Security
SGG: Chapters 14 & 15

I. Overview & Background

A. So far, we've been focused on ensuring correctness, increase reliability and generally preventing bad behavior/error that's **incidental**, e.g.
   1. deadlock due to incidental resource contention;
   2. process starvation due to pathological scheduling,
   3. isolating processes in memory
   4. avoiding the use of structures and algorithms that result in poor system performance, *etc*.
   5. In each of the above, the errors are a result of passive behavior, and without malice (wanting to cause harm)

B. However, the problem becomes more difficult with subjects (e.g. processes, users) become **adversarial** and **actively attack** the system
   1. adversaries can range from: **passive** (honest but curious, eavesdropping) to **active** (malicious, system misuse, malware, DOS, data destruction)

C. **Scope** and **variety** of threats have increased with
   1. Increased value of digital data:
      a) Types of data: financial data, PII, IP, etc.
      b) Owned by a variety of targets: governments, private industry, individuals
   2. More targets (both users and machines)
      a) Devices: laptops, desktops, mobile, industrial control, cars, houses, medical devices
      b) Used for: personal, business, national defense, critical infrastructure
   3. Pervasive networks make targets easier to get to

a) Internet connection is standard and presumed for functionality
b) No longer requires physical access
4. Ubiquity of homogenous systems
a) Greater ROI for an attack

II. Guiding Principle & Challenges

A. **Principle of Least Privilege**: Only grant the permissions an subject needs to accomplish task, and nothing more
1. Example: Does the print driver need write access to any resource but the printer? Does it need read access to anything but the print queue?
a) Superficially, the answer is no, but “real life” implementations get complicated
b) Does print spooler maintain a log file?
c) Is it a network printer, does it check for software updates?
d) Does it notify when printing is done or has an error?
e) Is it a user-space application or a kernel-space application?
f) Solutions are nuanced, and this is just a print driver!
2. **Goal**: Develop security mechanism that protect execution and data is a system with very complex interaction.

B. Difficult to design systems that are **functional** and **secure**
1. Increased functionality leads to complexity
2. Complexity leads to misunderstandings
3. Misunderstandings lead to errors
4. Errors lead to vulnerabilities
5. Vulnerabilities lead to exploits

C. In reality: Two pronged approach to system design
1. highly constrained systems that enforce security goals with a high degree of assurance
a) e.g. OS that runs a nuclear power plant or space shuttle
b) Expensive and slow to design and implement, little functionality
2. general-purpose system that enforce very limited security

goals, with far fewer constraints; greater functionality and flexibility

a) e.g. an OS that can play a networked game of minesweeper

D. Security has not been a primary design criteria for general purpose operating systems

1. (then we plugged those systems into the Internet)
2. Result is legacy software & systems, that are highly vulnerable and more difficult to secure
3. Playing catchup with a constant barrage of bolt-on fixes, which increase complexity
4. Good news: we know how to build secure systems
5. Bad news: few do

III. Goals for a Secure OS

A. A secure OS must have mechanisms that ensure security goals are enforced despite threats

B. Typical goals: confidentiality, integrity, availability, (CIA) and trust

C. **Confidentiality**: keeping data and execution private

1. Only authorized subjects (e.g. processes and users) can access (read) confidential information

D. **Integrity**: keep data and code unmodified

1. Only authorized subjects can modify

E. **Availability**: data must be widely available

1. No system is useful if we can never access it (due to over bearing security or an attacker's denial of service)

F. **Trust**: our belief that an entity will behave correctly

1. Any kind of enforcement requires **trust**
2. The challenge is: **establishing** and **verifying** trust

a) Do we trust a large amount of code is bug free?

b) Do we trust our CPU to compute something correctly?

c) Do we trust that the OS will enforce access control bits set on files?

d) Do we trust the OS to enforce the isolation of processes?

IV. Access Control Fundamentals

A. One of the key security primitives provided by the OS is providing an access control mechanism and enforcing access control policies

B. Terminology: An access enforcement mechanism authorizes requests (e.g., system calls) from multiple **subjects/entities** (e.g., users, processes, etc.) to perform **operations** (e.g., read, write, up, down etc.) on **objects** (e.g., files, sockets, disk drives, semaphores etc.).

