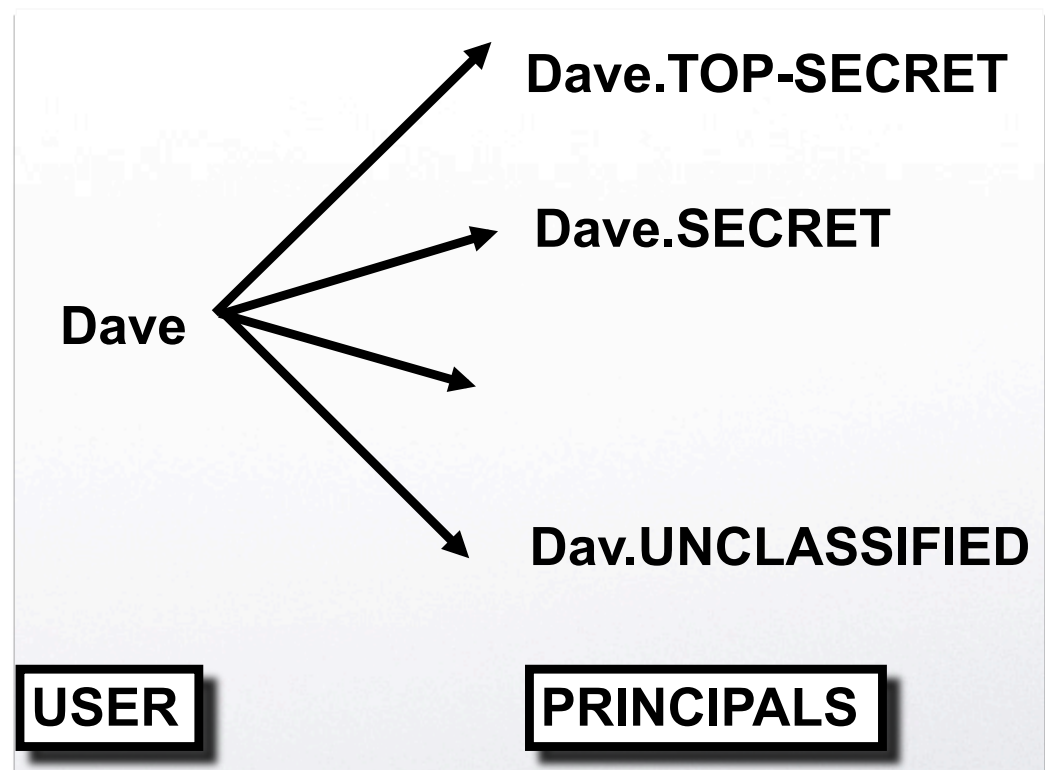
Access Control Abstract Model -IV

- **Users and principals: one to many relationship**
 - A user may have many principals
 - Each principal is associated with an unique user
 - The system authenticates the user in context of a particular principal



Access Control Abstract Model -V

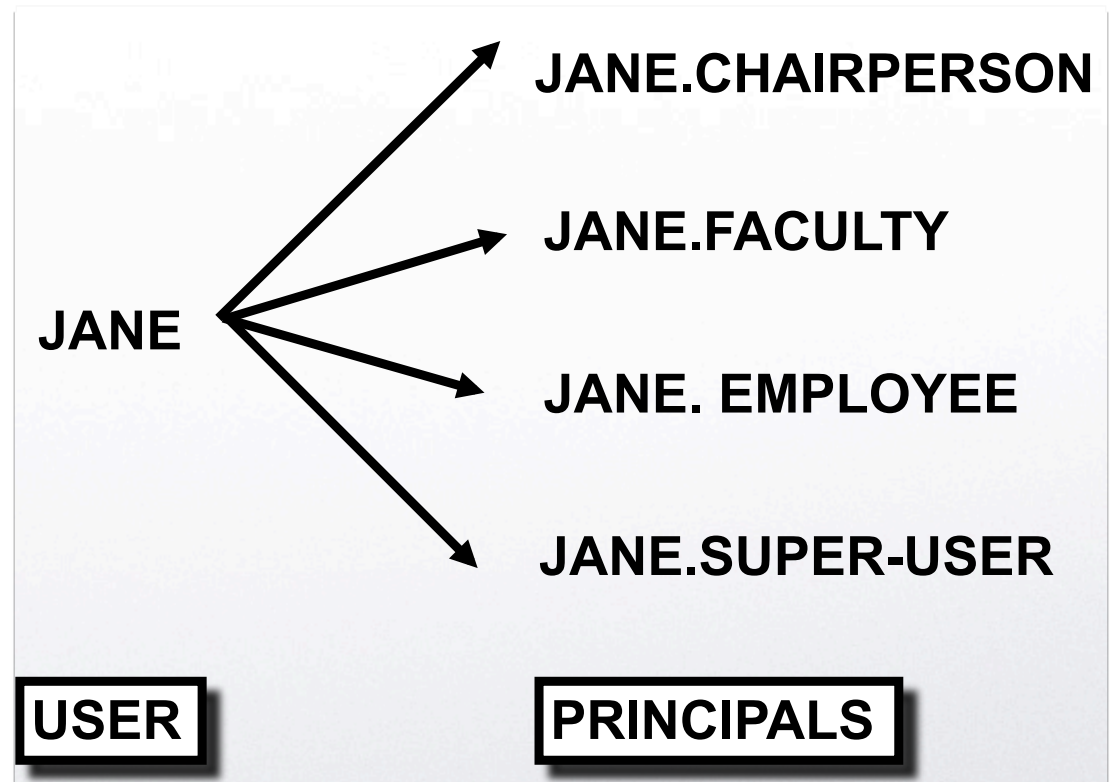
- Users and principals
 - Based on access control





