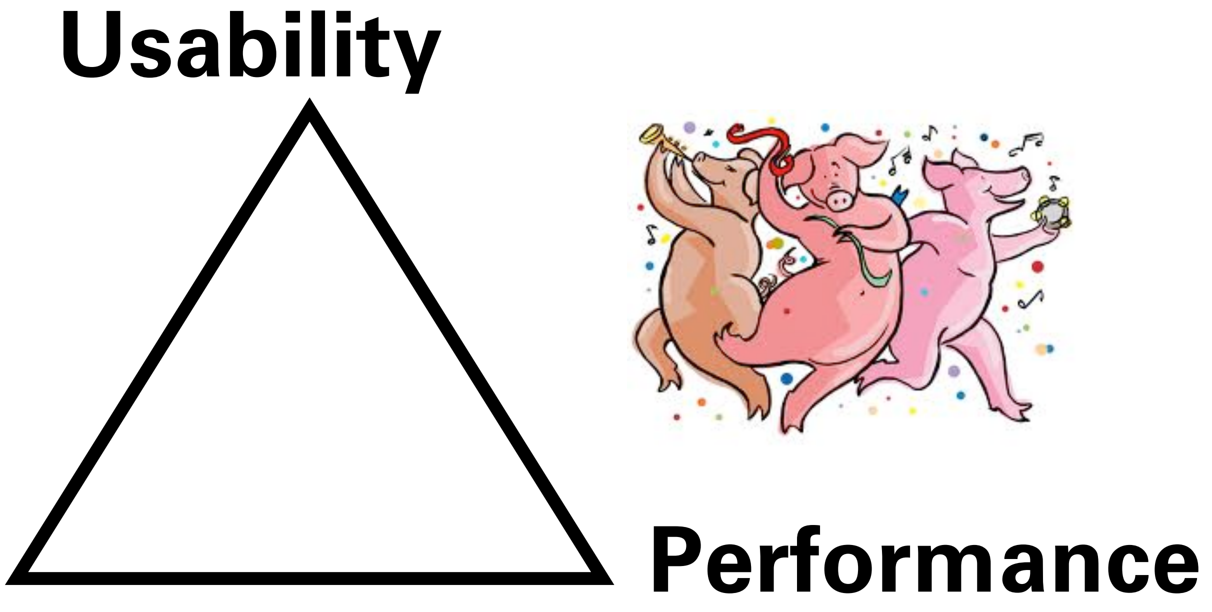
Security Mechanisms and Architectures, Access Control, Cryptography

Information Security

- ISO 17799:
  - Information security is the protection of information [Assets] from a wide range of threats in order to ensure business continuity, minimize business risks and maximize return on investment and business opportunities

- *"The user's going to pick dancing pigs over security every time."* — Bruce Schneier



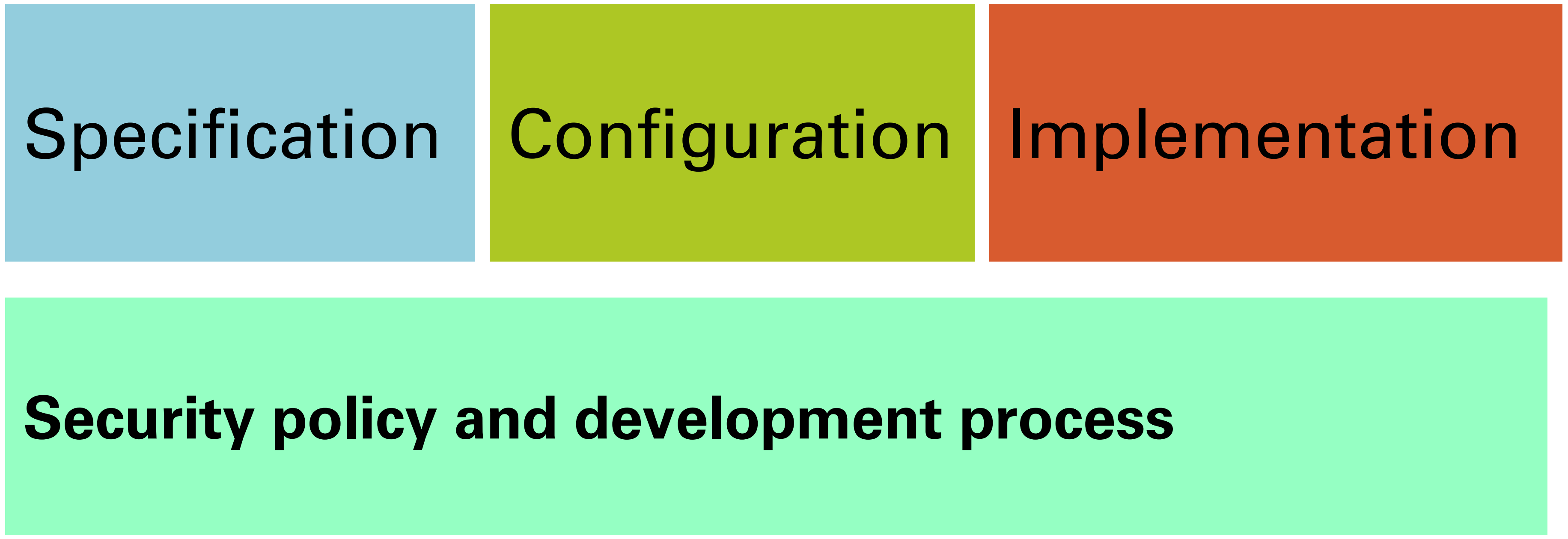
- comfort, functionality and usability
  - vs. security and safety, access control to information system

Information Security

- NIST Special Publication 800-100 (Information Security Handbook: A Guide for Managers):
  - The purpose of information security governance is to ensure that agencies are proactively implementing appropriate information security controls to support their mission in a cost-effective manner, while managing evolving information security risks. As such, information security governance has its own set of requirements, challenges, activities, and types of possible structures. Information security governance also has a defining role in identifying key information security roles and responsibilities, and it influences information security policy development and oversight and ongoing monitoring activities.

“Security is a process”

- A systematic approach is necessary



The security “process”

- The definition of a security policy over the overall system and its environnement
- The deployment of this policy using various security mechanisms
- The monitoring of potential known or new threats during system test or execution (attacks, anomalies, frauds)

Threats to Information Systems

- Defense perimeter
  - Functions deployed over the IS
  - Organizational aspect (hierarchy, roles, etc.)
- Communications
  - Malware injection (e-mail, …)
  - Denial of Service (Spam)
- Servers (not. Web)
  - Privileged target (representative of the organisation)
  - Protection of sensitive data (especially private ones)
- Clients
  - Infection risks (USB interfaces with keys and disks)
  - Physical access to workstations, interception, theft
  - The user may be the weak link (passwords …)

Security Concepts

- Security is about imposing countermeasures to reduce risks to assets to acceptable levels
  - “Perfect security” is not necessary and costly
- A security policy is a specification of what security requirements/goals the countermeasures are intended to achieve
  - secure against what and from whom ?
- Security mechanisms to enforce the policy
  - What actions we should take under an attack?

