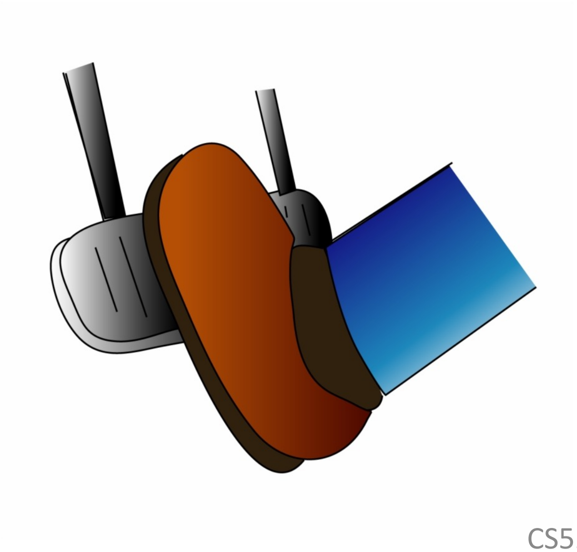
CS5231: Systems Security

Lecture 5: Isolation

Second Line of Defense

- First Line of Defense
  - Directly prevent the attack from happening
- Second Line of Defense
  - Assume that attack happens, minimize the impact



Vs.

Sandboxing: Access Control

Access Control Primitives

- Definitions:
  - Resource Objects

    "Elements that need to be protected"
  - Authorities or Principals

    "Subjects accessing the resources"
  - Permissions

    "Access Rights"
  - Isolation Environment (or protection domain)

    "A domain in which program executes. It determines what the program will do."

Access Control Matrix

Directory

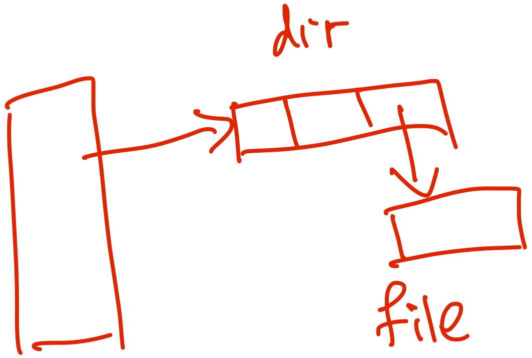
	BIBLIOG	TEMP	F	HELP.TXT	C_COMP	LINKER	SYS_CLOCK	PRINTER
USER A	ORW	ORW	ORW	R	X	X	R	W
USER B	R	-	-	R	X	X	R	W
USER S	RW	-	R	R	X	X	R	W
USER T	-	-	-	R	X	X	R	W
SYS_MGR	-	-	-	RW	OX	OX	ORW	O
USER_SVCS	-	-	-	O	X	X	R	W

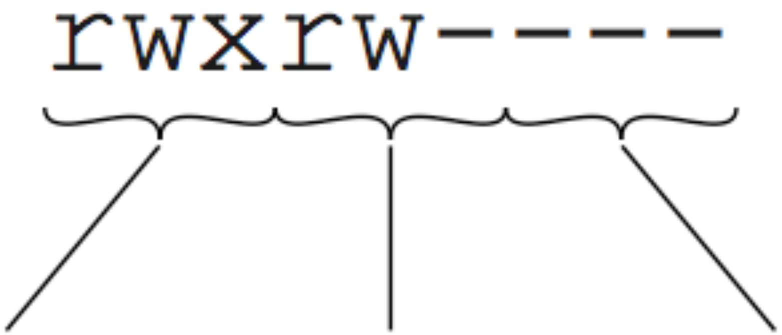
Access
Control List

Access Rights or Permissions

Protection and access control in operating systems [Lampson'72]

Example: UNIX File Access Control





rwxrw----

Owner can read, write, and execute the file

Any user in the owner's group can read and write the file

Users outside the group cannot read, write, or execute the file.

x for entering directories

Example of Delegation & Groups: UNIX File Access Control

- “set user ID”(SetUID) or “set group ID”(SetGID)
  - system temporarily uses rights of the file owner / group in addition to the real user’s rights when making access control decisions
  - enables privileged programs to access files / resources not generally accessible
- sticky bit
  - on directory limits rename/move/delete to owner
- superuser
  - is exempt from usual access control restrictions

Example of Delegation & Groups: UNIX Access Control Lists

- modern UNIX systems support ACLs
- can specify any number of additional users / groups and associated rwx permissions
- ACLs are optional extensions to std perms
- group perms also set max ACL perms
- when access is required
  - select most appropriate ACL
    - owner, named users, owning / named groups, others
  - check if have sufficient permissions for access

Summary of Definitions: Access Control Primitives

- Definitions:
  - Authorities or Principals

    "Subjects accessing the resources"
  - Resource Objects

    "Elements that need to be protected"
  - Permissions

    "Access Rights"
  - Isolation Environment (or protection domain)

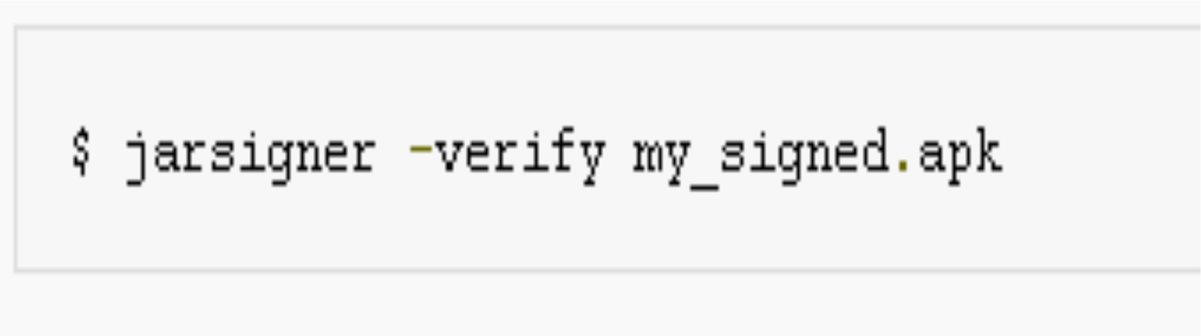
    "A domain in which program executes. It determines what the program will do."

An Example: Android OS

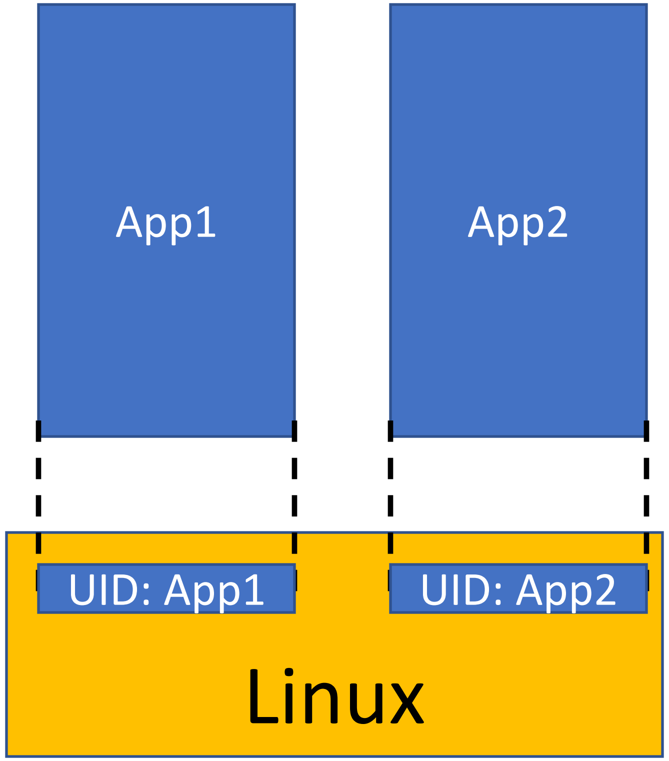
Reading: [The Android Platform Security Model]

Authorities / Principals

- Each applications is a unique authority (Unix user ids)
- Each app is **signed**



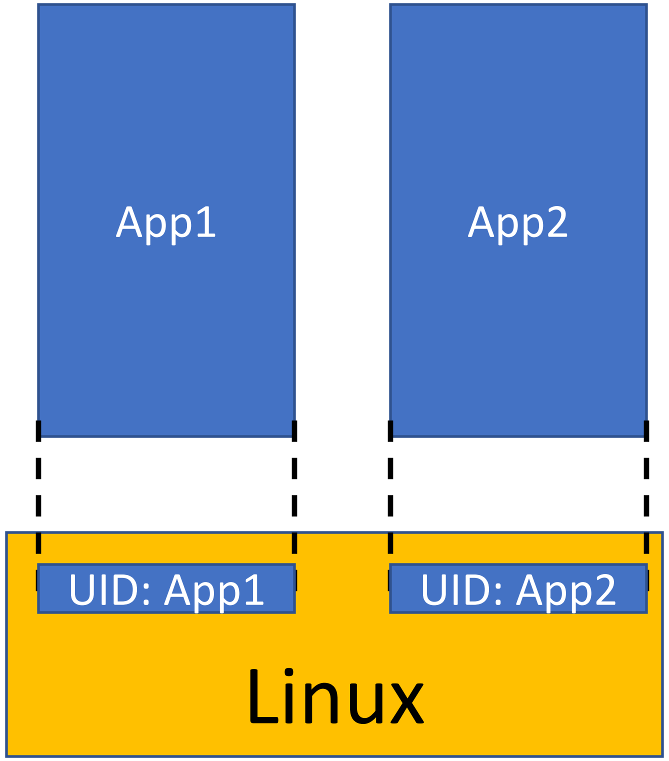
```
$ jarsigner -verify my_signed.apk
```



Isolation Environment

- Isolation via OS Processes
- Why is it better?
  - E.g. Apple iOS browser bug
    - Safari exploit [Miller'08]
    - Lead to compromising the Whole phone!

