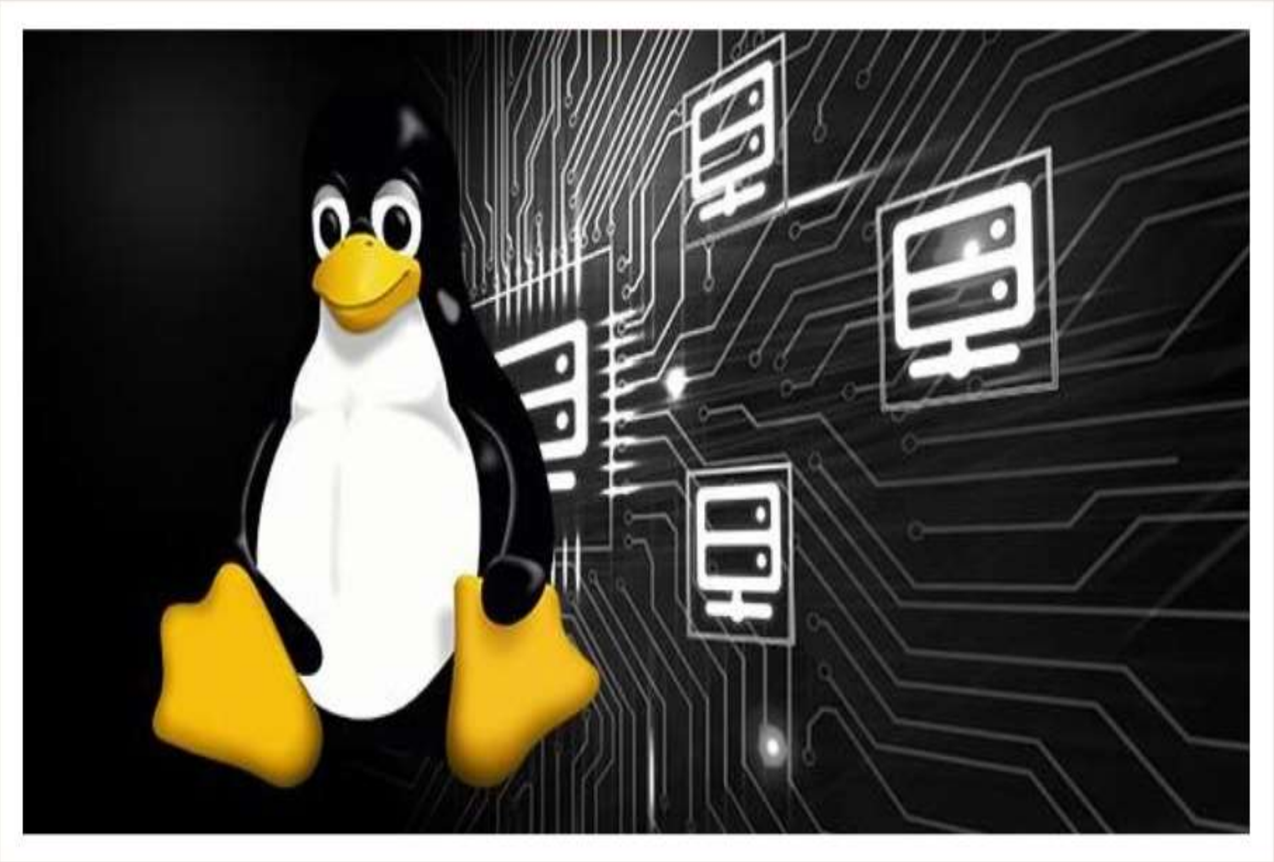


Operating systems

By

I Ravindra kumar, B.Tech, M.Tech,(Ph.D.)

Assistant professor,

Dept of CSE, VNR VJIET

Goals of Protection

Obviously to prevent malicious misuse of the system by users or programs

- To ensure that each shared resource is used only in accordance with system *policies*, which may be set either by system designers or by system administrators.
- To ensure that errant programs cause the minimal amount of damage possible.
- Note that protection systems only provide the *mechanisms* for enforcing policies and ensuring reliable systems.
- It is up to administrators and users to implement those mechanisms effectively.

Principles of Protection

The ***principle of least privilege*** dictates that programs, users, and systems be given just enough privileges to perform their tasks.