Security Objectives/Goals: Information System

- CIA triad

- Confidentiality (or secrecy)
  - unauthorized users cannot read information
- Integrity
  - unauthorized users cannot alter information
- Availability
  - authorized users can always access information

- Related properties:
  - Availability: Robustness (critical systems …)
  - Confidentiality/Integrity : Privacy Protection

Security Services: processes/communication

- **Security services according to ITU-T X-800 (OSI context) :**
  - **Confidentiality** (prevent acquiring knowledge of some data)
    - Information (application data, network data like headers)
    - Network traffic (metadata revealing information flow between entities)
  - **Integrity**
    - Of data (preventing data modification)
    - Of program and their execution (original code, vulnerabilities)
  - **Authentication**
    - Of data / program origin (prevent the injection of fake data)
    - Of entities (prevent identity theft)
  - **Authorization / Access Control** (prevent unauthorized accesses)
  - **Non-repudiation** (prevent a posteriori denial of a transaction)
    - With proof of origin
    - With proof of delivery
- **More:**
  - traceability and auditing (forensics)
  - monitoring (real-time auditing)
  - multi-level security
  - privacy & anonymity

Privacy Protection

Privacy: “the degree to which an individual can determine which of his **personal information** must be **shared** with **whom** and in which **purpose”**

Has become VERY IMPORTANT for companies in Europe with GDPR directive (important fines!!).

• ***privacy is*** different from data confidentiality
• superset of the services (confidentiality, access control) targetting “personally identifiable information” (PII)

Specific objectives :
• *data* minimisation (PII disclosed)
• anonymity – not being identifiable among a set of subjets
• unlinkability– preventing knowledge of relationships between entities
• undetectability/unobservability – absence of knowledge of items of interest

Objectives/Services and mechanisms

- Each objective/service targets a set of threats
  - For instance, the confidentiality objective/service protects against information disclosure, eavesdropping threats, unauthorized access to a file, information leaks, etc.
  - Integrity protects against tampering with information
  - Authentication protects against spoofing and replay
  - Access control protects against unauthorized access and intrusions

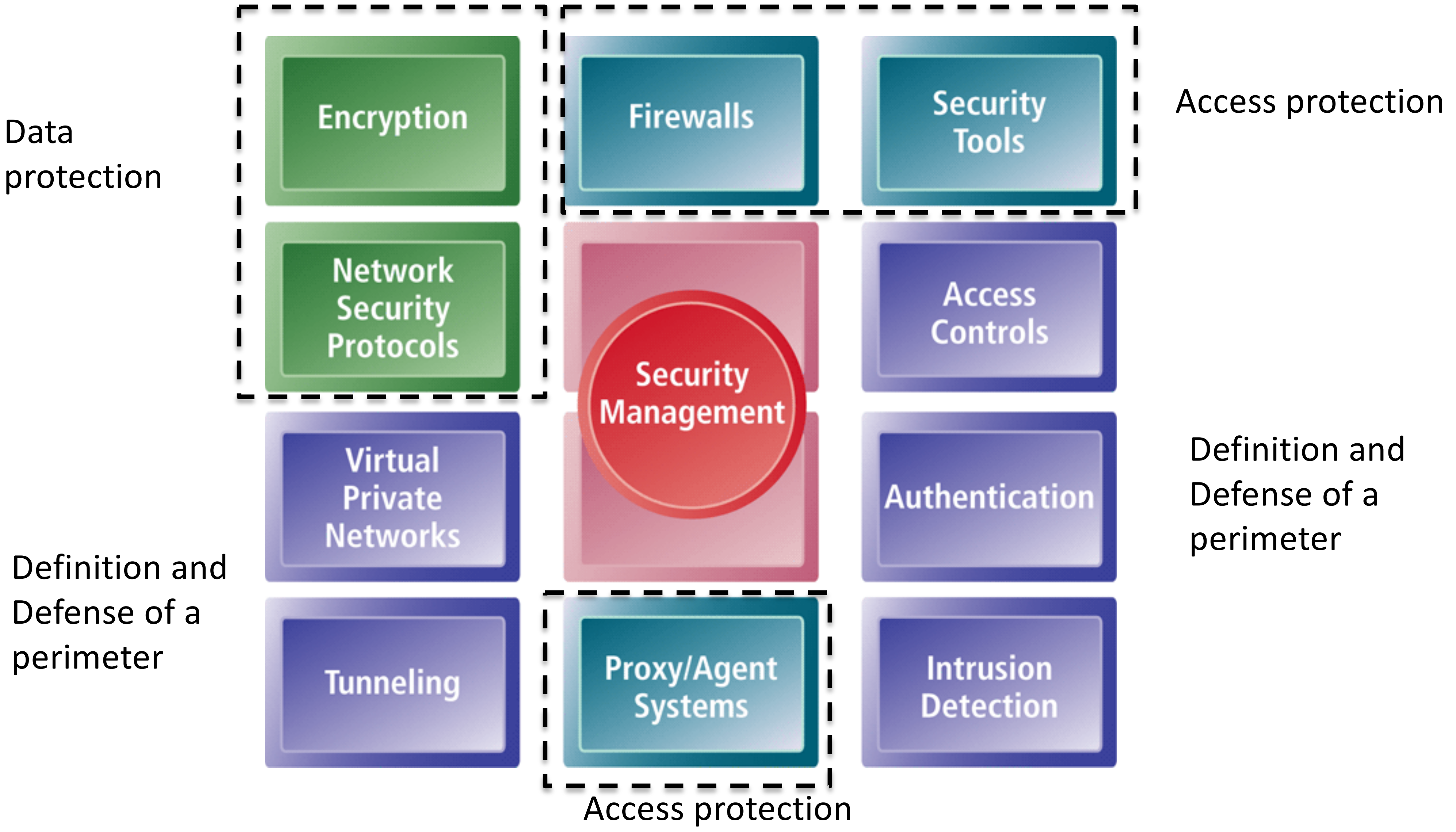
- The objective/security services are themselves implemented using security mechanisms
  - Several objectives/services can use the same mechanism
  - For instance, encryption is used for confidentiality, integrity, and authentication services

- The Services may regard several layers of the OSI model
  - May lead to using various mechanisms, depending on the location of information and logical functions to be protected
  - Privacy protection for instance requires the use of mechanisms at all layers
  - The highest layers provide for the finer granularity but also the most complex and difficult integration of security

How to Realize Security Objectives?

- Approaches:
  - Prevention
    - measures to stop breaches of security goals
  - Detection
    - measures to detect breaches of security goals
  - Reaction
    - measures to recover assets, repair damage, and fight (and deter) offenders
- Good prevention does not make detection & reaction superfluous
  - E.g., breaking into any house with windows is trivial; despite this absence of prevention, detection & reaction still deter burglars
- Defense in depth!!