  - On Android, confined to browser app (UID) only!



Access Control Policies

Variety of Policies Enforcable…

- Linux *seccomp*
  - cannot make any syscalls except exit(), sigreturn(), read() and write() to already-open file-desc
- Linux *seccomp-bpf*
  - Configurable policies
- Linux Security Modules

- Policies can include syscall data args as well

Policy Design Principle: Allow-listing > Block-listing

- Allow-listing vs. Block-listing in Policies
  - Better to Specify what's allowed
  - Rather than Specify what's not allowed

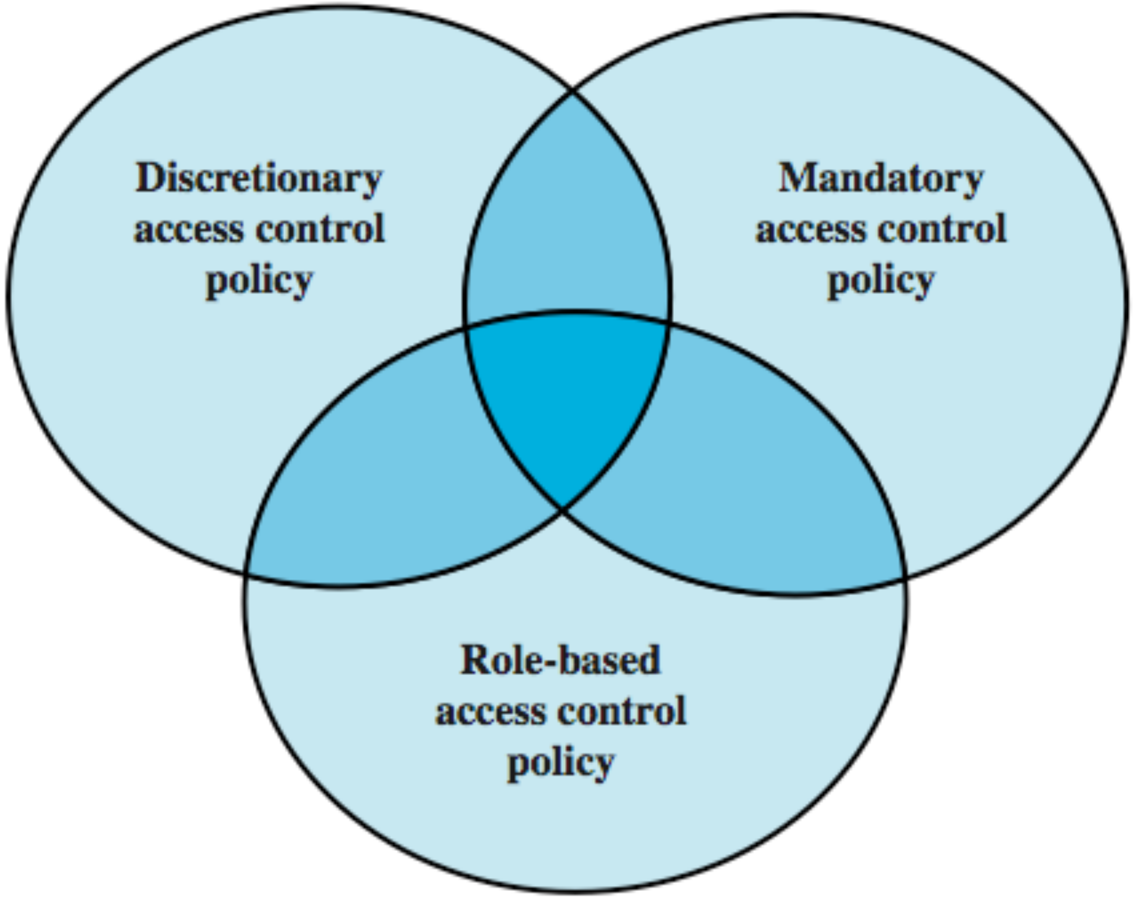
- Block-listing: E.g. No exec-after-read

- Allow-listing: E.g. *seccomp()* allows 4 syscalls!
- Follows the principle of *least privilege*

Access Control Policies

DAC

User decides control


Discretionary access control policy
Mandatory access control policy
Role-based access control policy

MAC

System decides control

Discretionary Access Control

- No fixed policy!
- Each owner decides the access rules
- Example: UNIX File Systems

Mandatory Access Control

- Policy fixed by the administrator
- Each owner cannot change access rights of objects created or owned by it

Examples of Mandatory AC: Same-origin Policy

http://evil.com
http://google.com

No direct access between these frames !

https://**www.nus.edu.sg**:**443**/~prateeks/add.php?q=x

PROTOCOL

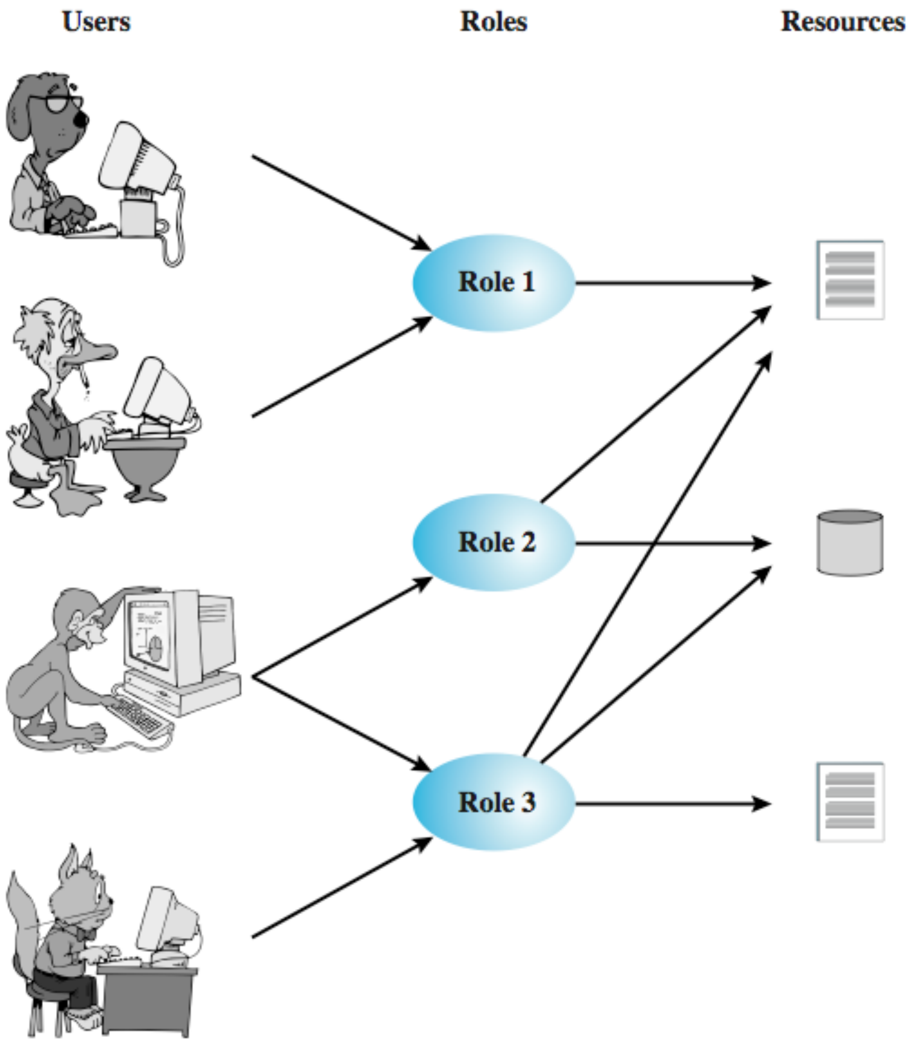
HOST
(Domain)

PORT

WEB ORIGIN = PROTOCOL + HOST+ PORT

1. [Same-origin policy [Wikipedia]]
2. [RFC 6454]