- This ensures that failures do the least amount of harm and allow the least of harm to be done.
- For example, if a program needs special privileges to perform a task, it is better to make it a SGID program with group ownership of "network" or "backup" or some other pseudo group, rather than SUID with root ownership.
  - This limits the amount of damage that can occur if something goes wrong.
- Typically each user is given their own account, and has only enough privilege to modify their own files.
- The root account should not be used for normal day to day activities
- The System Administrator should also have an ordinary account, and reserve use of the root account for only those tasks which need the root privileges

Domain of Protection

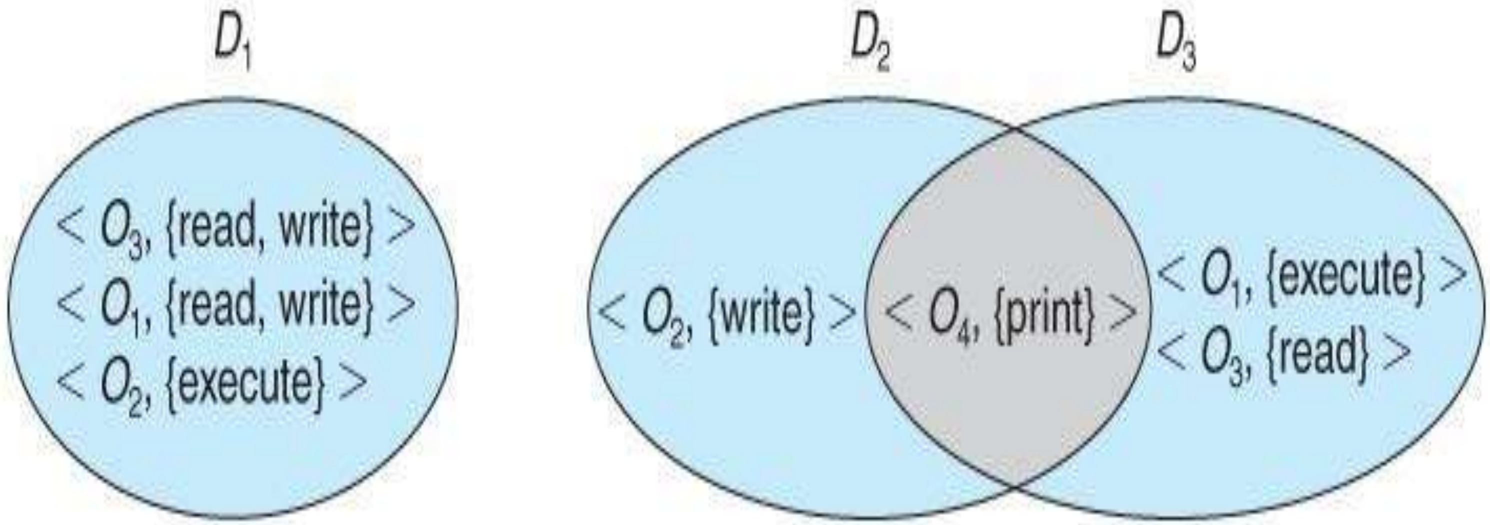
A computer can be viewed as a collection of *processes* and *objects* (both H/W & S/W).

- The ***need to know principle*** states that a process should only have access to those objects it needs to accomplish its task,
  - and furthermore only in the modes for which it needs access and only during the time frame when it needs access.
- The modes available for a particular object may depend upon its type.

