Security

Threat analysis
Aspects of security
Encryption
Authentication
Authorisation – access control

Security Goals

Prevent unauthorized access to system
or disclosure of information

Permit authorized sharing of resources

Data confidentiality

- ✧ Attack: theft of data

Data integrity

- ✧ Attack: destruction or alteration of data

System availability

- ✧ Attack: denial of service

Policy vs. Mechanism

Security policy specifies what security is provided:

- What is protected
- Who has access
- What access is permitted

Security mechanisms

- How to implement security policy
- Same mechanisms can support different policies

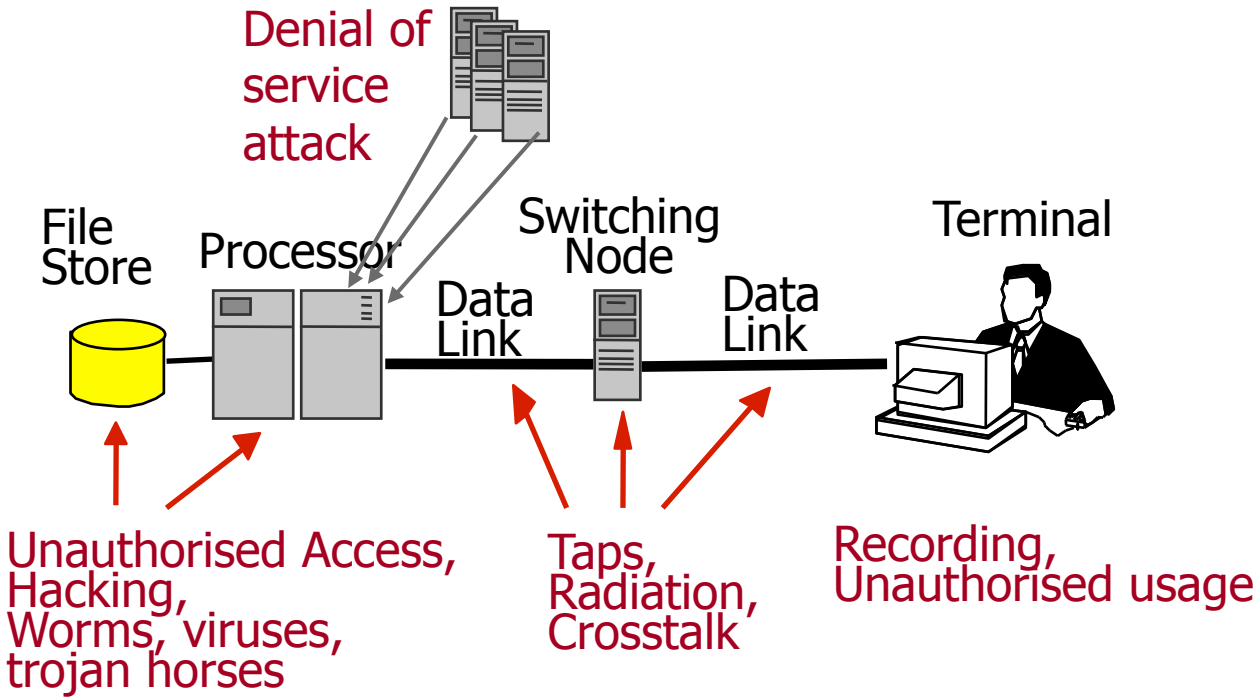
Security includes:

- Threat analysis
- Access control
- Confidentiality & encryption
- Security management

Threats: Identify potential attacks

Identify:

- What security breaches can occur
- Where they can occur
- How they occur



Threat Analysis 1

Information or Resources

- ✧ Theft / copying, disclosure
- ✧ Modification, corruption or fabrication
- ✧ Destruction

Services

- ✧ Unauthorised utilisation of resources
- ✧ Disruption of service
- ✧ Denial of access to authorised users

Users

- ✧ Abuse of privilege by legitimate user
- ✧ Masquerading – impersonation of the identity of another authorised user.