Security Mechanisms



Security Mechanisms

- Non exhaustive classification:
  - Data confidentiality mechanisms
  - Data integrity mechanisms
  - Digital signature
  - Access control mechanisms
  - Authentication mechanisms
  - Privacy enforcement techniques (PET)
  - Notarization
  - Key management
  - Audit
  - Intrusion or anomaly detection
  - Platform integrity protection

- These mechanisms make use of mathematical, software, or hardware components for their implementation, notably:
  - Data protection: cryptographic or steganographic primitives (encryption algorithms, hash functions, …), or hardware components (trusted computing)
  - Access protection: filtering tools (packet filter, application proxy)
  - Perimeter protection: virtual private networks, monitoring tools for network and systems, user databases

Security Mechanisms

- Language-based security
  - for threats related to misbehaving programs
    - typing, memory-safety
    - sandboxing
  - E.g., Java, .NET/C#
- Security Architectures and Logical Access control
  - for threats related to misbehaving users
  - Definition and defense of a perimeter
  - Data / access protection
  - Access control models (E.g., role-based access control)
- Cryptography
  - for threats related to insecure communication and storage
  - Also used for algorithmic access control

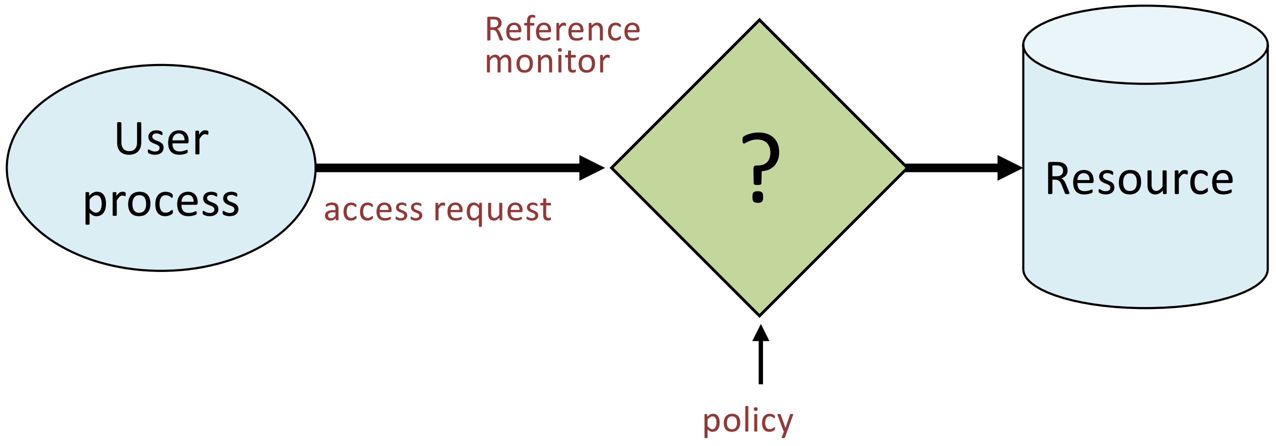
Authorization, Access Control

Authorization, Access Control

- What is a subject/principal authorized to do?
- Definition of one or several perimeters of protection
  - Authorisations over assets/resources or groups of resources called “objects”
  - The authors of actions are called “subjects”
  - The “rights” granted to subjects over objects are formalized in a matrix
  - Security policy focused on the notion of perimeter
- The enforcement of these perimeters can intervene at different levels:
  - Applications
  - Operating system (or virtual machine)
  - Network

(Logical) Access control models

- Assumptions
  - System knows who the user is
    - Authentication via name and password, other credential
  - Access requests pass through gatekeeper (reference monitor)
    - System must not allow monitor to be bypassed



Access control matrix [Lampson]

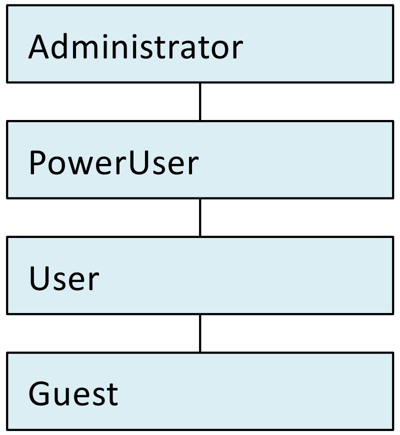
Objects

Subjects

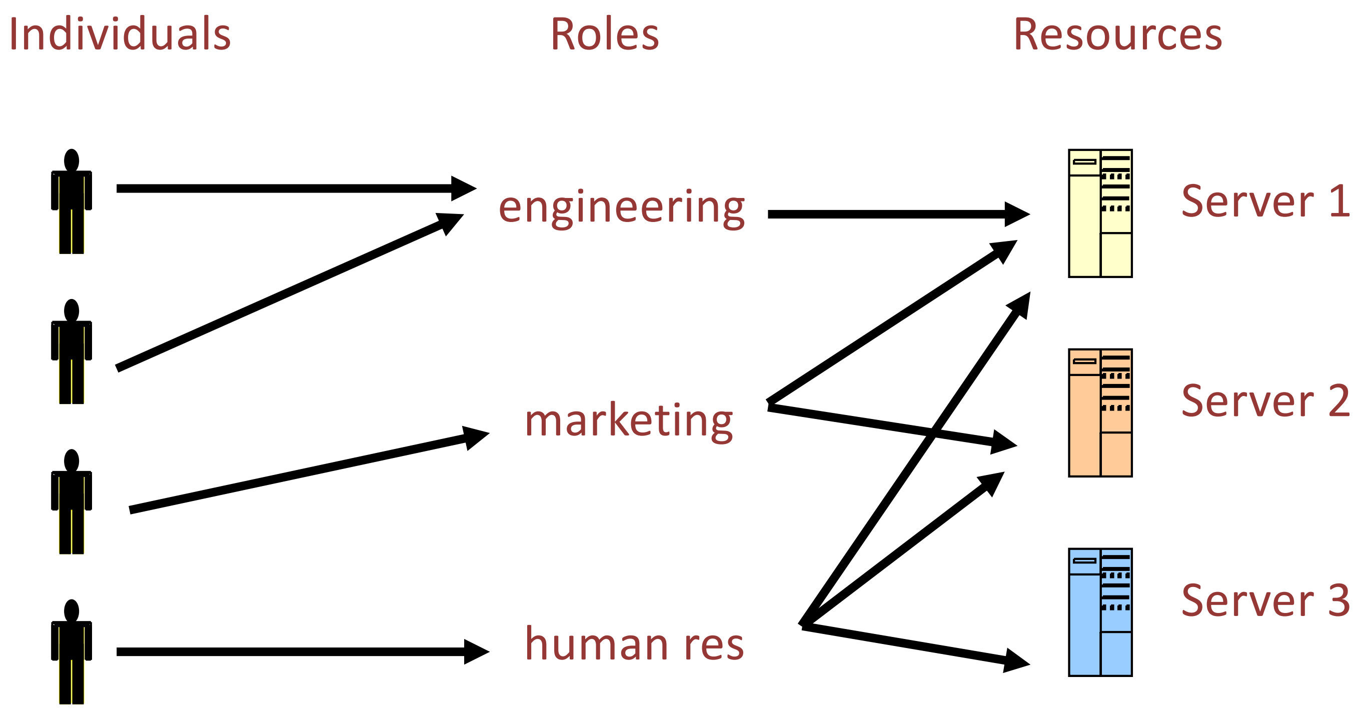
	File 1	File 2	File 3	…	File n
User 1	read	write	-	-	read
User 2	write	write	write	-	-
User 3	-	-	-	read	read
…					
User m	read	write	read	write	read