Access Control Abstract Model -VI

- Users and principals
 - Based on authorisation

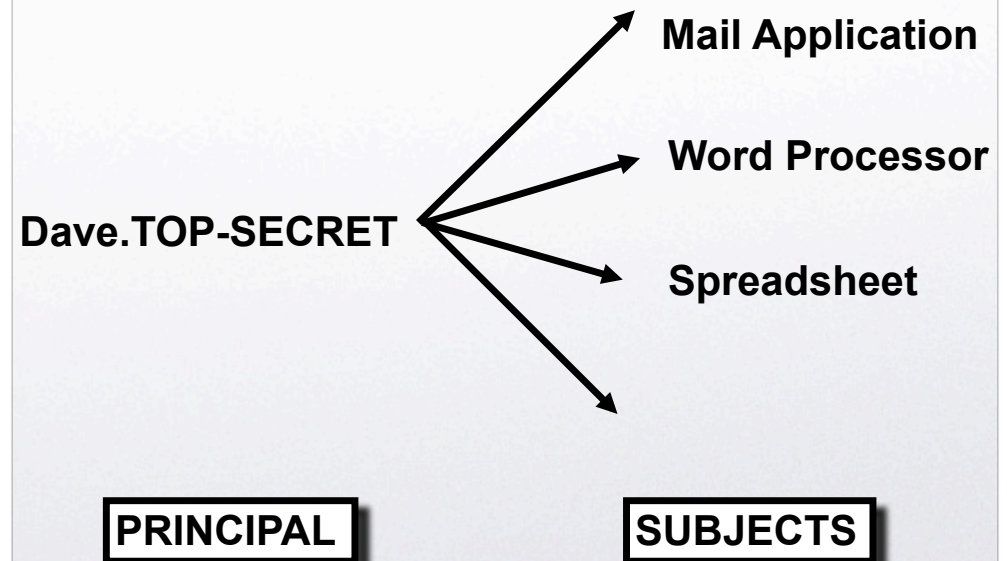




Access Control Abstract Model -VII

- **Principals and subjects**

- A subject: a program or application executing on behalf of a principal
- A principal may have one or more subjects executing on its behalf
- Each subject is associated with a unique principal
- All subjects of a principal have identical rights





Access Control Abstract Model -VIII

- **Object: a thing on which a subject can operates**
 - File Directory, directory, memory segment
 - Subjects can also be objects, with operations, such as kill, suspend, etc.



Access Control Abstract Model -IX

- Adding access rights
 - Read/write, change ownership

		Objects			
		A	B	C	D
subjects	alice	r	r/w	r	-
	bob	r	r	-	r/w
	charlie	-	-	w	-
	dave		r/w	-	w



Access Control Abstract Model -X

- **Grouping**

- Subjects: GROUPS, E.G., STAFF = {alice, dave}, students = {bob, charlie}
- Objects: Types e.g. system_file = {A, B}, user_file = {C, D}
- Compound names: e.g., in AFS tag: friends, system:backup



Access Control Lists vs. Capability -I

- ACL, a way to simplify the access right management by breaking down access control matrix by columns
- ACL stores the access control matrix a column at a time, along the resource to which the column refers, i.e., Access Control List.