Role-Based Access Control

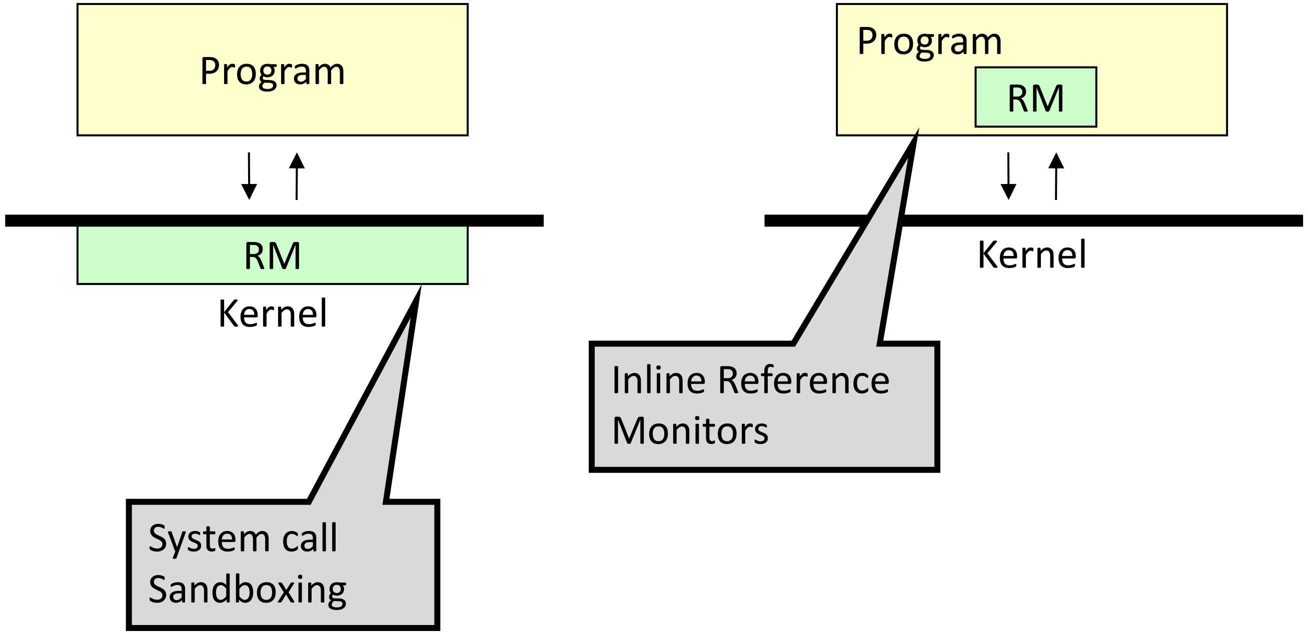


Process sandboxing & Inline Reference Monitors

Reference Monitors

Reference Monitor: A piece of code that *checks all* **references** to an **object**

Syscall Sandbox: A reference monitor for protecting OS resource objects from an app



Slide from Shmatikov et. al.

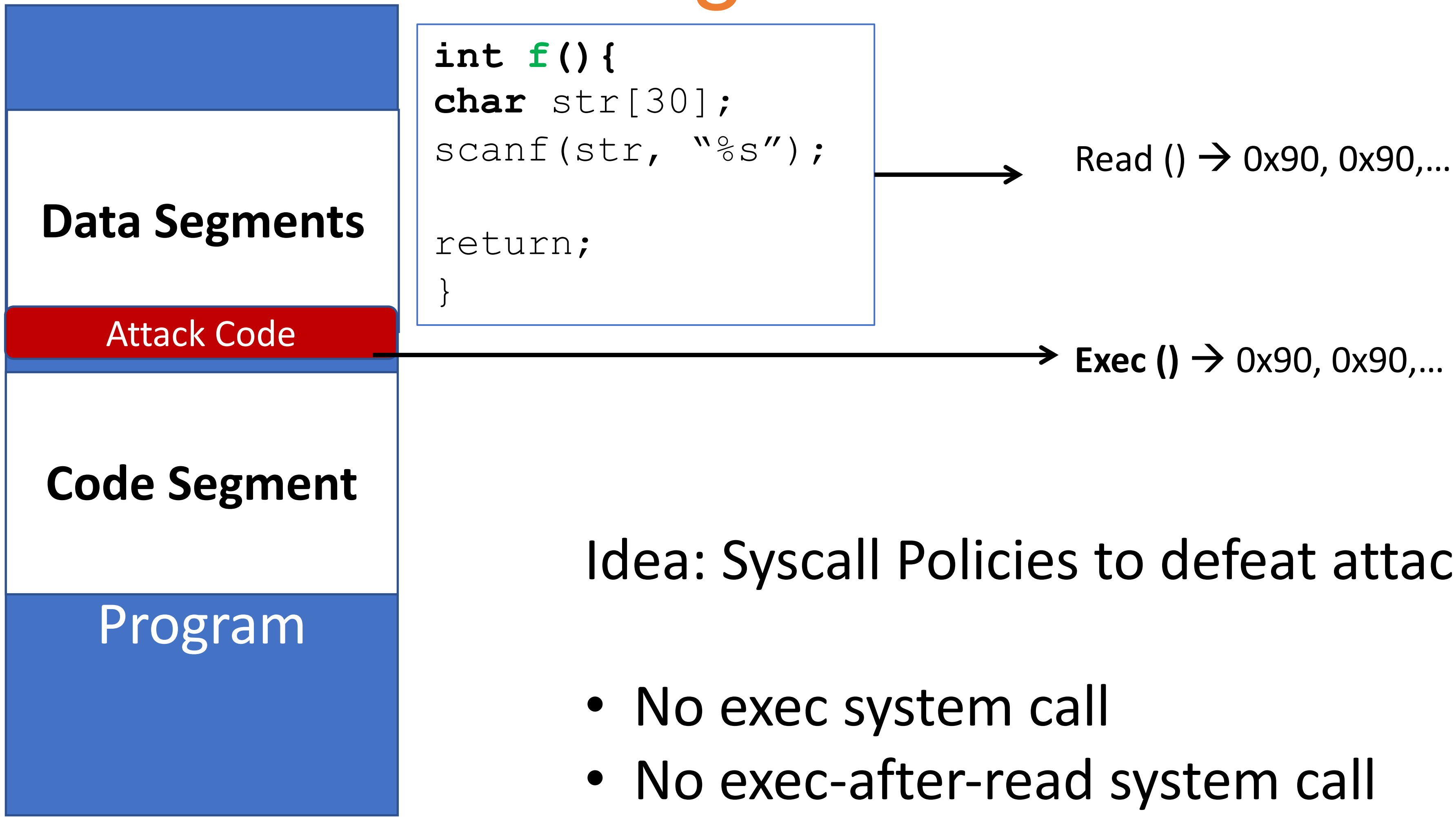
3 Security Principles

- Separation of Concerns:
  - Separate the **policy** from its **enforcement**
- Minimize Trusted Code Base (TCB)
  - Reduce what one needs to trust
  - Separate **verifier** from the **enforcement**
- Least Privilege
  - Give each component only the privileges necessary

Policy vs. Enforcement Mechanism

- Access Control Policies
- Enforcement:
  - Process sandboxing
  - Inline Reference Monitors
  - Virtualization
  - Hardware-based isolation / Trusted Execution Env.

Process Sandboxing



```
int f(){
char str[30];
scanf(str, "%s");

return;
}
```

Idea: Syscall Policies to defeat attacks

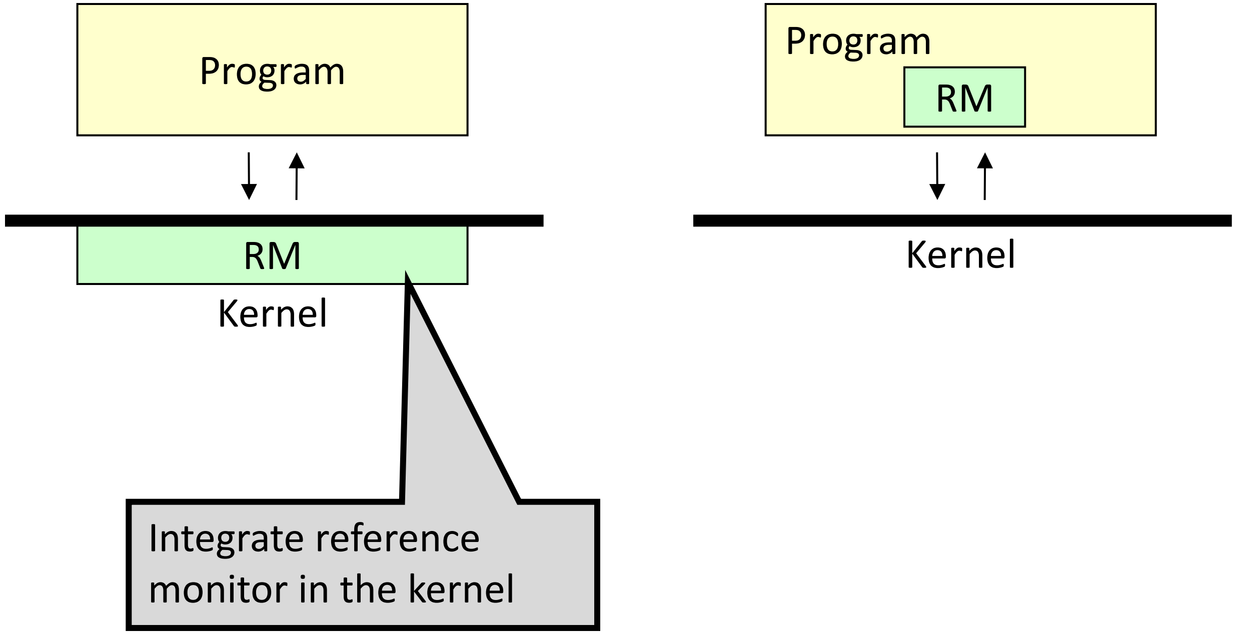
- No exec system call
- No exec-after-read system call

Enforcement Mechanisms: Process Isolation / Sandboxing

System Call Sandboxing

Reference Monitor: A piece of code that *checks all* **references** to an **object**

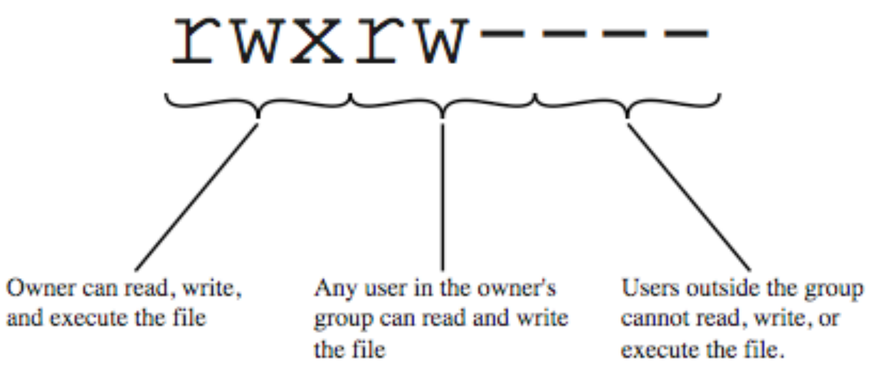
Syscall Sandbox: A reference monitor for protecting OS resource objects from an app



Integrate reference monitor in the kernel

Slide from Shmatikov et. al.