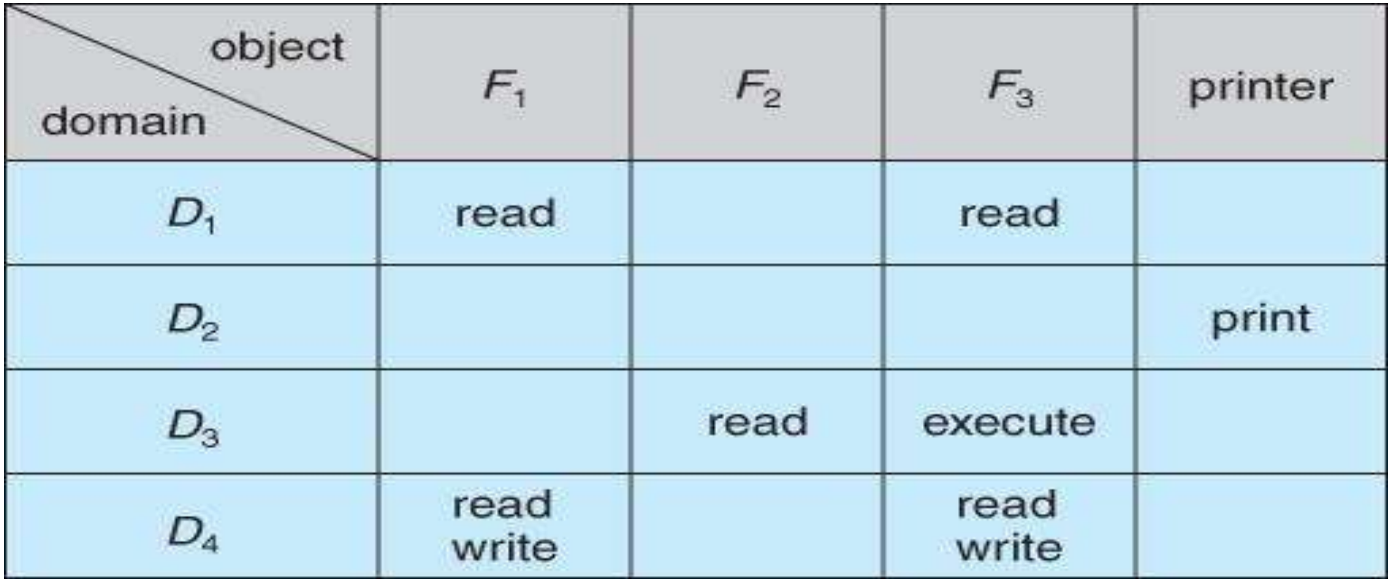
Domain Structure

A ***protection domain*** specifies the resources that a process may access.

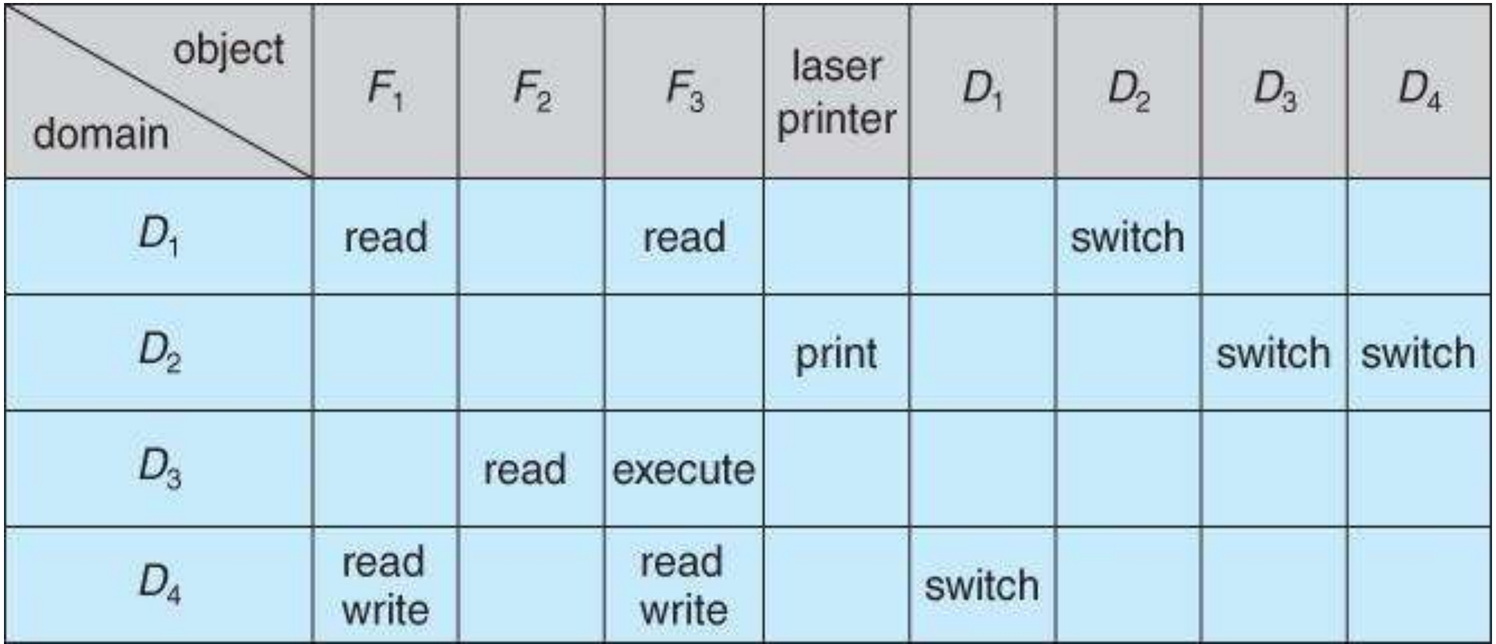
- Each domain defines a set of objects and the types of operations that may be invoked on each object.
- An ***access right*** is the ability to execute an operation on an object.
- A domain is defined as a set of < object, { access right set } > pairs, as shown below.
- Note that some domains may be disjoint while others overlap.