Physical security is insufficient for networked systems

Threat Analysis 2

Passive Attack

- ✧ Observe information in network without interference
- ✧ Message content – break confidentiality

Active Attack

- ✧ Modify or delete data
- ✧ Masquerade as authorised user
- ✧ Denial of service by flooding servers with valid requests
- ✧ Passwords gained through passive attack can be used for active attack

Threat Analysis 3

- Vulnerability Analysis: identify potential weak elements within system – What is critical to the organisation?
- Threat Assessment: likelihood of a threat occurring which exploits the vulnerability detected
- Risk Analysis: analyse the potential consequences of problems arising from security breach + estimate cost of a successful attack, e.g. loss of revenue.
- Prevention techniques: what can be done to prevent security breaches and what are their cost?
- Cost benefit analysis: do the consequences of security breaches justify the cost of protection.

  If security controls cause too much inconvenience or loss in performance they will be bypassed.
- Recovery: may be less costly than prevention?

Security Aspects

People security

- ✧ Insider, social engineering attacks e.g. phishing

Hardware security

- ✧ E.g., steal hard disk to get at data

Software security

- ✧ E.g., exploit bug to become superuser

Network based attacks

- ✧ Denial of service, malware from web etc
- ✧ Covered in Network & Distributed Systems course

System is as secure ***as it's weakest link!***

People Security

A high proportion of computer crime by insiders

- Employees need privileges to carry out duties
- Tempting to abuse privileges for own gain

Social engineering

- People often not security conscious: phishing attacks, people tailgating into building, etc.

People working around security measures for convenience

- E.g., reusing passwords, providing insecure way for resetting passwords, etc.
- Need balance between policies and convenience e.g. too complicated passwords will result in people writing them on postit notes.

People with wrong security expectations

- E.g. “one cannot forge a sender’s email address”

Hardware Security

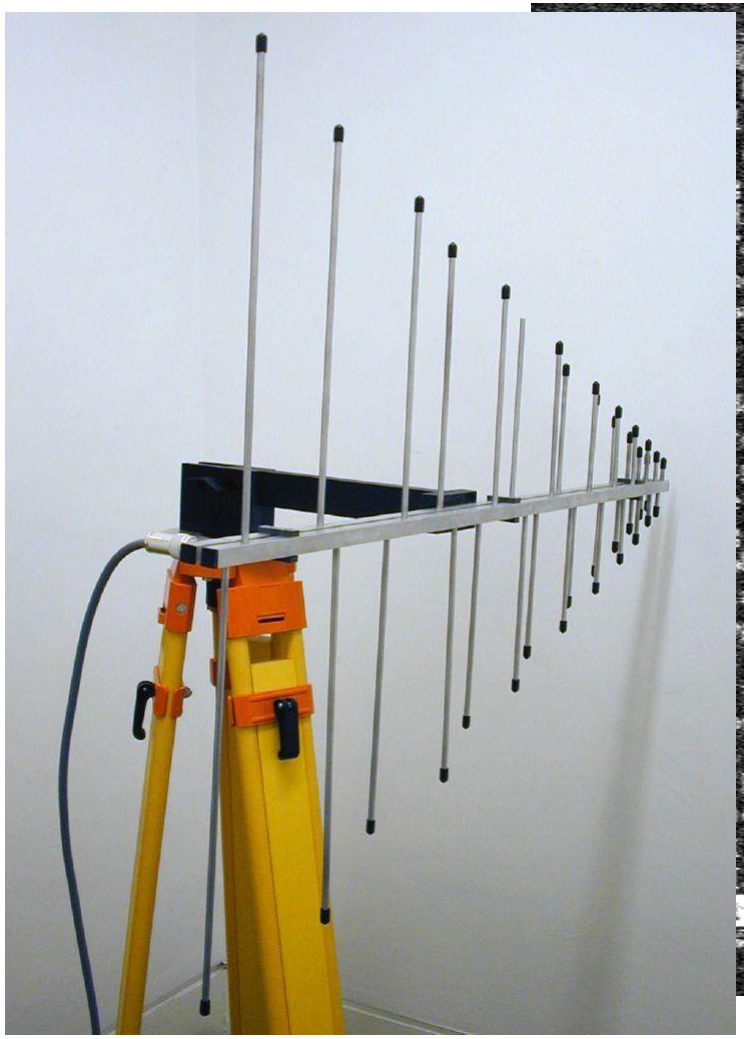
With physical access to computer/peripherals one can:

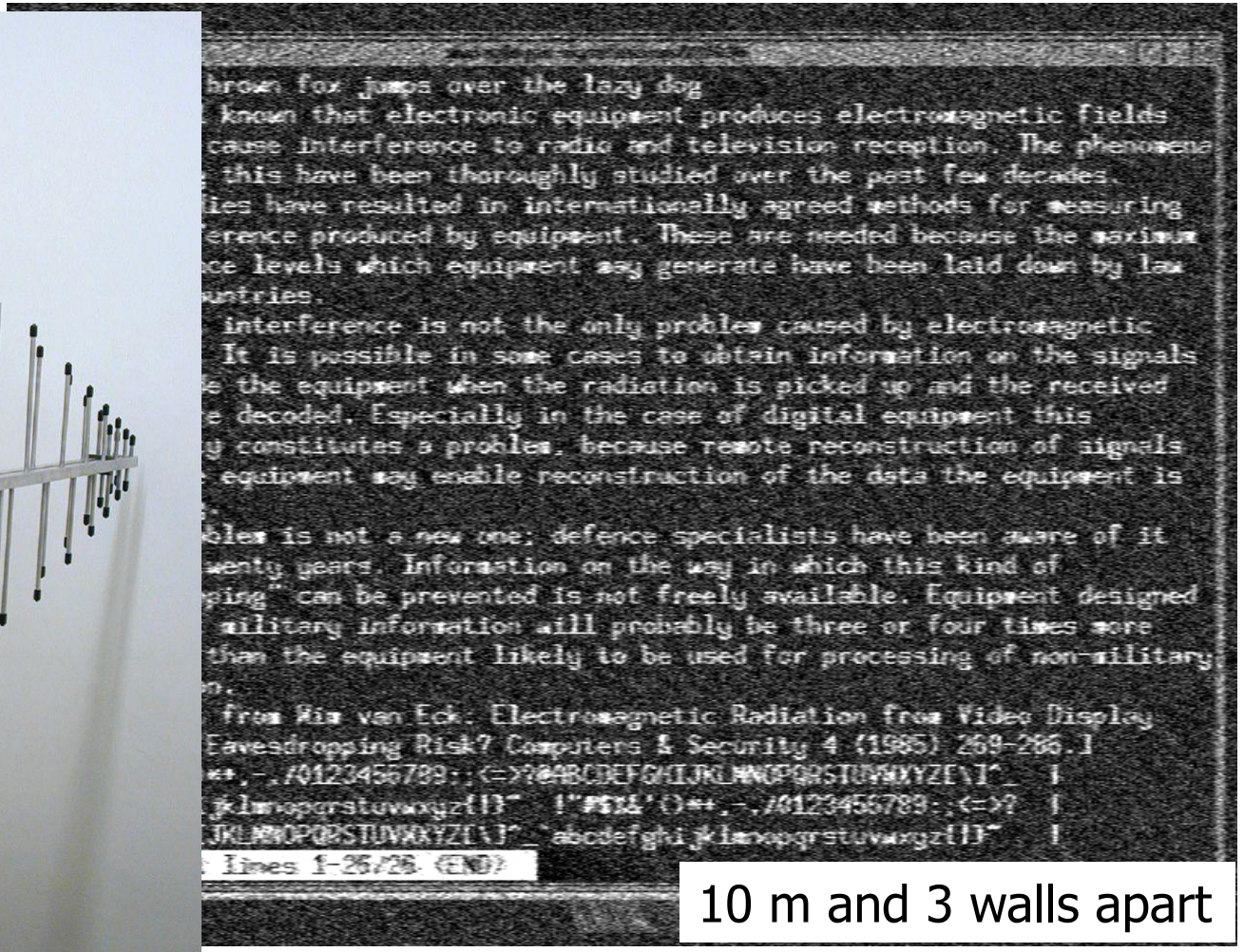
- ✧ Read contents of memory/disks
- ✧ Listen to network traffic including (unencrypted) passwords
- ✧ Alter contents of memory/disks
- ✧ Forge messages on network
- ✧ Steal or physically damage machines

Emission security:

- ✧ Computers give off electromagnetic waves
- ✧ Attacker can listen to emissions to tell what computer is doing
- ✧ Attacker can use strong emission source to destroy data
  - Military puts most sensitive computers in Faraday cages

Electromagnetic Eavesdropping [Kuhn 2004]