Kernelized Syscall Sandbox (I): Access Control Lists



rwxrw----

Owner can read, write, and execute the file

Any user in the owner's group can read and write the file

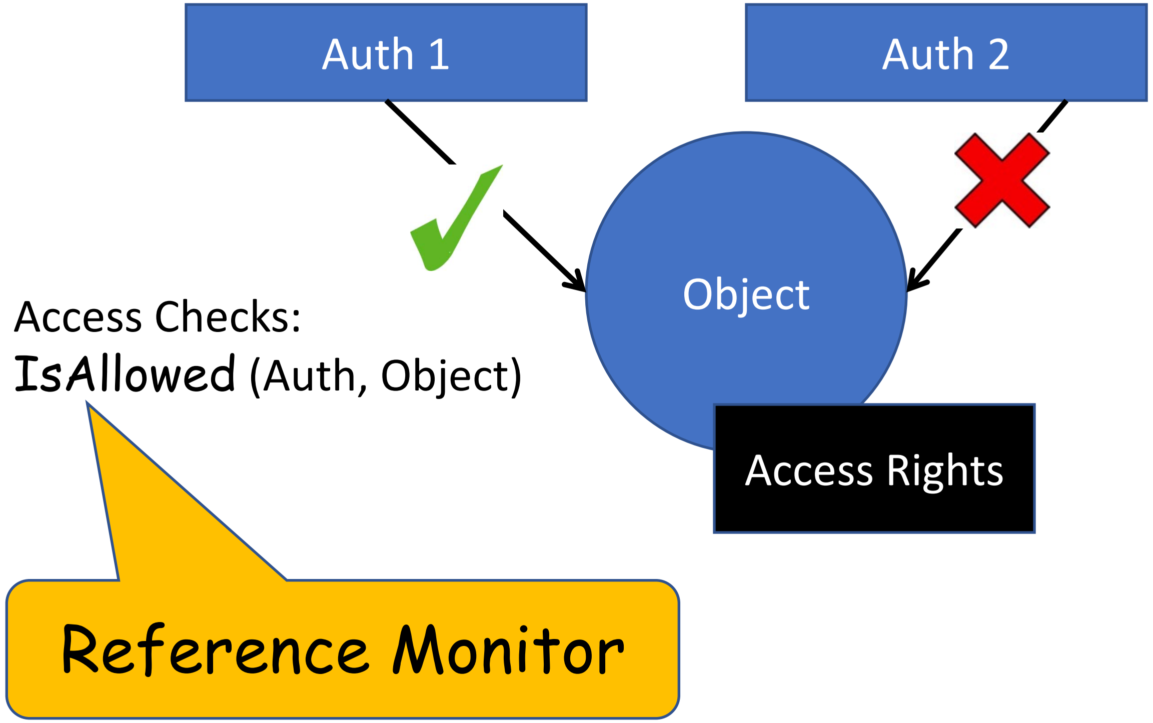
Users outside the group cannot read, write, or execute the file.

Access Control List

	BIBLIOG	TEMP	F	HELP.TXT	C COMP	LINKER	SYS CLOCK	PRINTER
USER A	ORW	ORW	ORW	R	X	X	R	W
USER B	R	-	-	R	X	X	R	W
USER S	RW	-	R	R	X	X	R	W
USER T	-	-	-	R	X	X	R	W
SYS_MGR	-	-	-	RW	OX	OX	ORW	O
USER_SVCS	-	-	-	O	X	X	R	W

Protection and access control in operating systems [Lampson'72]

Kernelized Syscall Sandbox (I): Access Control Lists



Auth 1

Auth 2

Object

Access Rights

Access Checks:
IsAllowed (Auth, Object)

Reference Monitor

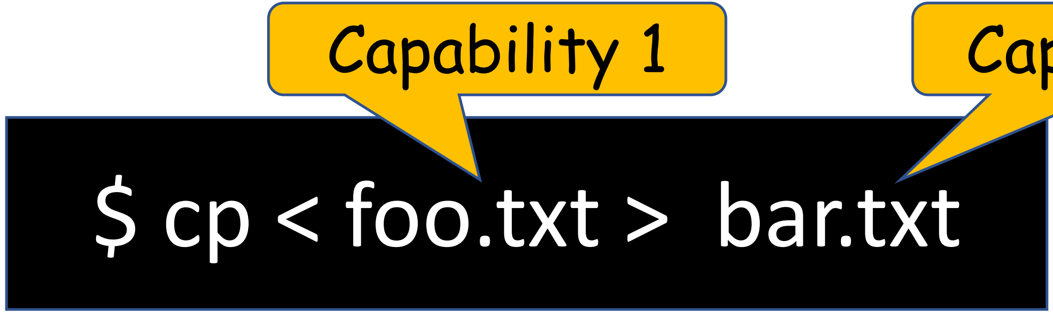
Challenge: Ambient Authority

```
$ cp foo.txt bar.txt
```

The “cp” program has authority to write to **any** file on the system.

This is not in line with “Principle of Least Privilege”

Kernelized Syscall Sandbox (II): Capabilities



Capability 1

Capability 2

```
$ cp < foo.txt >  bar.txt
```

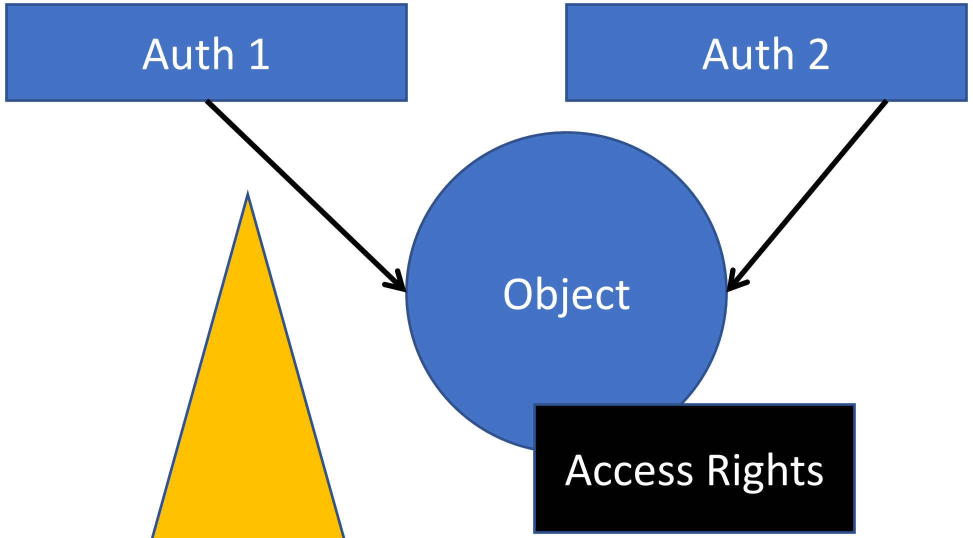
The “cp” program has no authority, by default.
It can only use “capabilities” it is given (e.g. UNIX file handles)

Definition of a **Capability**:
- An *identifier* which, when presented, provides certain access rights

Properties of a **Capability**:
- Unforgeable: Can’t manufacture without explicitly getting it.