C. We need some way to concisely describe the allowable operations by subjects on objects

D. Lampson **Access Matrix**

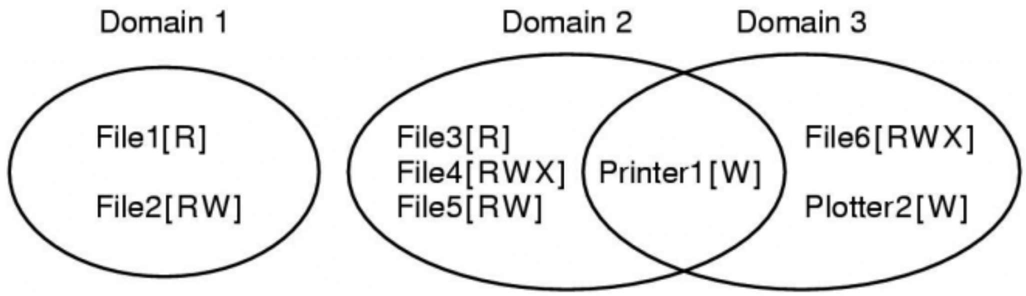
1. A matrix where rows represent subjects; columns represent objects and entries define the allowable operations

	File 1	File 2	File 3	Process 1	Process 2
Process 1	Read	Read, Write	Read, Write	Read	-
Process 2	-	Read	Read, Write	-	Read

2. Matrix defines the allowable operations on objects by subjects
3. Also, defines operations that determine which subjects can modify cells
4. Note: Subjects can also be objects!
5. Matrix is used to define the **protection domain** of a process

E. Domains of Protection

1. A **domain**: a set of object-right pair
   a) a **right** is the set of operations permissible
2. A **protection domain** specifies the set of resources that a process/user can access, and the operations that can be performed
   a) A domain can correspond to a single object (e.g. process or user) or a collection (e.g. a group of users)



3. At any moment in time, a process is in some domain, but may also switch domains during execution (dynamic)
   a) May be able to switch into a new domain, but not switch back
   b) The OS enforces all of the above
4. In UNIX: domains are associated with a user and a group
   a) domains are defined by UID and GID
   b) A user's shell gets its UID/GID from the password file
   c) All child processes inherit domain
   d) Different users will have different UIDs, but may share GIDs
   e) An user can temporarily change its domain using the SETUID bit

V. Implementation of Access Matrix

A. Too expensive to keep a global table of all subjects and objects
   1. Plus, the table is largely sparse

B. Access Control Lists
   1. Column-wise view of the matrix
   2. Defines the access rights of an object, stored with the object
      a) e.g. keypad door lock, bounce at a club
   3. Can specify both positive and negative permissions
   4. This is a common model for most commodity OSes
      a) 9 permission bits: r,w,x for u,g,a
      b) Advantageous when objects are persistent and numerous

C. Capabilities
   1. Row-wise view of the matrix

2. Defines the access rights of a subject, stored with object or subject
    a) e.g. concert ticket, car key
3. Subjects carry around with them their abilities

D. ACLs or Capabilities?

E. Discretionary Access Control
1. Most commodity OSes support a discretionary access control (DAC) policy
2. The empowers users to change permissions, enforced by the OS
3. This can be abused by malicious software (e.g. Trojan Horse)
    a) Do all the things a user can do, include change permissions

F. Mandatory Access Control
1. Both setting and enforcing permission done by a trusted component in the OS
2. Users are disempowered from modifying their permissions
3. Used in multi-level security systems (systems that support multiple security levels concurrently)

G. Trusted Computing
1. Reference monitor

H. Covert Channels

VI. Authentication

A. Ensuring only those users authorized to access the system do so

B. How does a subject prove her identity to a system?

C. Factors of authentication: Know, Have, Are

D. Common are passwords

VII. Memory Protection

A. As mentioned, one of the biggest issues making computer insecure today is the confusion between data and code

B. Think back to Program 1

1. How easy was it to modify the flow of execution (i.e. manipulate the stack)?

C. Buffer overflows

D. Permission Bits

1. Memory pages need bits to know what kind or what operations should be allowed
2. NoExec

E. StackGuard

F. ASLR

1. Makes address layout in logical space randomized
2. e.g. makes jumping to a function at a known location more difficult

VIII. Cryptographic Services

A. Storer of Secrets

1. In software (data that the OS protects)
2. In hardware (resource that the OS manages)

IX. Defenses

A. Firewalls

B. IDS

C. Code Signing

D. Jailing/Sandboxing