- **Access Matrix**

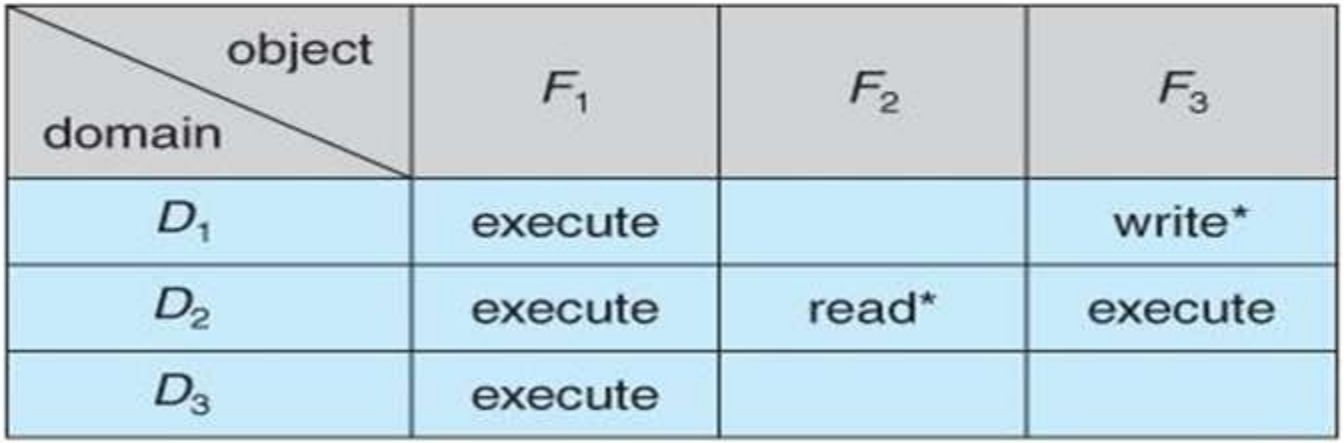


object domain	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

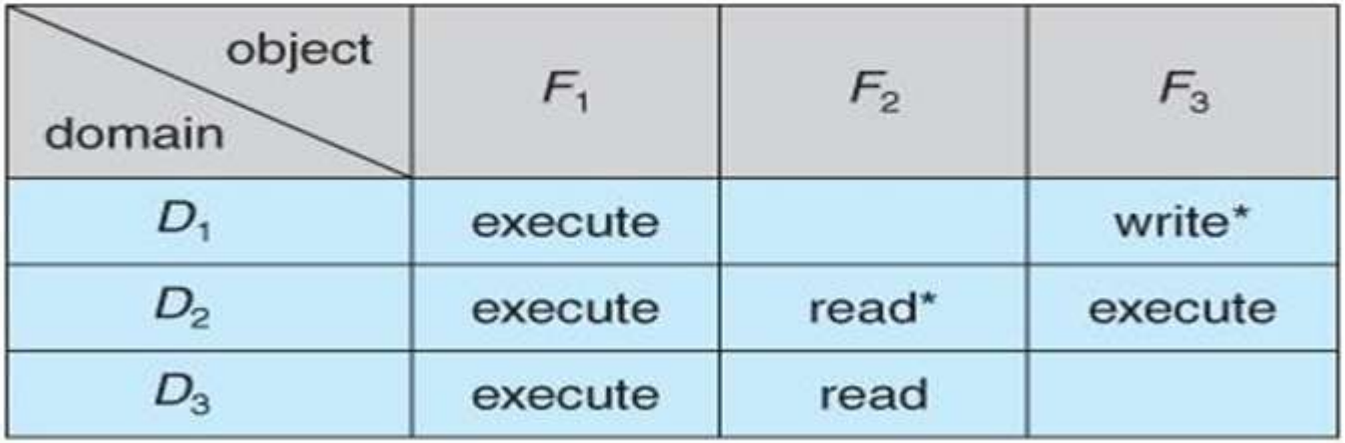


object domain	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