User	Accounting Data
Sam	rw
Alice	rw
Bob	r



Access Control Lists vs. Capability - II

- Capabilities: the opposite of ACLs, another way to manage the access control matrix by breaking the matrix down by rows

User	Operating System	Accounts Program	Accounting Data	Audit Trail
Bob	rx	r	r	r

Source: Ross Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems



Access Control Lists vs. Capability - III

- ACLs require authentication of subjects. While capabilities do not but do require unforgeability and control of propagation of capabilities
- ACLs support superior access review on a per-object basis, while capabilities support the superior access review on a per-subject basis
- ACL support superior revocation facilities on a per-object basis, while capabilities support it on a per-subject basis
- Most OSs is based on the per-object basis
- Many OSs use abbreviation for ACLs using 'owner', 'group' and 'other'



Trojan Horses - I

- A trojan horse is a destructive software that is disguised as legitimate software. It performs a desirable as a user expects. In addition, it exploits the user's legitimate privileges to cause a security breach
- Trojan horse can be deliberately attached to otherwise useful software by a cracker, or can be spread by tricking users into believing that it is a useful program
- It can not replicate themselves not like viruses or worms



Trojan Horses - II

- Given principles A and B, we have file F1 and F2
- Principle A (PA) has read/write to F1 and can write F2
- Principal B (PB) has read permission to F2 but can not read/write file F1
- The Access Control List can be represented as follows:

ACL

PA:	R (Read)
PA:	

File F1

ACL

PB:	R (Read)
PA:	

File F2

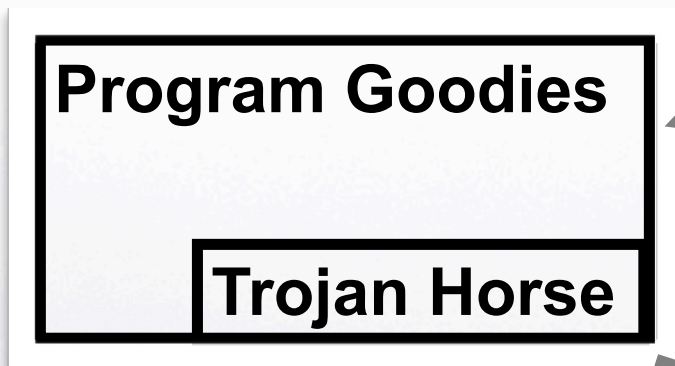


Trojan Horses -III

- Now, after Trojan horse is installed, Principal B can read contents of file F1 copied to file F2

PA:

ACL



Read

PA:

R (Read)

PA:

File F1

ACL

PB:

R (Read)

PA:

File F2

Write



Trojan Horses -IV

- Types of trojans:
- It may be used to erase or overwrite data
- It may spread other malicious software. The trojan horse in this case is called dropper.
- backdoor installation
- Phishing attacks



Covert Channel

- A communication channel based on the use of system resources not normally intended for communication between subjects /processes in the system
- It uses nonstandard methods to transfer information
- It allows multiple parties to communicate unseen. Communication is obscured
- unlike encryption, where the communication is obvious but obscure