Reference: First 20 minutes of Object Capabilities for Security

Kernelized Syscall Sandbox (II): Capabilities



No complex Access Control Checks

Access Control Lists vs. Capabilities

ACL

- Pros:
  - When the checks are simple and centralized, easier to implement ACL
  - Works well when rights change
- Cons:
  - Ambient Authority
  - Incomplete mediation:
    - Missing access control checks

Capabilities

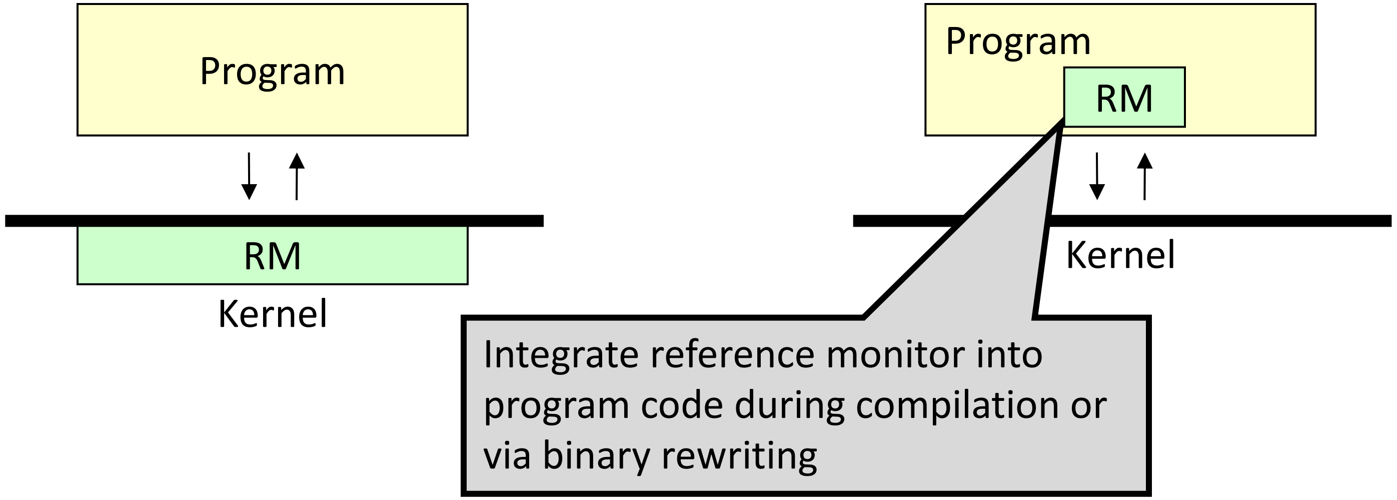
- Pros:
  - Eliminates access check logic
  - No pre-specification of who is allowed to access, i.e., can follow the natural flow of access rights
  - No ambient authority
    - Recall Least Privilege
- Cons:
  - Unsuitable when access rights change frequently
  - Capabilities can leak!

Inline Reference Monitors

Inline Reference Monitors

Reference Monitor: A piece of code that *checks all* **references** to an **object**

Syscall Sandbox: A reference monitor for protecting OS resource objects from an app



Integrate reference monitor into program code during compilation or via binary rewriting

Slide from Shmatikov et. al.

Inline Reference Monitors Can Check…

- Complete Memory Safety

  "Access memory objects in an intended way"
- Fault Isolation

  "Each module only accesses pre-determined data / code"
- No foreign code

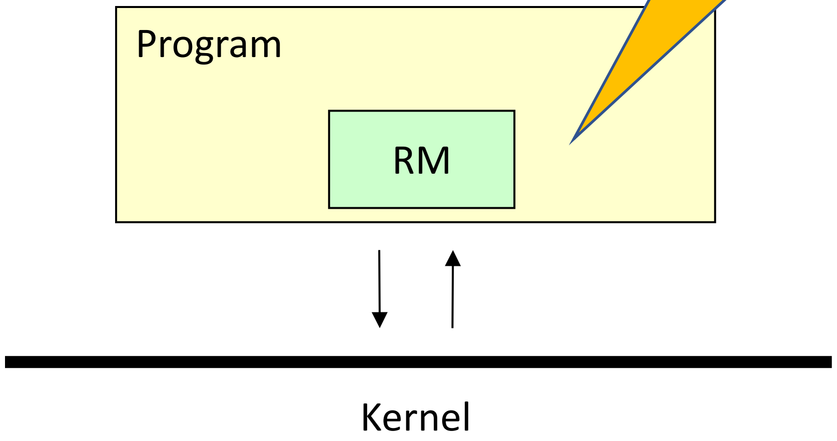
  "Execute only predetermined code"
- Control Flow Integrity

  "Control transfers are to legitimate points only"
- System Call Sandboxing

  "Access only a subset of system calls"
- (Code) Pointers / Data Integrity

  "Ensure (code) pointers / data have valid values"
- Data Flow Integrity…

Challenges in Inline / Wrapper-based Enforcement



Attacker & Checker in the same Privilege level

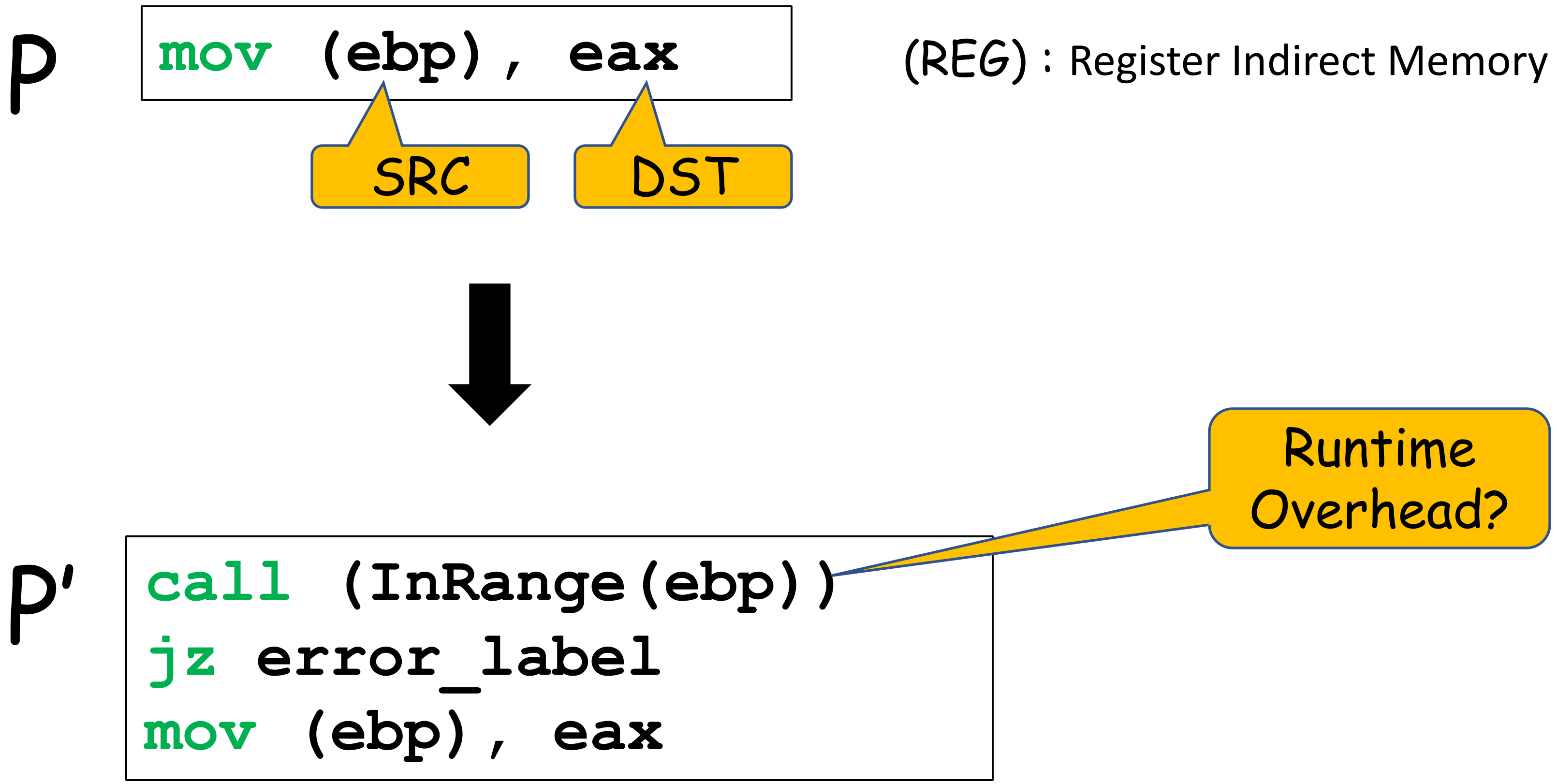
Inline Reference Monitors: Software Fault Isolation

Software Fault Isolation (SFI)

- Goal: Fault Isolation
  - Confine read/write to certain region M
  - This goal is also called “address sandboxing”

- Attacker controls all memory values in M
- Mechanism: Inline instrumentation of D
- Limit all memory accesses to region M
- Take an example: Let M be [0xbe00, 0xbeff]

Naïve SFI Implementation



P

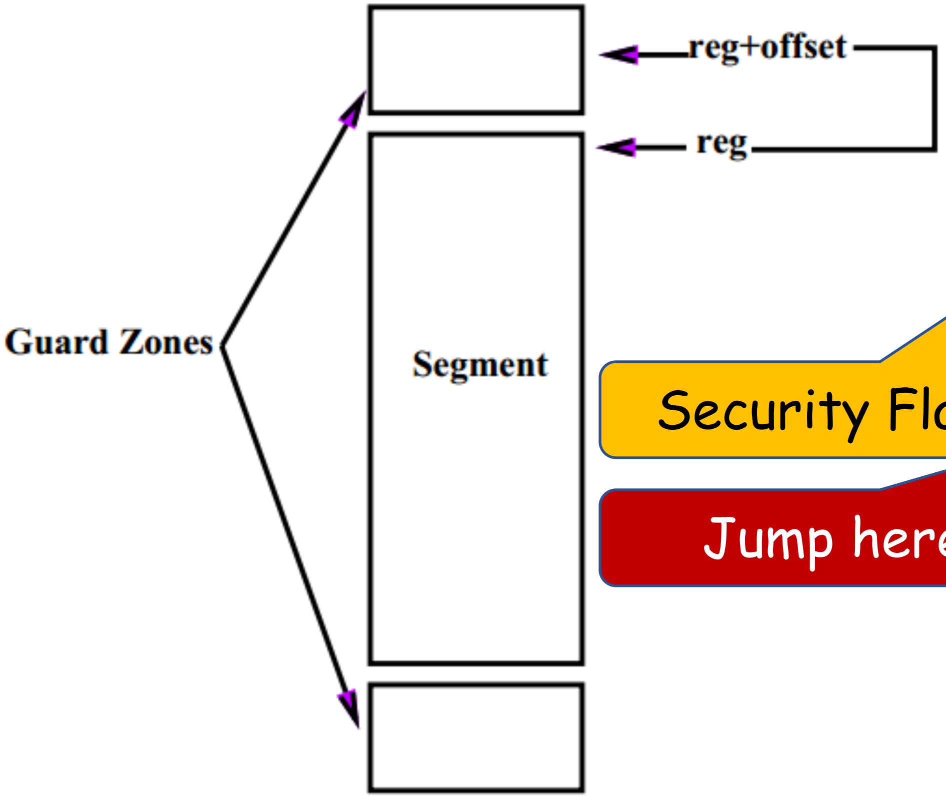
```
mov (ebp), eax
```

(REG) : Register Indirect Memory access

P'

```
call (InRange(ebp))
jz error_label
mov (ebp), eax
```

Fast SFI Implementation



```
mov  (ebp), eax
```

```
and  ebp, 0x00ff
or   ebp, 0xbe00
mov  (ebp), eax
```

Security Flaw?