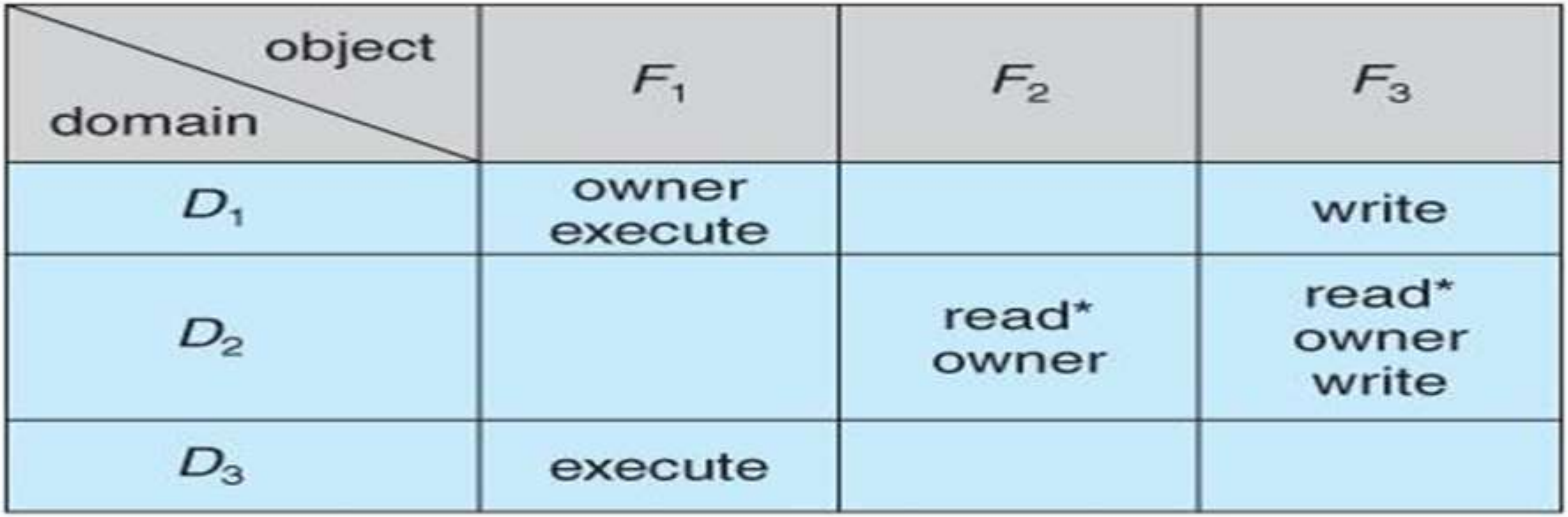
- The ability to ***copy*** rights is denoted by an asterisk, indicating that processes in that domain have the right to copy that access within the same column, i.e. for the same object.