Roles (aka Groups)

- Role = set of users
  - Administrator, PowerUser, User, Guest
  - Assign permissions to roles; each user gets permission
- Role hierarchy
  - Partial order of roles
  - Each role gets permissions of roles below
  - List only new permissions given to each role



Role-Based Access Control



Advantage: users change more frequently than roles

Implementation concepts

- Access control list (ACL)
  - Store column of matrix with the resource
- Capability
  - User holds a “ticket” for each resource
  - Two variations
    - store row of matrix with user, under OS control
    - unforgeable ticket in user space

	File 1	File 2	…
User 1	read	write	-
User 2	write	write	-
User 3	-	-	read
…			
User m	Read	write	write

Access control lists are widely used, often with groups

Some aspects of capability concept are used in many systems

ACL vs Capabilities

- Access control list
  - Associate list with each object
  - Check user/group against list
  - Relies on authentication: need to know user
- Capabilities
  - Capability is unforgeable ticket
    - Random bit sequence, or managed by OS
    - Can be passed from one process to another
  - Reference monitor checks ticket
    - Does not need to know identify of user/process

ACL vs Capabilities

- Delegation
  - Cap: Process can pass capability at run time
  - ACL: Try to get owner to add permission to list?
    - More common: let other process act under current user
- Revocation
  - ACL: Remove user or group from list
    - Or prevent process from acting as owner
  - Cap: Try to get capability back from process?
    - Possible in some systems if appropriate bookkeeping
      - OS knows which data is capability
      - If capability is used for multiple resources, have to revoke all or none …
    - Indirection: capability points to pointer to resource
      - If $C \rightarrow P \rightarrow R$, then revoke capability C by setting $P=0$

Access control: summary

- Access control involves reference monitor
  - Check permissions: $\langle$user info, action$\rangle \rightarrow$ yes/no
  - Important: no way around this check
- Access control matrix
  - Access control lists vs capabilities
  - Advantages and disadvantages of each
- Role-based access control
  - Use group as “user info”; use group hierarchies

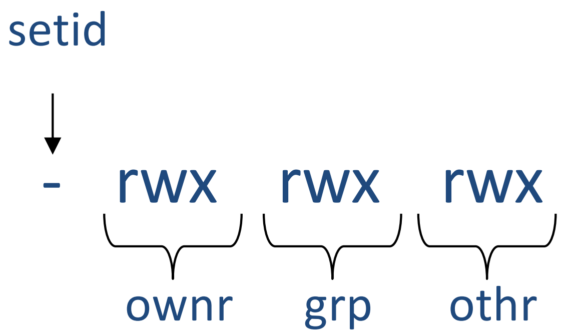
Unix access control

- Process has user id
  - Inherit from creating process
  - Process can change id
    - Restricted set of options
  - Special “root” id
    - All access allowed
- File has access control list (ACL)
  - Grants permission to user ids
  - Owner, group, other

	File 1	File 2	…
User 1	read	write	-
User 2	write	write	-
User 3	-	-	read
…			
User m	Read	write	write

Unix file access control list

- Each file has owner and group
- Permissions set by owner
  - Read, write, execute
  - Owner, group, other
  - Represented by vector of four octal values



- Only owner, root can change permissions
  - This privilege cannot be delegated or shared
- Setid bits – Discussed in a few slides

Process effective user id (EUID)

- Each process has three Ids (+ more under Linux)
  - Real User ID (RUID)
    - same as the user ID of parent (unless changed)
    - used to determine which user started the process
    - System call *access()* determines permission based on RUID
  - Effective User ID (EUID)
    - from set user ID bit on the file being executed, or sys call
    - determines the permissions for process
      - file access and port binding
  - Saved User ID (SUID)
    - So previous EUID can be restored