10 m and 3 walls apart

Software Security

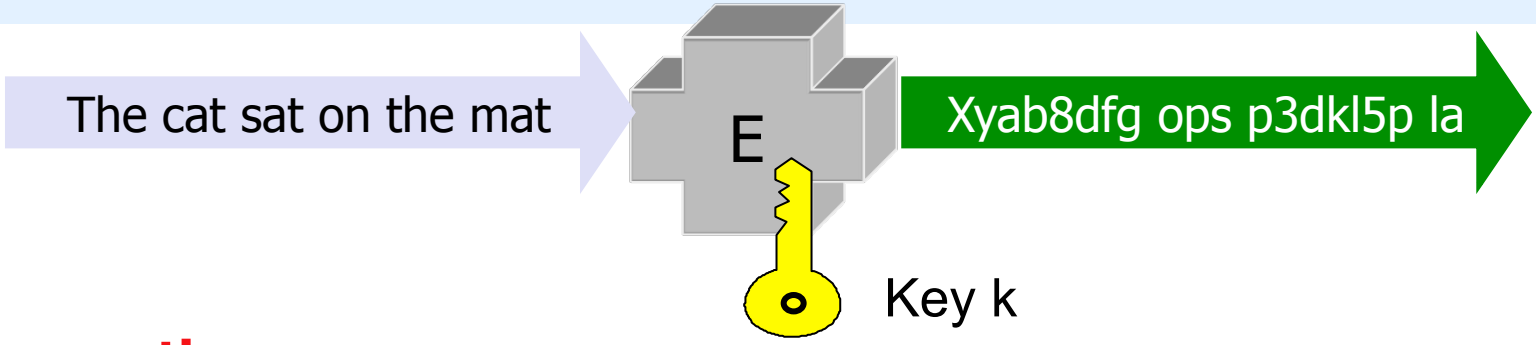
Software bugs may allow attackers to compromise system e.g. via web malware

- ✧ Gain root privileges
- ✧ Crash application
- ✧ Steal data
- ✧ Compromise data integrity

Attacks may exploit

- ✧ Buffer overflows
- ✧ Integer overflows
- ✧ Format string vulnerabilities

Cryptography mechanisms



Encryption:

Transformation of information based on:

Substitution e.g. table look up

Transposition e.g. exchange bytes 1 & 3, 2 & 4 etc.

Combine with a key which specifies what substitution or transposition to use

➔ Encryption *Function* *E* + *Key k*

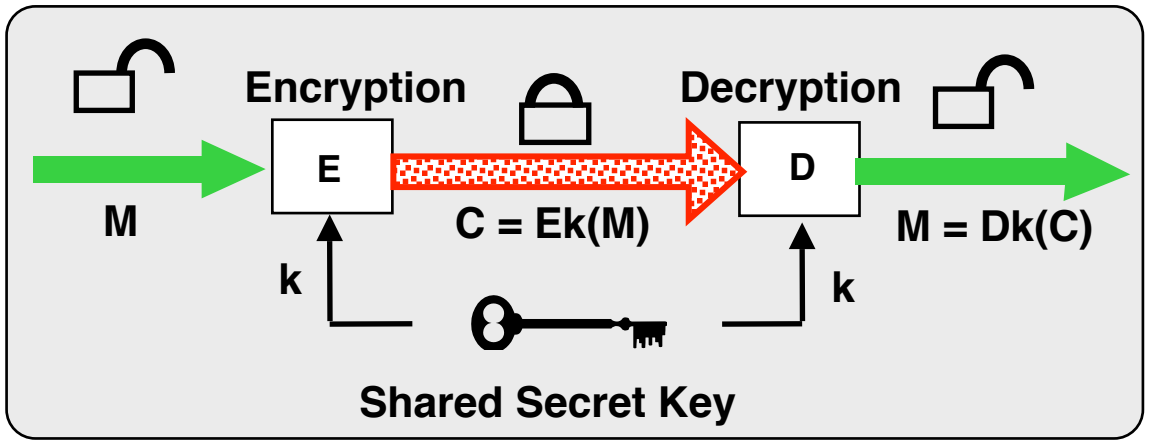
Decryption:

Inverse of encryption to obtain original information

Computation time required to decrypt without the key makes it impractical but not impossible.