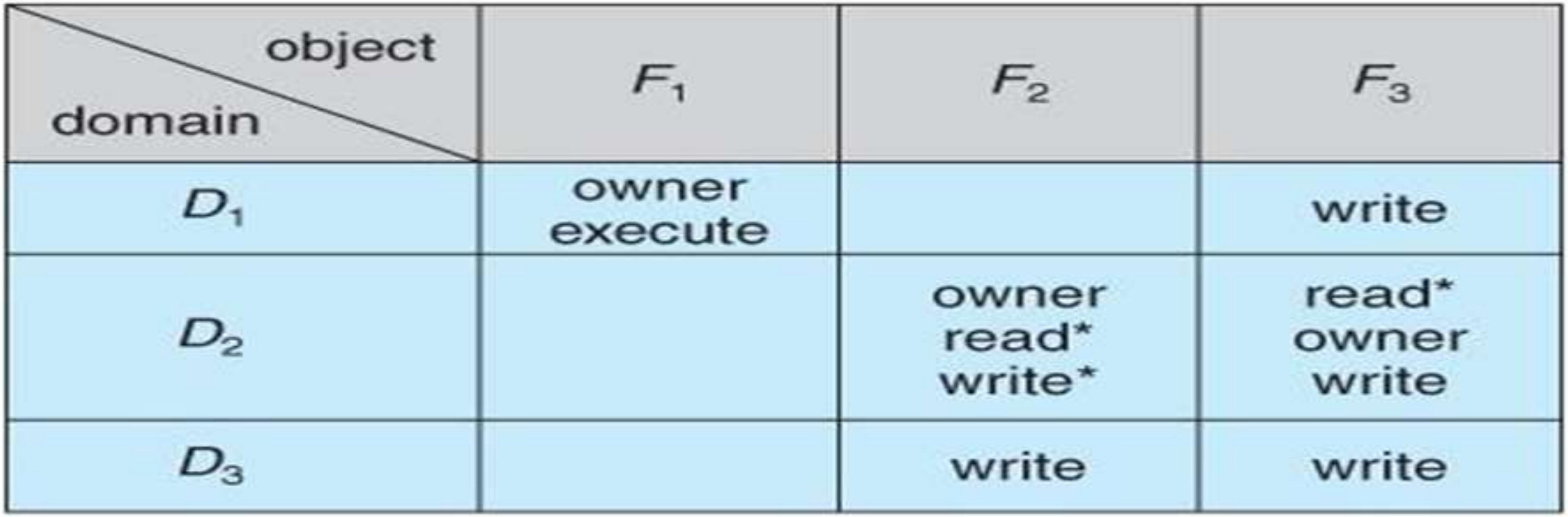
object domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute		



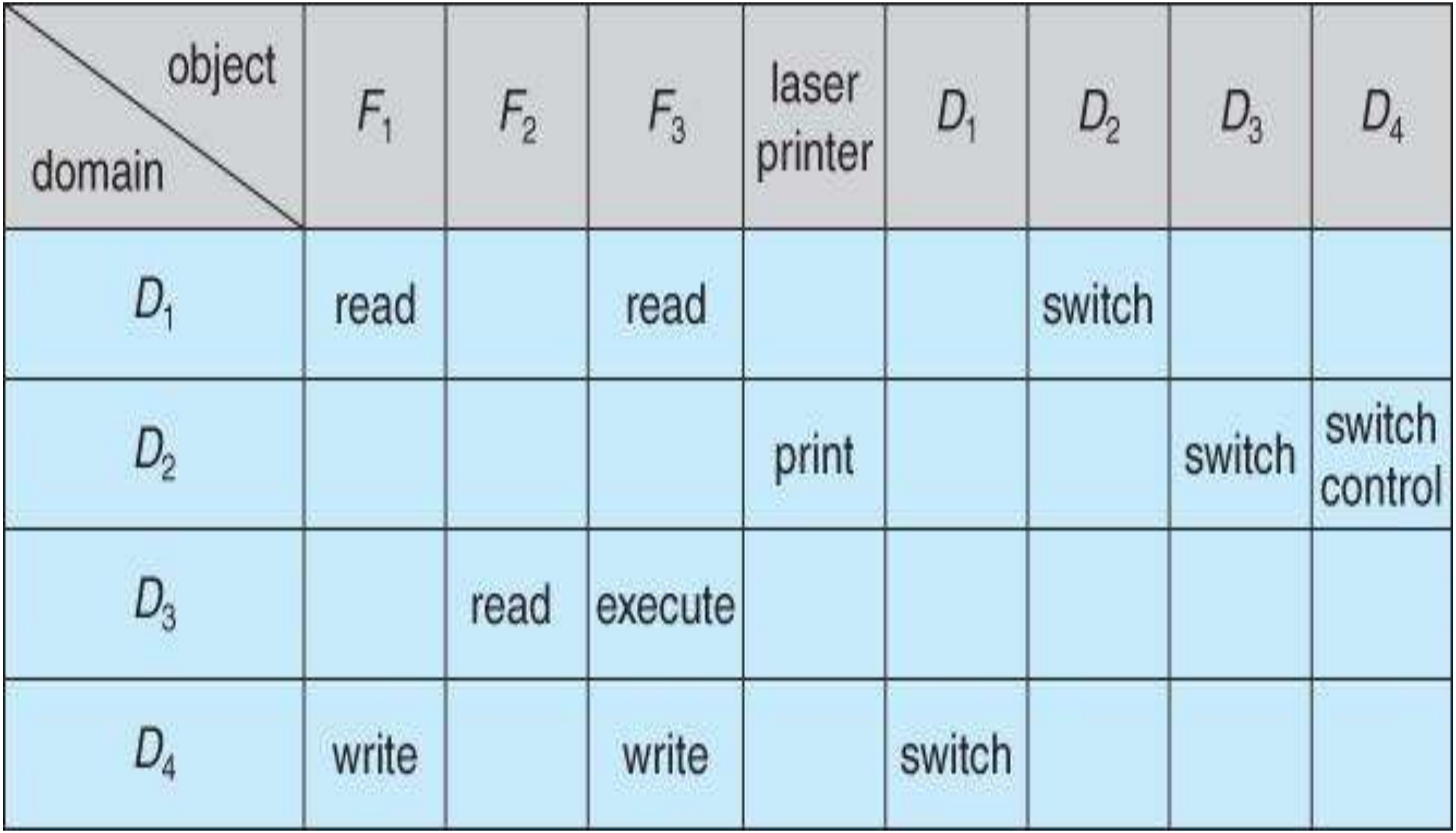
object domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute	read	