- Real Group ID, Effective Group ID, used similarly

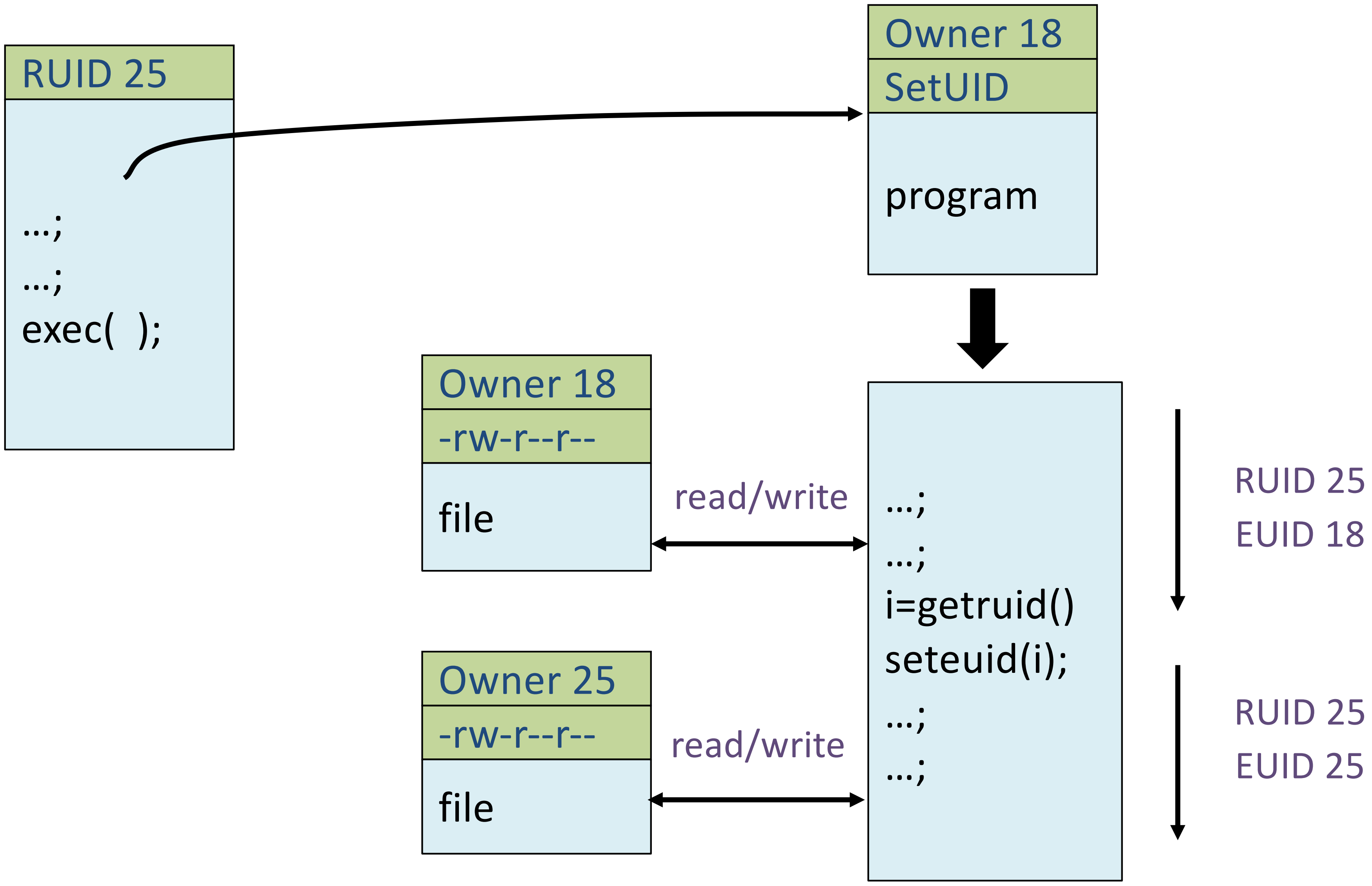
Setid bits on executable Unix file

- Three setid bits
  - Setuid – set EUID of process to ID of file owner
    - “chmod u+s”
  - Setgid – set EGID of process to GID of file
    - “chmod g+s”
  - Sticky
    - Off: if user has write permission on directory, can rename or remove files, even if not owner
    - On: only file owner, directory owner, and root can rename or remove file in the directory, but all owners with a write permission can modify it
    - ”chmod +t”

Process Operations and IDs

- Root
  - ID=0 for superuser root; can access any file
- Fork and Exec
  - Inherit three IDs
  - except exec of file with setuid bit (then: euid and suid receive uid of owner of process invoked)
- Setuid system calls
  - setuid(newid) can set RUID to newid (if you have the right to, e.g. root)
    - However user root or program setuid that changes to newid != 0 cannot regain root privileges !
  - seteuid(newid) can set EUID to
    - Real ID or saved ID, regardless of current EUID
    - Any ID, if EUID=0
  - Details are actually more complicated: several different calls: setuid, seteuid, setreuid, setgid, …

Example


RUID 25
...;
...;
exec( );
Owner 18
SetUID
program
Owner 18
-rw-r--r--
file
read/write
Owner 25
-rw-r--r--
file
read/write
...;
...;
i=getruid()
seteuid(i);
...;
...;
RUID 25
EUID 18
RUID 25
EUID 25

Unix summary

- Advantages
  - Some protection from most users
  - Flexible enough to make things possible
- Main limitations
  - Too tempting to use root privileges
  - No way to assume some root privileges without all root privileges

Mandatory Access Control (MAC)

- Access Control Models: Mandatory vs. Discretionary
  - Discretionary (DAC) = at the discretion of the resource owner
- A means of restricting access to objects based on the sensitivity of the information contained in the objects and whether they are authorized to access information of such sensitivity
- Authorization is based on prerequisites being met, resulting in an individual gaining access
- Enables the ability to deny users full control over the access to resources that they create
- access control is based on the compatibility of the security properties of the data and the clearance properties of the individual

SELinux

- Originally started by the Information Assurance Research Group of the NSA, working with Secure Computing Corporation.
- Based on a strong, flexible mandatory access control architecture based on Type Enforcement, a mechanism first developed for the LOCK system
- Originally started as two prototypes: DTMach and DTOS which were eventually transferred over to the Fluke research operating system
- Eventually the architecture was enhanced and renamed Flask. The NSA has now integrated the Flask architecture with Linux (SELinux)

SELinux: Background

- An example of how mandatory access controls can be added into Linux
  - Confining actions of a process, including a superuser process
- The security mechanisms implemented in the system provide flexible support for a wide range of security policies.
- Make it possible to configure the system to meet a wide range of security requirements.
- Documentation and source code is provided.