Basic Security Mechanism

Single Secret Key



Secret Key Cryptography

- ✧ Basis for Data Encryption Standard (DES)
- ✧ Same algorithm applied for Encryption and Decryption
  i.e. E = D
- ✧ 56 bit key applied to blocks of 64 bits of data – easily cracked
- ✧ Advanced Encryption Standard (AES) selected in March 2001 after public competition: Rijndael from Belgium
  128, 192 or 256 bit keys

Access Control

Identification

- ✧ Establishing identity of a user
- ✧ Who are you?
  Generally login id: name or number supplied by user
- ✧ What other means could be used?

Authentication:

- ✧ Verify identity of users (**principals:** people & processes)
- ✧ Are you who you say you are?

Authorization:

- ✧ Allow principals to perform action only when authorized
- ✧ Specify who can access, what they access and what operations are permitted ➔ policy decisions

Authentication

Verification of identity of principal based on:

- ✧Personal characteristics
- ✧Knowledge
- ✧Possessions
- ✧Signed certificate

Authentication: Personal Characteristics

Authentication based on hard to forge, personal characteristics:

- ✧ Fingerprints
- ✧ Voiceprints
- ✧ Retina patterns
- ✧ Signature analysis
- ✧ Signature motion analysis
- ✧ Typing rhythm analysis

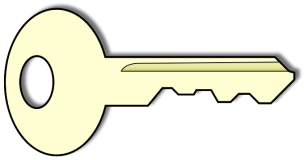
Can suffer from:

- ✧ High equipment cost
- ✧ False positives / negatives

Authentication: Possessions

Authentication based on securely-kept possessions

Possession of keys most widely used system



- Can ensure physical security of computers and other things
- Keys – give authentication without identification
- Being superseded by coded magnetic cards, RFID cards, implanted sensors etc.
  Combine identification with authentication

Can suffer from:

- Impersonation attacks if key lost
- High equipment costs

Authentication: Knowledge

Authentication based on secret knowledge (***password***):

- ✧ Very cheap to implement

Limitations:

- ✧ ***Dictionary attacks*** can find most passwords:
  - Good guesses include login name, first names, street names, dictionary words, any of these reversed or doubled
- ✧ ***Password reuse***
  - Users tend to reuse passwords
  - *Security as good as security of weakest system*

Limitations of Passwords

Password reuse:

- Users tend to reuse passwords
- *Security as good as security of weakest system*

Password turnover:

- Password vulnerable to guessing attacks throughout lifetime
- Well-chosen password (with good encryption algorithm) can only be cracked by exhaustive search

Large number of systems we use requiring passwords

- Difficult to remember so many different passwords

Change password regularly (every *n* weeks/months)

- Crackers have to begin search anew
- But people get lazy: **`mypasswd1`**, **`mypasswd2`**, …

Password Protection: One-Way Cryptographic Hash

Some OSs used to store user passwords unencrypted in protected file

- ✧ Vulnerable to data theft, accidental disclosure/abuse by system administrators

Modern OS store only *encrypted* versions of passwords

- ✧ Use one-way cryptographic hash function for encryption
- ✧ Compare encrypted version of the string entered by user A with the encrypted password stored for A
- ✧ Given hash h, it should be infeasible to find M such that H(M) = h

Password Encryption

Encryption based on one-way hash functions

- UNIX's based on DES + use of Salt
- Salt s: random value, often based on time
- Triple **`(userid, s, e(P|s))`** stored in password file
- At login, **`e(P|s)`** re-computed and compared with stored value

Bobbie, 4238, e(Dog4238)
Tony, 2918, e(6%%TaeFF2918)
Laura, 6902, e(Shakespeare6902)
Mark, 1694, e(XaB@Bwcz1694)
Deborah, 1092, e(LordByron,1092)

UserID Salt Password

Guessing only feasible way to find cleartext password from encrypted password

- Choose inherently slow encryption function to limit number of guesses

Salt in Passwords

Password file only readable by Superuser. File not encrypted, but passwords are

If password file is accessed, can use dictionary attack

eg e(dog) and search for all users with password dog

Use of salt prevents

- ✧ Duplicate passwords from having same encrypted value
- ✧ Need to use e(s l dictionary word) for each user
  - ➔ more time consuming
- ✧ Cannot reuse stored encrypted dictionary words