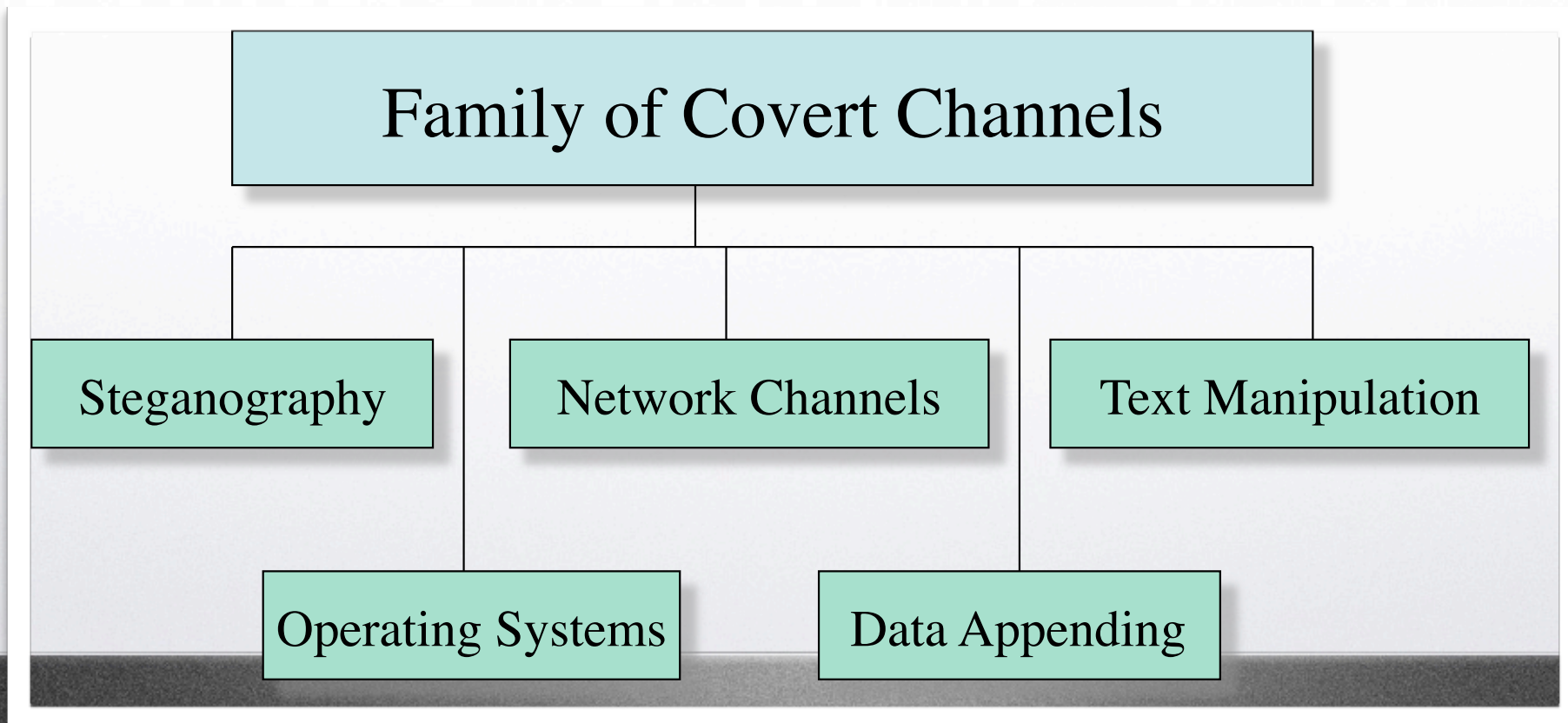
Covert Channels - I

- Types of covert channel
 - Steganography: images/audio
 - Network based: tcp/ip channels
 - Text Manipulation: word manipulation/substitution
 - Operating Systems: data hiding/alternate data streams
 - Data appending: EOF/Headers/Footers



Covert Channels -II

- Types of covert channel





Unix Access Control - I

- Unix design is based on the ideas of large, multiuser, timeshared systems. The basic goal is to keep users and programs from maliciously modifying other users' data or operating system data
- Unix includes files, network and other resources



Unix Access Control - II

- Unix file security:
 - Each file has owner and group,
 - permission set by owner
 - read, write, execute;
 - owner, group, other
 - represented by vector of four octal values
 - Only owner, root can change permissions: the privilege can not be delegated or shared
 - Setid bits



Unix Access Control - III

- Unix identification and authentication mechanisms

Users have a username

- Internally identified with a user ID (UID)
- Username to UID info in `/etc/passwd`
- Super UID = 0 (root)
 - can access any file
- Every user belongs to a group – has GID
 - also supplemental groups

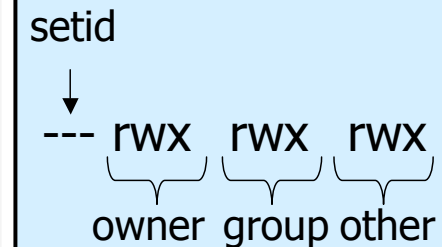
Passwords to authenticate

- in `/etc/passwd`
- shadow file `/etc/shadow`



Unix Access Control - IV

- Unix file permissions



File type, owner, group, others

```
drwx-----  2 usr1  cert  512  Aug 20  2003 risk management
lrwxrwxrwx   1 usr1  cert   15  Apr  7  09:11 risk_m->risk management
-rw-r--r--   1 usr1  cert 1754  Mar  8  18:11 words05.ps
-r-sr-xr-x   1 root  bin   9176  Apr  6  2002 /usr/bin/rs
-r-sr-sr-x   1 root  sys   2196  Apr  6  2002 /usr/bin/passwd
```

File type: regular **-**, directory **d**, symlink **l**, device **b/c**, socket **s**, fifo **f/p**

Permission: **r**, **w**, **x**, **s** (set.id), **t** (sticky)

While accessing files

- Process effective user ID (**EUID**) compared against the file's owner UID
- GIDs are compared, and then others are tested



Unix Access Control - V

- Unix user ID
 - Each process has three user IDs
 - real user ID (RUID): same as the user ID of parent (unless changed; it is used to determine which user started the process)
 - effective user ID (EUID): from set user ID bit on the file being executed, or sys call; it determines the permissions for process
 - saved user ID (SUID)
 - Real group ID, effective group ID, used similarly



Unix Access Control - VI

- Process operations and IDs
 - Root : ID=0 for superuser root, can access any file
 - Fork and Exec: inherit three IDs, except when executing a file with setuid bit on
 - Setuid system calls: can set EUID to real ID or saved ID
 - three setid bits: set EUID of process to ID of file owner
 - set EGID of process to GID of file
 - sticky: off (if user has write permission on directory, can rename or remove file); on (only file owner and root can rename or remove files in the directory)



Unix Access Control - VII

- How Unix execute a setid program