Jump here!

```
and  reg1, 0x00ff
or   reg1, 0xbe00
mov  (reg1), eax
```

FIX: Using Dediated Regs.

Co-erce
Rather than Check

Efficient Software-based Fault Isolation [SOSP’93]

Verifying Correctness of Fast-SFI

1. Check if these IRM instructions exist before memory access
2. All memory accesses use the dedicated register
3. The dedicated registers are used only in IRM instructions

```
and reg1, 0x00ff
or  reg1, 0xbe00
mov (reg1), eax
```

Efficient Software-based Fault Isolation [SOSP’93]

3 Security Principles

- Separation of Concerns:
  - Separate the **policy** from its **enforcement**

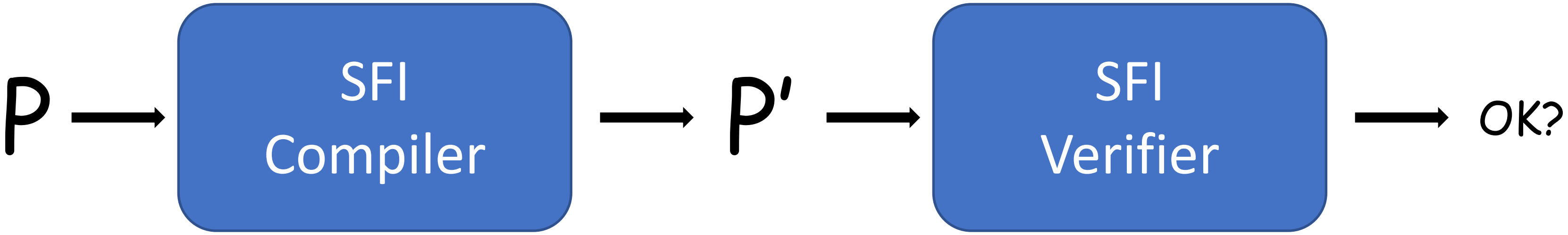
- Minimize Trusted Code Base (TCB)
  - Reduce what one needs to trust
  - Separate **verifier** from the **enforcement**

- Least Privilege
  - Give each component only the privileges necessary

SFI Has a Small TCB…

- Goal of Software Fault Isolation:
  - Address Sandboxing

    “ Access memory segments statically verified”



- Trusted Computing Base (TCB):

  “The trusted codebase for ensuring security properties”

- Smaller the TCB, the better the design

Aiding Syscall Sandboxing: Privilege Separation

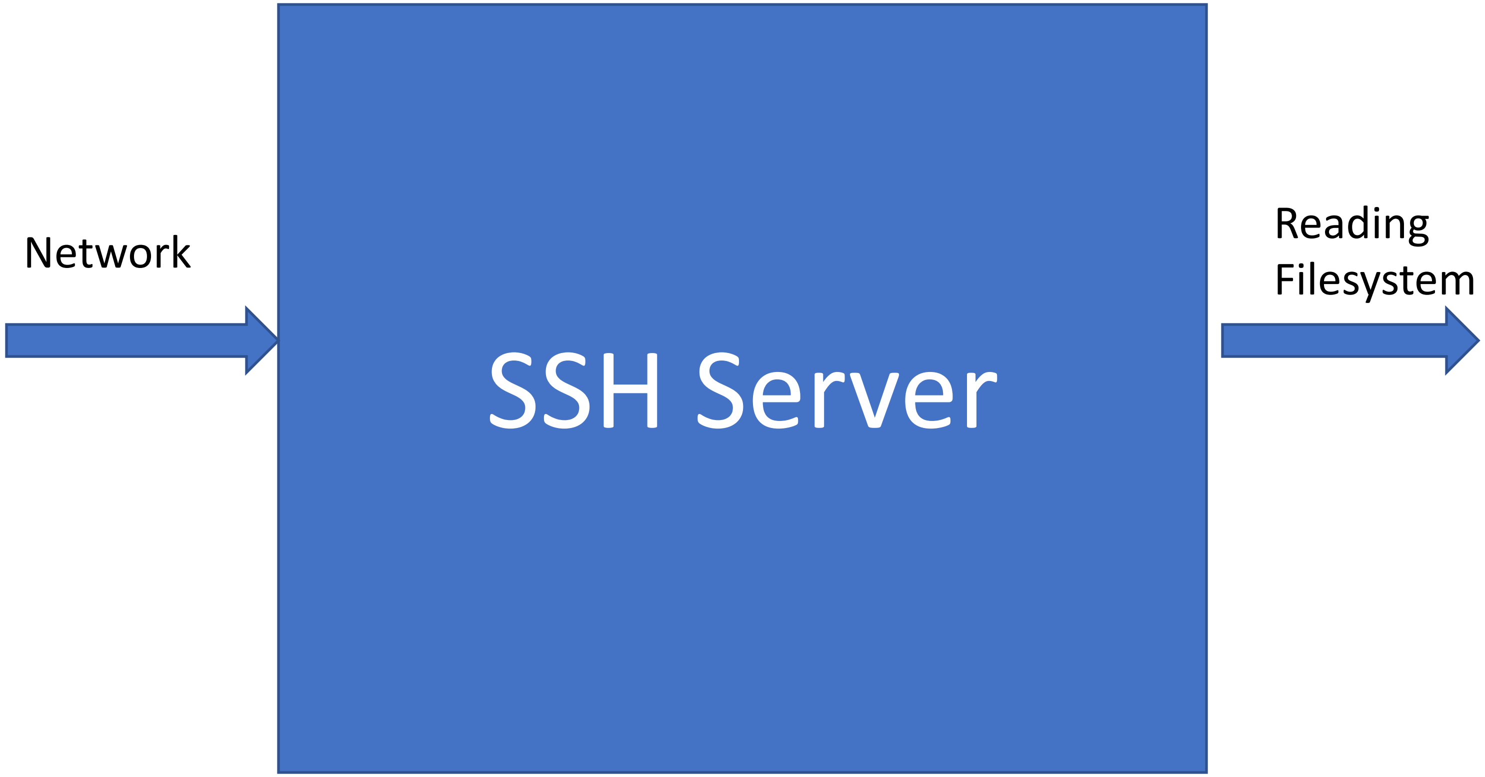
Takeaways: 3 Security Principles

- Separation of Concerns:
  - Separate the **policy** from its **enforcement**

- Minimize Trusted Code Base (TCB)
  - Reduce what one needs to trust
  - Separate **verifier** from the **enforcement**