Adobe – Leaked passwords

Nov 2013: 130,324,429 leaked passwords, no salt, hints not encrypted

#	Count	Ciphertext	Plaintext
1.	1,911,938	EQ7fIpT7i/Q=	123456
2.	44,6162	j9p+HwtWWT86aMjgZFLzYg==	123456789
3.	34,5834	L8qbAD3jl3jioxG6CatHBw==	password
4.	21,1659	BB4e6X+b2xLioxG6CatHBw==	adobe123
5.	20,1580	j9p+HwtWWT/ioxG6CatHBw==	12345678
6.	130,832	5djv7ZCI2ws=	qwerty
7.	124,253	dQi0asWPYvQ=	1234567
8.	113,884	7LqYzKVeq8I=	111111
9.	83,411	PMDTbP0LZxu03SwrFUvYGA==	photoshop
10.	82,694	e6MPXQ5G6a8=	123123

Authorisation

Specifies:

- *who* can access
- *what* they can access
- *how* they access can (what operations)

Policy decision: what should be the default authorisation?

- no access?
- all access?

Principle of Least Privilege

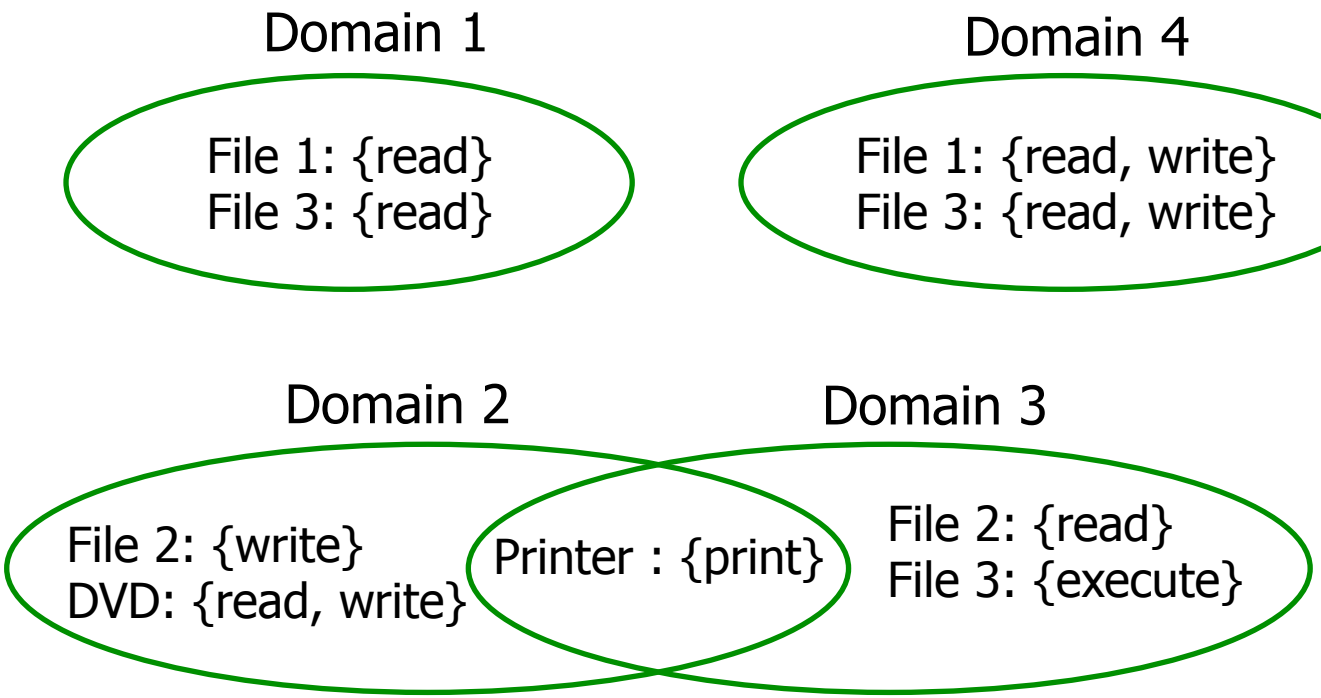
- Gives user minimum access rights required to carry out assigned task

Protection Domains

Set of **access rights** defined as:

- ✧ Set of **objects**
- ✧ **Operations** permitted on them

Principal executing in **domain** D has access rights specified by D



Access Control Matrix

Specifies **authorisation policy**

- ✧ Rows represent principals
  - e.g. users, user groups, …
- ✧ Columns represent target objects
  - e.g. files, devices, processes, …