Secure Architecture Principles

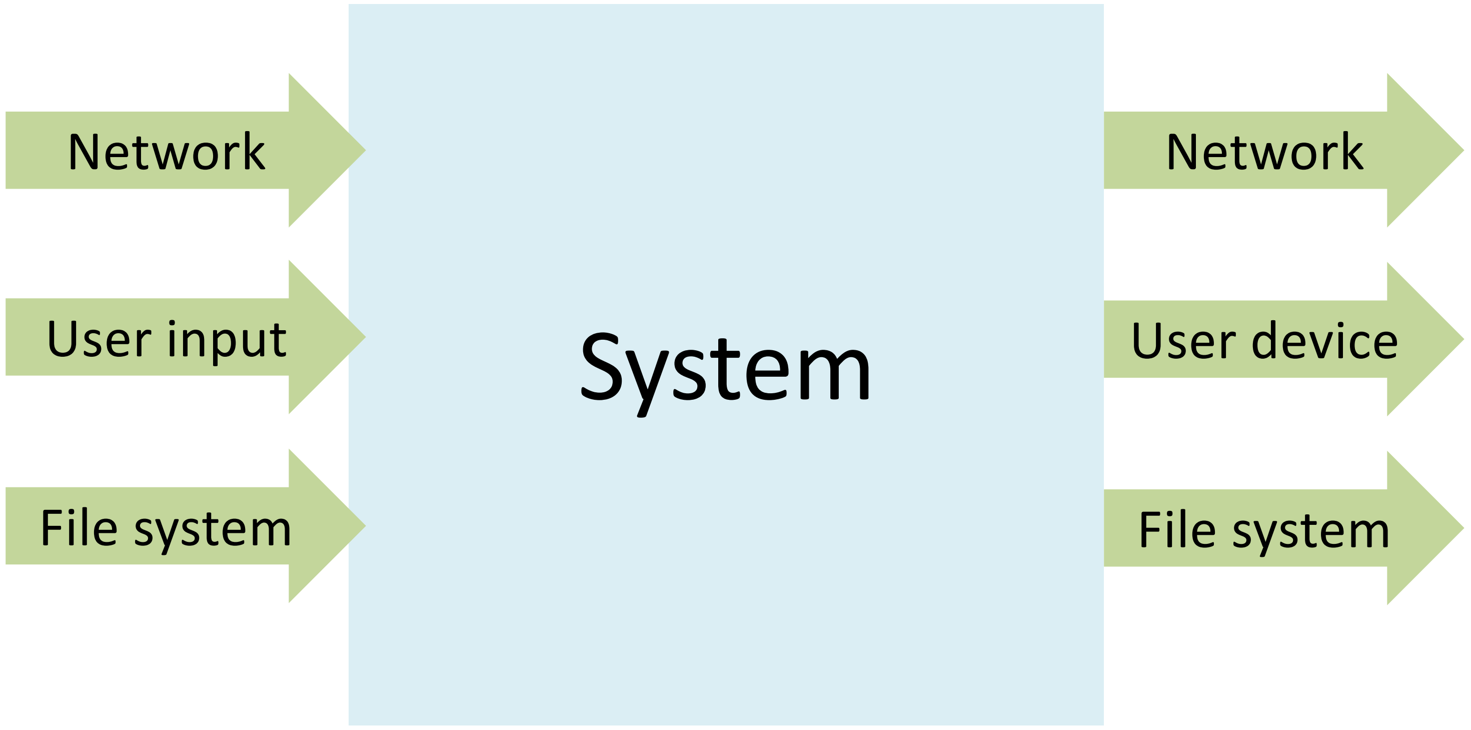
Principles of Secure Design

- Compartmentalization
  - Principle of least privilege
  - Isolation
- Defense in depth
  - Use more than one security mechanism
  - Secure the weakest link
  - Fail securely
- Keep it simple

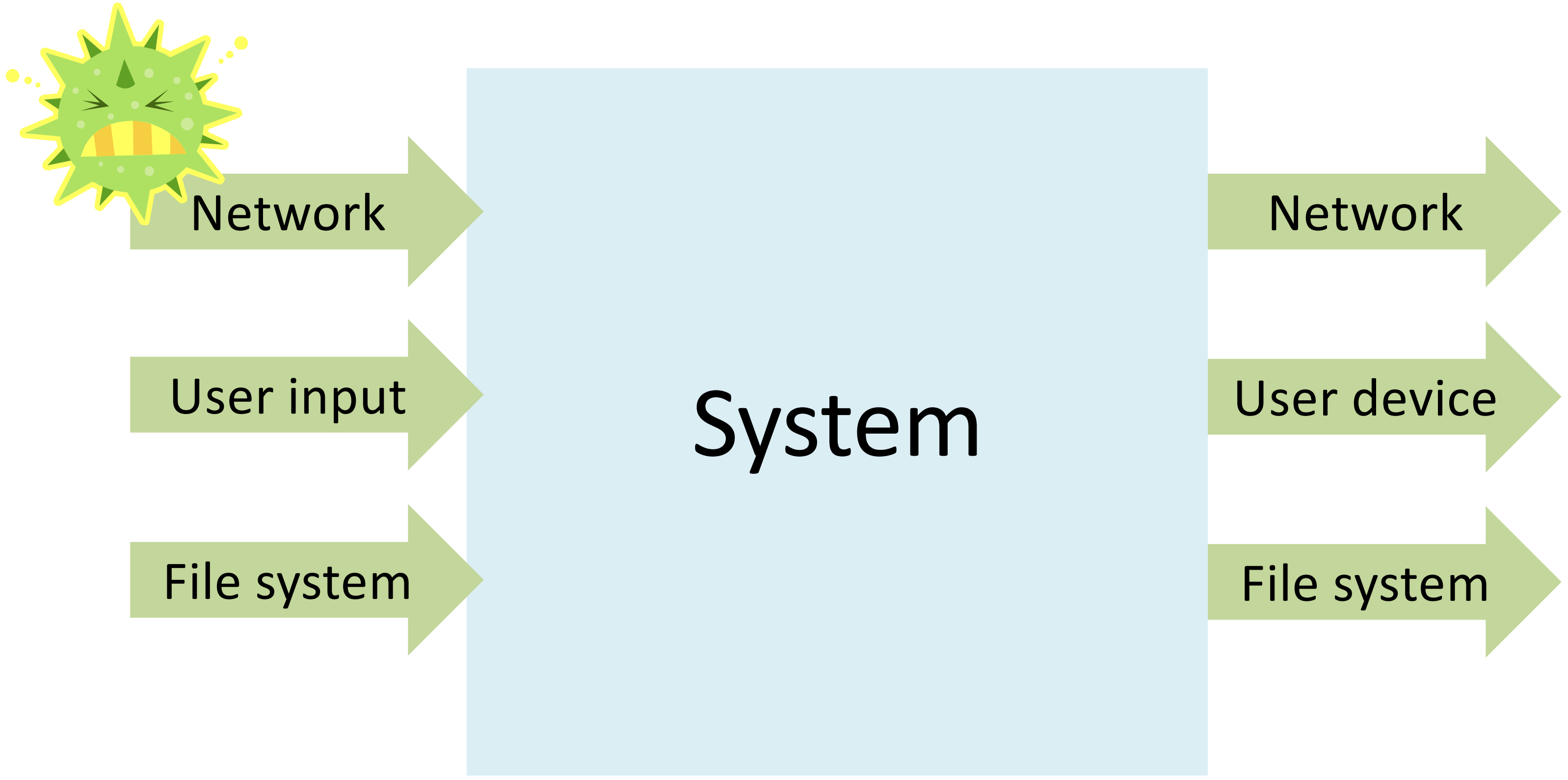
Principle of Least Privilege

- What’s a privilege?
  - Ability to access or modify a resource
- Assume compartmentalization and isolation
  - Separate the system into isolated compartments
  - Limit interaction between compartments
- Principle of Least Privilege
  - A system module should only have the minimal privileges needed for its intended purposes