object domain	F_1	F_2	F_3
D_1	owner execute		write
D_2		read* owner	read* owner write
D_3	execute		



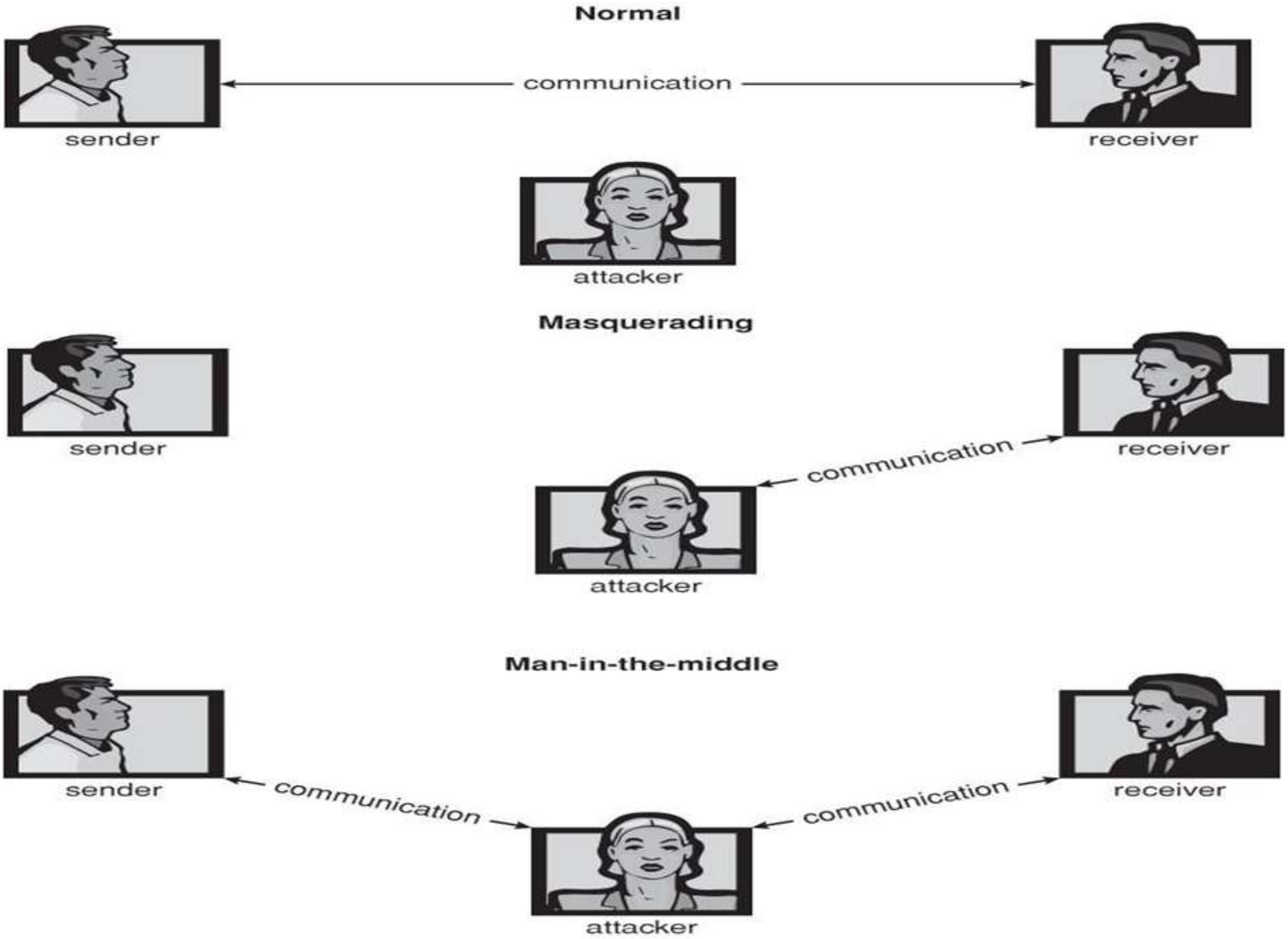
object domain	F_1	F_2	F_3
D_1	owner execute		write
D_2		owner read* write*	read* owner write
D_3		write	write