	Object 1	Object 2	Object 3	Object 4	Object 5
Principal 1	read		read		read
Principal 2		execute		read, print	
Principal 3	read	read, print		execute	read
Principal 4	read, write		read, write		

Access Control Matrix: Implementation

Expensive to implement matrix as global 2D array

Two options:

- ✧ Access-Control Lists (ACLs)
  - Associate with target object,
  - Indicate who, and what access
- ✧ Capabilities
  - Associate with principal
  - Indicate what target and what access

Mildly controversial topic

- ✧ Both options have pros and cons
- ✧ In practice, most operating systems implement ACLs

Access Control List

Each column of access matrix stored as **access control list** (ACL)

An ACL is ***stored with each object***:

- The principals that can access it
- The operations each principal can perform on it

Example: UNIX/Linux

Only three domains for each file:

user (u)	Owner of the file
group (g)	Group of the file
other (o)	Rest of the world

- ✧ Each user can belong to multiple groups
- ✧ Each file can only belong to one group
- ✧ Superuser **root** can access any file as owner
  - only superuser can change ownership of files

Operations: **read (R)**, **write (W)**, **execute (X)**

- ✧ For directories, these operations mean:

Read	Can list contents of directory
Write	Create/delete (owned) files
Execute	Enter directory, get access to files

Unix Example File Protection

Binary	Symbolic	Allowed file accesses
111000000	rwx------	Owner can read, write, and execute
111111000	rwxrwx---	Owner and group can read, write, and execute
110100000	rw-r-----	Owner can read and write; group can read
110100100	rw-r--r--	Owner can read and write; all others can read
111101101	rwxr-xr-x	Owner can do everything, rest can read and execute
000000000	---------	Nobody has any access
000000111	------rwx	Only outsiders have access (strange, but legal)

Each row corresponds to a file, directory or device.

SUID (set user id) bit

- ✧ File switches user ID to file owner when executed
- ✧ increases privileges when using system programs:
  e.g. For changing passwords
- ✧ *Violates principal of least privilege*.

Capabilities

A row of an access matrix is stored as a list of capabilities

A capability list is associated with each principal (process)

Each capability indicates target object and permitted operations on that object.

Capabilities must be protected from user tampering:

1. Stored in Kernel space
2. Tagged memory: a bit set on each memory indicates whether it is a capability and then only allows kernel access
3. Cryptographically protect capability.
   The capability contains a field created by the kernel by encryption of the object ID, the rights and a check number it holds. The user does not know this number so cannot create or modify the capability.

Server	Object	Rights	E (Object, Rights, Check)

Capabilities 2

- Capability is a protected pointer to object specifying permitted operations on object
  E.g., file descriptor is similar to a capability
  - Possession of capability gives right to perform operations specified by it
    - Similar to possession of key or ticket
- Kernel provides procedures to create, delete, modify capabilities
- User can request Kernel to create capability with limited access rights to object eg read only access for file passed to print spooler.
- Kernel checks encrypted field when access takes place.

ACLs vs. Capabilities

Principle of least privilege: limited access capabilities giving only required rights can be created and passed to other processes. ACL based systems usually need system processes with full rights to everything.

Revocation: Capabilities easily revoked for a user but difficult to track down capabilities passed to others or to revoke rights to a specific object, which may be stored by multiple principals. Access control entry related to a specific object can be easily revoked but all rights for a specific principal is harder, as multiple ACLs in the system may have access rights set.

Persistence: Capabilities may point to objects that have been deleted.

DAC vs. MAC

Discretionary Access Control (DAC):

- ✧ Principals determine who may access their objects
- ✧ Generally used in commercial systems

Mandatory Access Control (MAC):

- ✧ Precise system rules that determine access to objects
- ✧ Can be very inflexible
- ✧ Used in military systems
- ✧ Emphasis on information flow