- Least Privilege
  - Give each component only the privileges necessary

Problem:
Bundling of Functionality

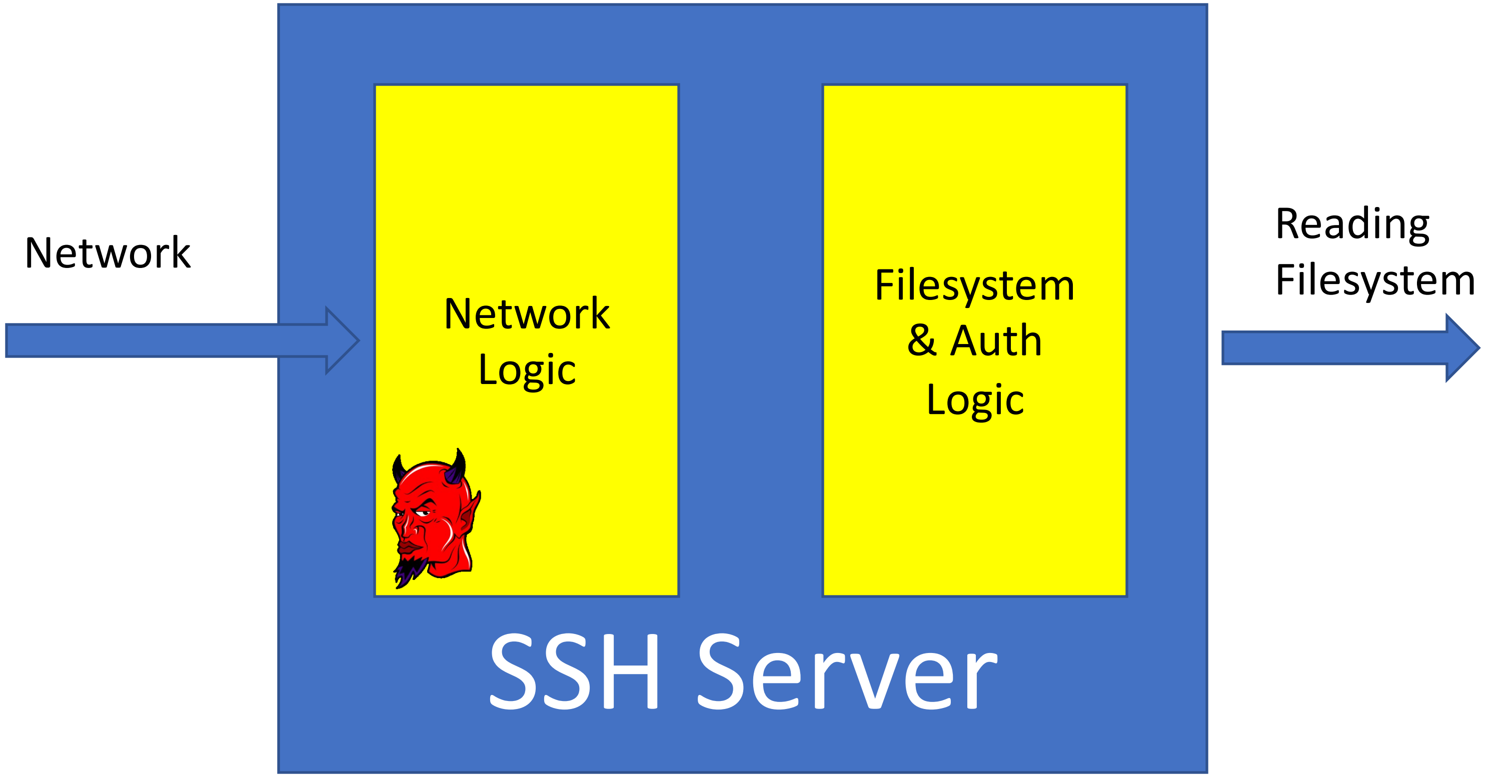


Network

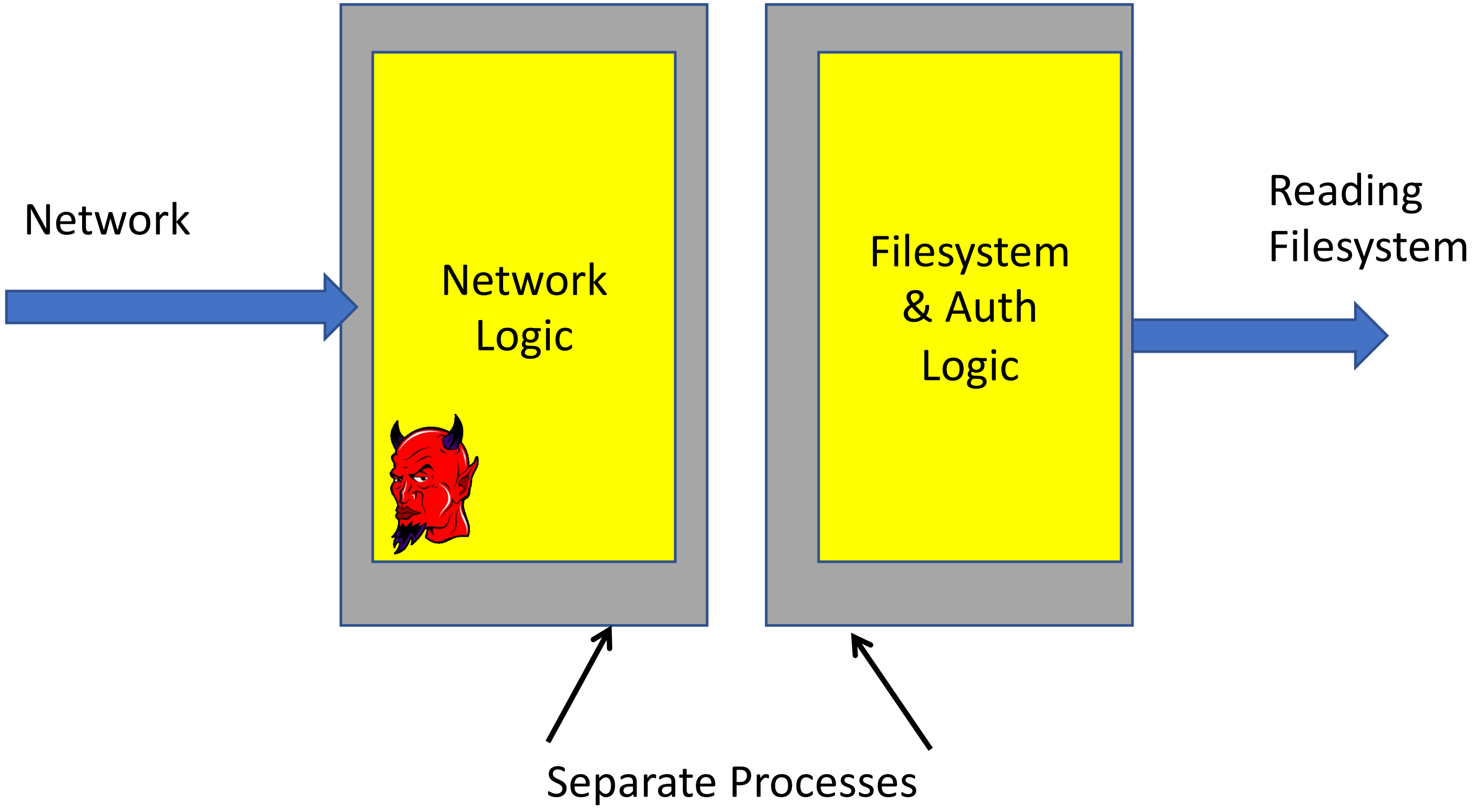
SSH Server

Reading Filesystem

Problem:
Bundling of Functionality



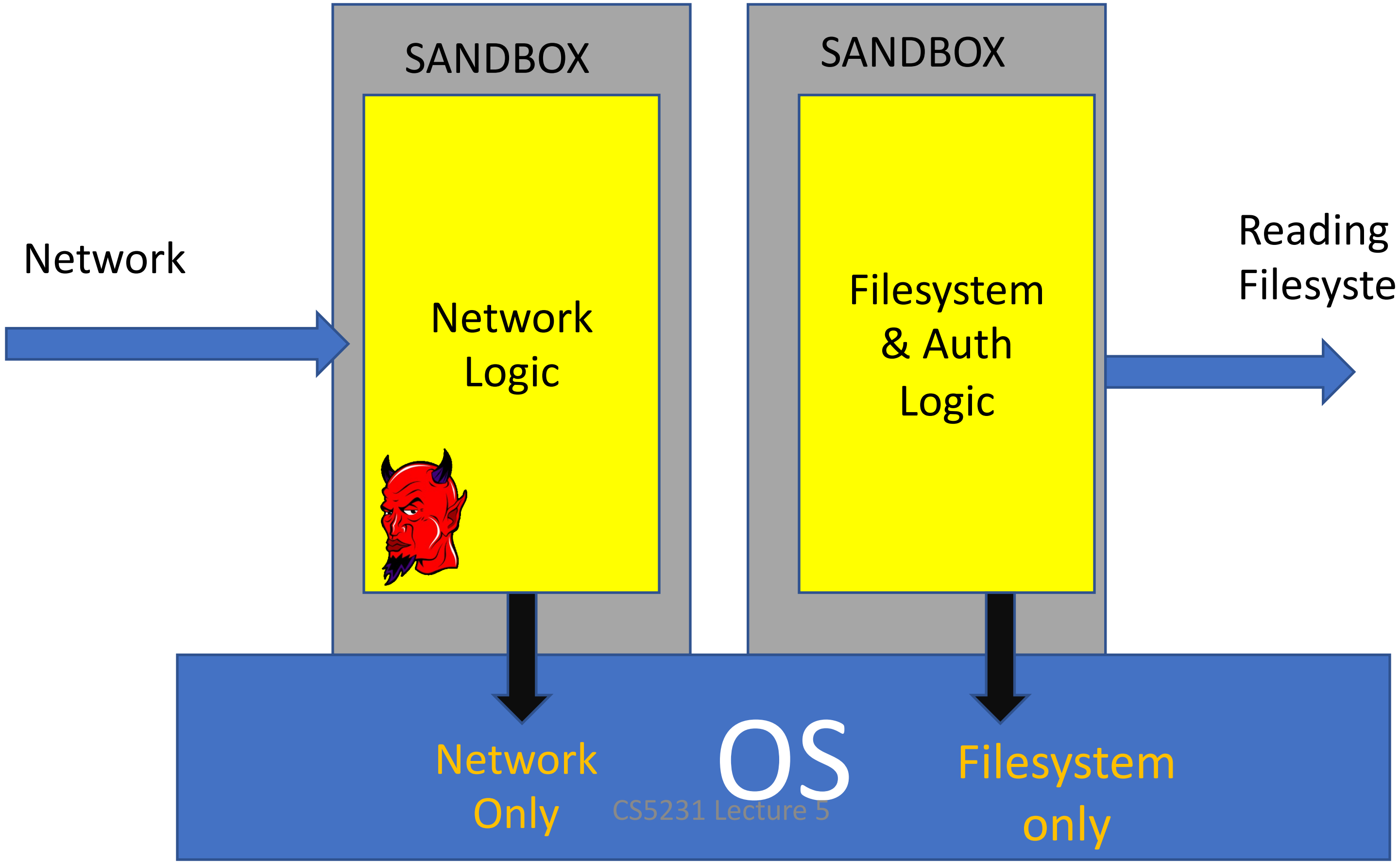
Solution: Privilege Separation



Courtesy: John Mitchell

Principle of Least Privilege

- Each compartment gets the least set of privileges it needs for its function

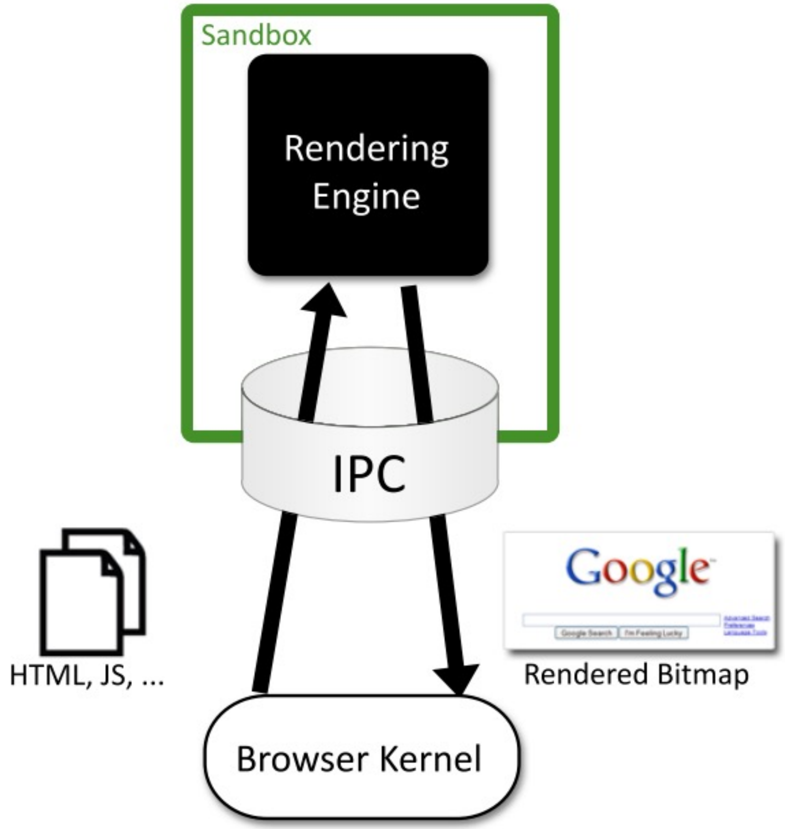


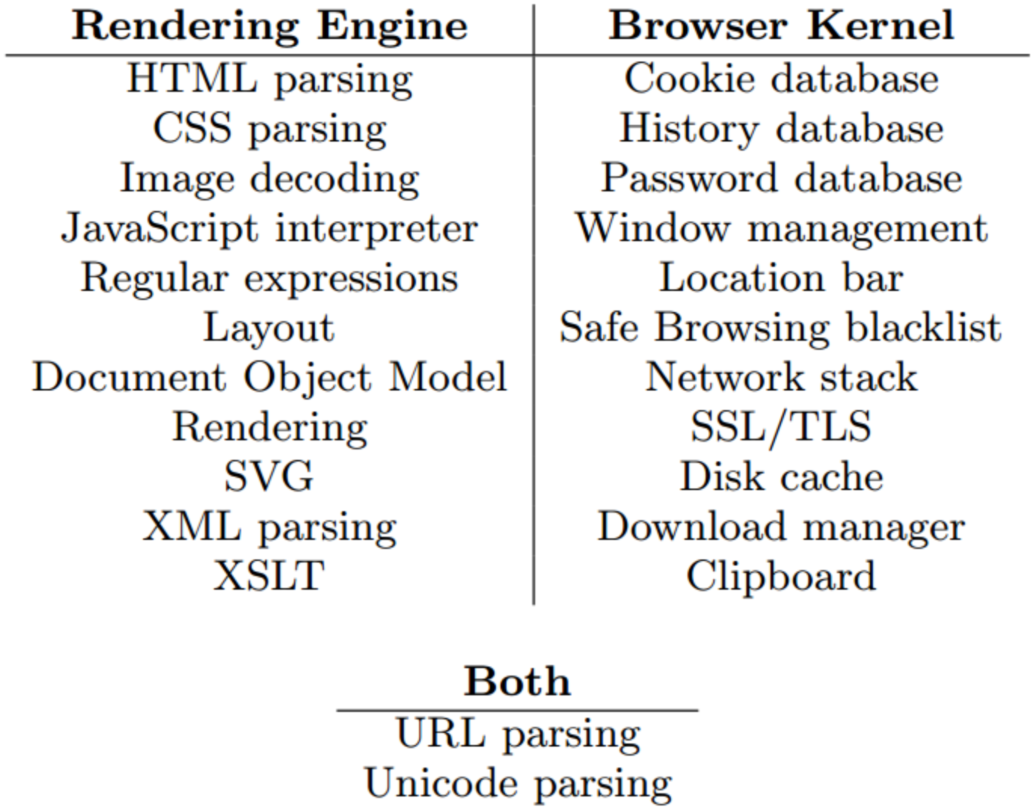
Design Browser With Isolation

- Problem with Old Browser Design (such as early Firefox): Single-process
  - Vulnerability leads to accessing all origins
- Solution: better Privilege Separation
  - Compartmentalize & assign least privilege
- Google Chrome
  - Goal: Separate filesystem from web code

Google Chrome Design

- Goal: Prevent web & network attacker from compromising OS resources (e.g. filesystem)





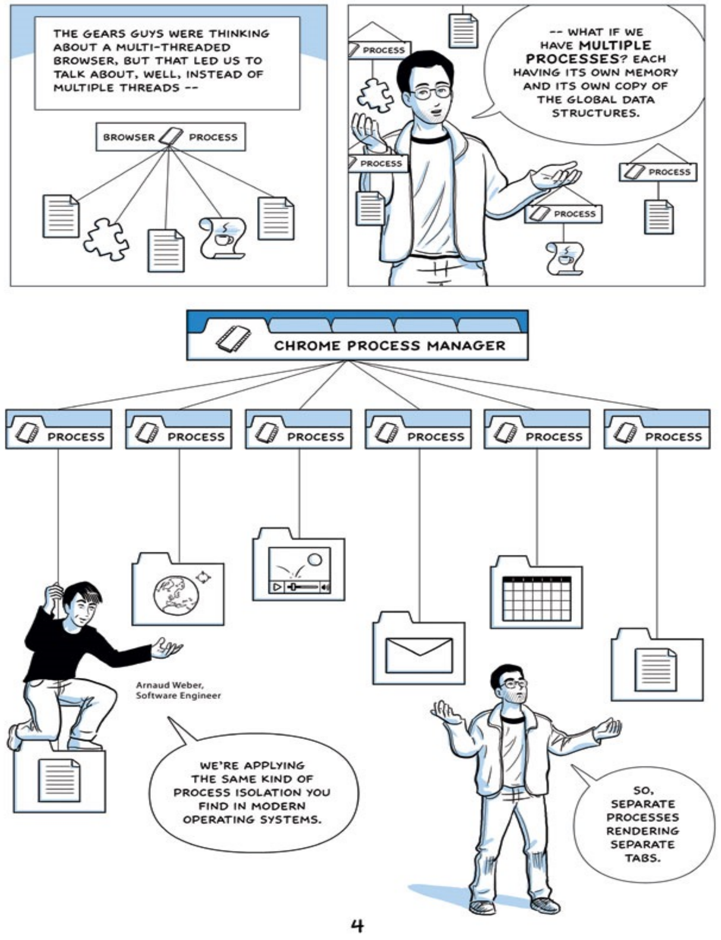
Rendering Engine	Browser Kernel
HTML parsing	Cookie database
CSS parsing	History database