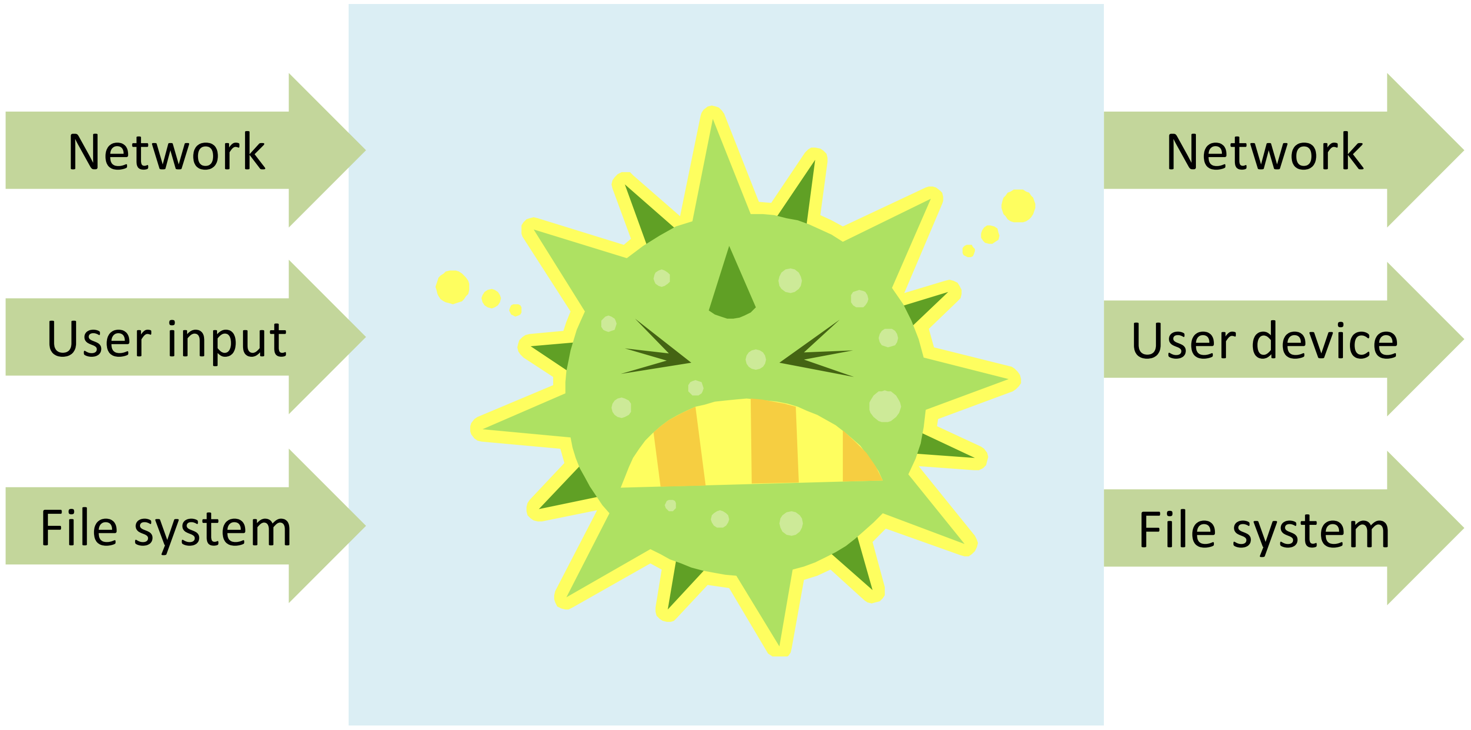
Monolithic design



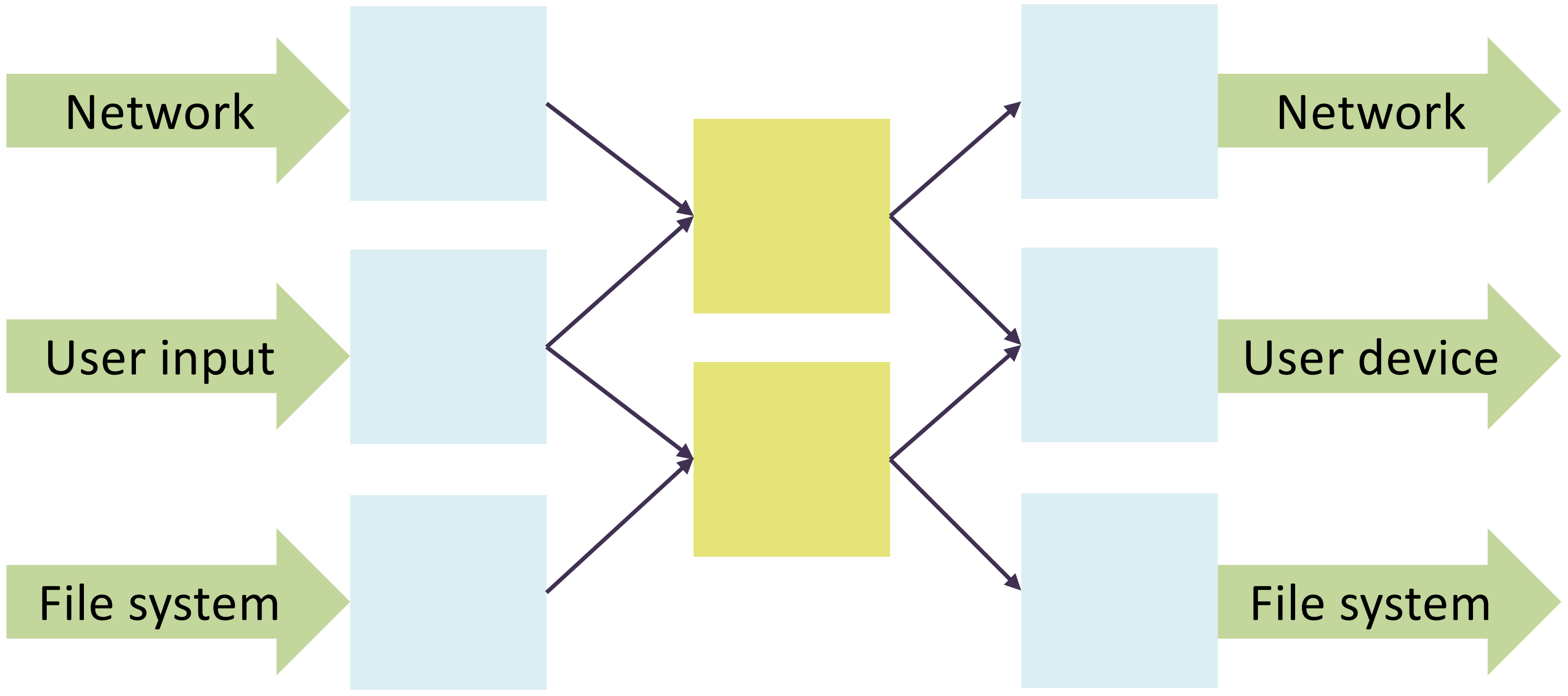
Monolithic design



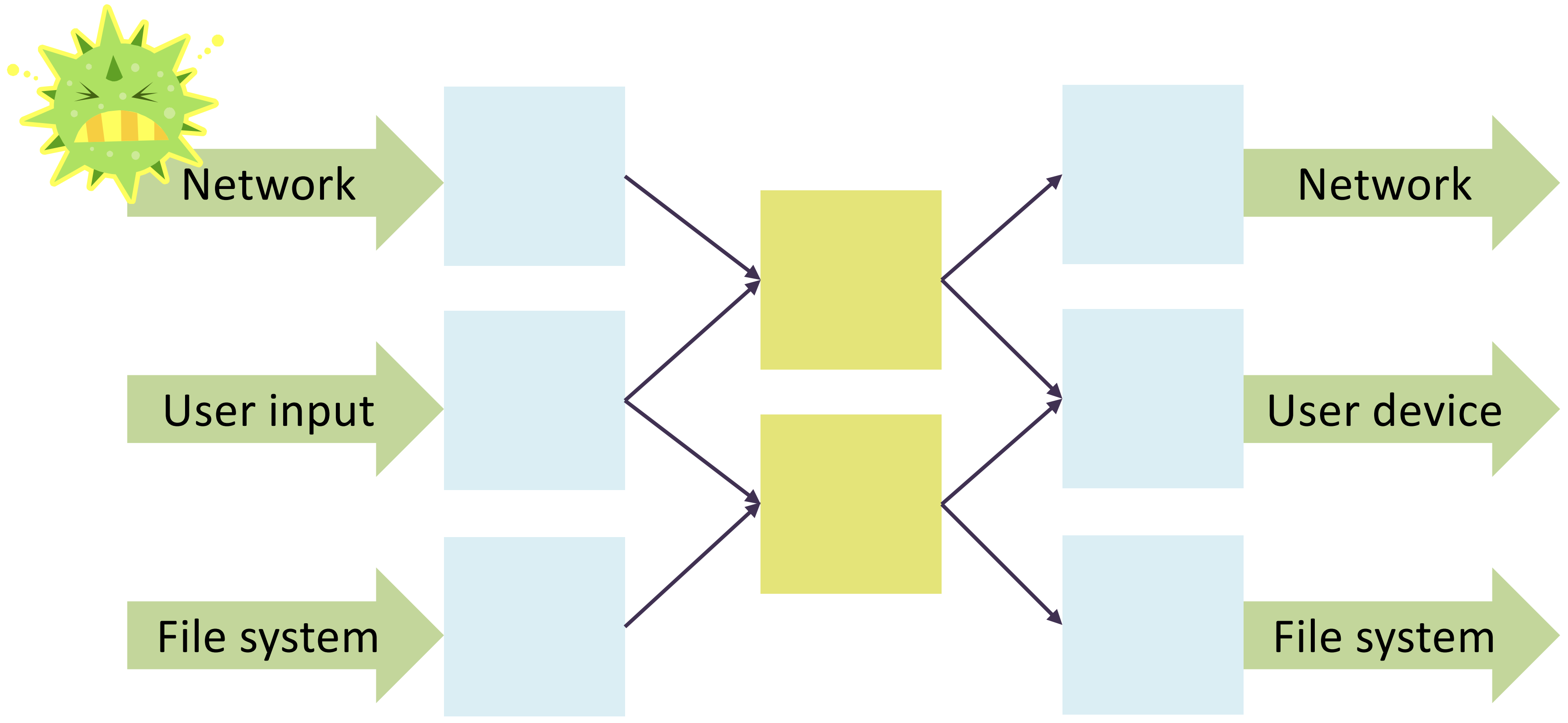
Monolithic design



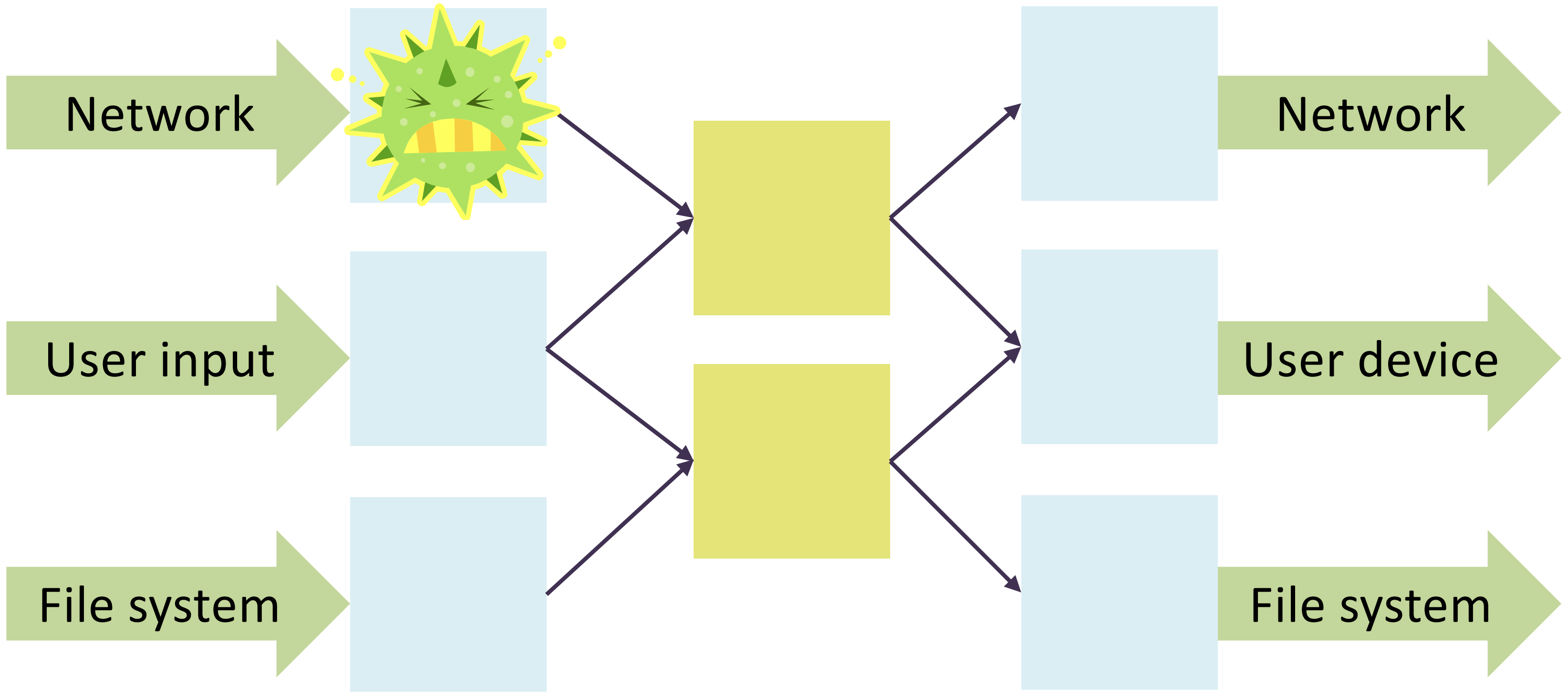
Component design



Component design



Component design



Principle of Least Privilege

- What's a privilege?
  - Ability to access or modify a resource
- Assume compartmentalization and isolation
  - Separate the system into isolated compartments
  - Limit interaction between compartments
- Principle of Least Privilege
  - A system module should only have the minimal privileges needed for its intended purposes

Isolation between processes

- Processes in OS:
  - A process may access files, network sockets, ….
    - Permission granted according to UID
  - Two processes with same UID have the same permissions
- Processes and privileges :
  - Compartment defined by UID
  - Privileges defined by actions allowed on system resources

Example: Mail Agent

- Requirements
  - Receive and send email over external network
  - Place incoming email into local user inbox files
- Sendmail
  - Traditional Unix
  - Monolithic design
  - Historical source of many vulnerabilities
- Qmail
  - Compartmentalized design

Qmail design

- Function isolation based on OS isolation
  - Separate modules run as separate “users”
  - Each user only has access to specific resources
- Least privilege
  - Minimal privileges for each UID
  - Only one “setuid” program
    - setuid allows program to run as different users
  - Only one “root” program
    - root program has all privileges