Bell-La Padula Security Model

Security level assigned to objects and principals

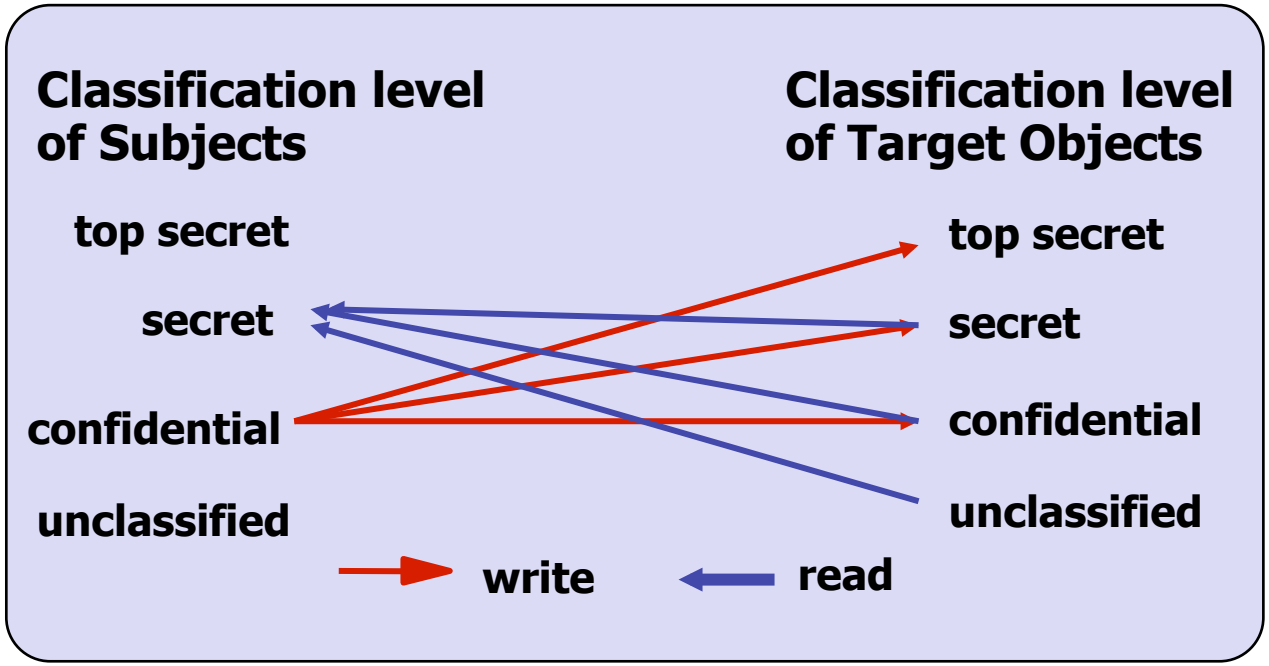
- ✧ E.g., unclassified, confidential, top secret

Categories of information also assigned

- ✧ E.g. nuclear, naval, airforce, drones etc.

Subject Label (clearance) =
(Level {categories}) e.g. secret {airforce, drones}

Target Label = (level {1 category})



Write up
Read down

Bell-Lapadula Model 2

1) Simple Security condition

Subject s may only read from target t

- if (s.level ≥ t.level) & (t.category ∈ {s. categories})
- i.e. read information of permitted category at same or lower classification eg officer cleared for secret cannot read top secret, but can read all lower classified documents

2) * Property

Subject s can write to a target t

- if (s.level ≤ t.level of t) & (t.category ∈ {s. categories})
- ie. write to documents at higher classification
  ➔ prevents copying information from higher to lower levels
- Can only have read/write access to documents at same level

Anomaly ?
Integrity?

Biba Model

Guarantees data integrity:

- **The simple integrity principle:** A process running at security level k can read only objects at its level or higher (no read down)
- **The integrity * property:** A process running at security level k can write only objects at its level or lower (no write up)
- **Invocation property:** a user cannot request a service (invoke) from a higher integrity level

- Note Biba (write up, read down) is opposite of BLP (read up, write down)

Design Principles for Security

System design should be public

- *"Security through obscurity" is usually a bad idea*

Default should be ***no access***

Give each process ***least privilege possible***

Protection mechanism should be simple and uniform

Security policies should not just be imposed without consultation and must not be inconvenient for users.

Keep it simple!

Computer Security: Summary

Security goals:

- ✧ Prevent unauthorized access to system
- ✧ Permit authorized sharing of resources

Need to consider many aspects

- – People, hardware, network access & software

Authentication: identification + verification of identity

- ✧ Personal characteristics, possessions, passwords
- ✧ Passwords have limitations but are still widely used

Access control: what & how

- ✧ ACLs most widely used
- ✧ Some systems use capabilities
- ✧ Bell-lapdula multi-level security used by military