object domain	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch control
D_3		read	execute					
D_4	write		write		switch			

The Security Problem

- Some of the most common types of ***violations*** include:
- ***Breach of Confidentiality* -** Theft of private or confidential information, such as credit-card numbers, trade secrets, patents, secret formulas, manufacturing procedures, medical information, financial information, etc.
- ***Breach of Integrity* -** Unauthorized ***modification*** of data, which may have serious indirect consequences
- ***Breach of Availability* -** Unauthorized ***destruction*** of data, often just for the "fun" of causing havoc and for bragging rites.
  - Vandalism of web sites is a common form of this violation.
- ***Theft of Service* -** Unauthorized use of resources, such as theft of CPU cycles, installation of daemons running an unauthorized file server, or tapping into the target's telephone or networking services.
- ***Denial of Service, DOS* -** Preventing legitimate users from using the system, often by overloading and overwhelming the system with an excess of requests for service.


Normal
sender
communication
receiver
attacker
Masquerading
sender
receiver
communication
attacker
Man-in-the-middle
sender
receiver
communication
communication
attacker

- Four levels at which a system must be protected:
- **Physical -** The easiest way to steal data is to pocket the backup tapes.
  - Also, access to the root console will often give the user special privileges, such as rebooting the system as root from removable media
- **Human -** There is some concern that the humans who are allowed access to a system be trustworthy, and that they cannot be coerced into breaching security.
  - **Phishing** involves sending an innocent-looking e-mail or web site designed to fool people into revealing confidential information.
  - **Dumpster Diving** involves searching the trash or other locations for passwords that are written down.
  - **Password Cracking** involves divining users passwords, either by watching them type in their passwords, knowing something about them like their pet's names, or simply trying all words in common dictionaries.
- **Operating System -** The OS must protect itself from security breaches, such as runaway processes (denial of service), memory-access violations, stack overflow violations, the launching of programs with excessive privileges, and many others.
- **Network -** As network communications become ever more important and pervasive in modern computing environments, it becomes ever more important to protect this area of the system.

Program Threats

There are many common threats to modern systems.

A ***Trojan Horse*** is a program that secretly performs some maliciousness in addition to its visible actions

A ***Trap Door*** is when a designer or a programmer (or hacker) deliberately inserts a security hole that they can use later to access the system.

A ***Logic Bomb*** is code that is not designed to cause havoc all the time, but only when a certain set of circumstances occurs, such as when a particular date or time is reached or some other noticeable event

Stack and Buffer Overflow

- This is a classic method of attack, which exploits bugs in system code that allows buffers to overflow.

A **virus** is a fragment of code embedded in an otherwise legitimate program, designed to replicate itself (by infecting other programs), and (eventually) wreaking havoc.

System and Network Threats

The threats in this section attack the operating system or the network itself, or leverage those systems to launch their attacks.

A ***worm*** is a process that uses the fork / spawn process to make copies of itself in order to wreak havoc on a system. Worms consume system resources,

Port Scanning is technically not an attack, but rather a search for vulnerabilities to attack.

The basic idea is to systematically attempt to connect to every known (or common or possible) network port on some remote machine, and to attempt to make contact.

Denial of Service (DOS) attacks do not attempt to actually access or damage systems, but merely to clog them up so badly that they cannot be used for any useful work.

Tight loops that repeatedly request system services are an obvious form of this attack.