Image decoding	Password database
JavaScript interpreter	Window management
Regular expressions	Location bar
Layout	Safe Browsing blacklist
Document Object Model	Network stack
Rendering	SSL/TLS
SVG	Disk cache
XML parsing	Download manager
XSLT	Clipboard

Both
URL parsing
Unicode parsing

The Security Architecture of the Chromium Browser

Google Chrome

- One excellent idea: Using OS mechanism to protect resources in browser
  - Run each tab in a separate process
  - Error in one tab won’t affect other tabs
- Read more: http://www.google.com/googlebooks/chrome/


THE GEARS GUYS WERE THINKING ABOUT A MULTI-THREADED BROWSER, BUT THAT LED US TO TALK ABOUT, WELL, INSTEAD OF MULTIPLE THREADS --
BROWSER PROCESS
-- WHAT IF WE HAVE MULTIPLE PROCESSES? EACH HAVING ITS OWN MEMORY AND ITS OWN COPY OF THE GLOBAL DATA STRUCTURES.
PROCESS
PROCESS
PROCESS
PROCESS
CHROME PROCESS MANAGER
PROCESS
PROCESS
PROCESS
PROCESS
PROCESS
PROCESS
Arnaud Weber, Software Engineer
WE'RE APPLYING THE SAME KIND OF PROCESS ISOLATION YOU FIND IN MODERN OPERATING SYSTEMS.
SO, SEPARATE PROCESSES RENDERING SEPARATE TABS.
4