COMP435: *SECURITY CONCEPTS!*

Lecture 13: Access Control + Capabilities

tinyurl.com/comp435-fa25

Quiz Topics

- MACs
- Cryptographic Hash Functions, Collison Resistance
- Race Conditions
- The Diffie-Hellman Key Exchange
- Secure System Design Principles

REVIEW: DESIGN PRINCIPLES

Design Principles (Saltzer and Schroeder)

- economy of mechanism
  - *keep it simple, easier to config & reason about*
- fail safe defaults
  - *If some security mechanism fails to execute in some way, the default situation is safe*
- complete mediation
  - *Every access to every object should be checked for the appropriate authority*
- open design
  - *The security of the system should not depend on the design being kept secret*
- separation of privilege
  - *Full permission to some access right to an object is never granted to a single subject*
- least privilege
  - *Every user and program should operate with the least set of privileges necessary to do its job*
- least common mechanism
  - *limit the mechanisms shared by and depended upon by multiple users. Each mechanism is a possible path for security critical information to flow between users.*
- psychological acceptability
  - *Interface must be intuitive and easy to use*

MAKING AUTHORIZATION DECISIONS

OS Responsibilities

- manage resources
- provide interface to HW
- provide user authentication
- mediate interprocess communication
- protect itself

All of these responsibilities are working in service of some ***security policy!***

OS Responsibilities

- manage resources
- provide interface to HW
- provide user authentication
- mediate interprocess communication
- protect itself

Ex: Process A is authorized to access memory region A, but not memory region B.

Access Control

Which users can access which resources and with which rights

Access Control

Who

How

Which users can access which resources and with which rights

What

Access Control

Subject

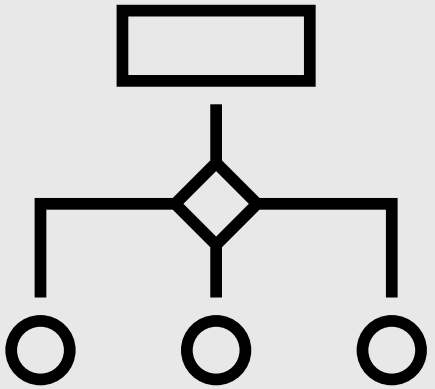
Right/Type

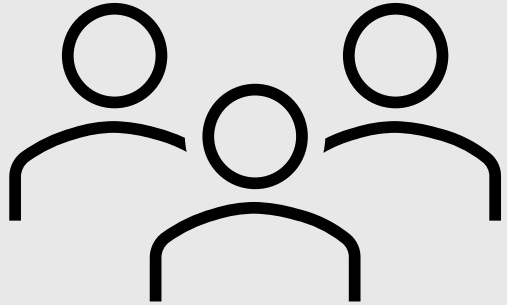
Which users can access which resources and with which rights

Object

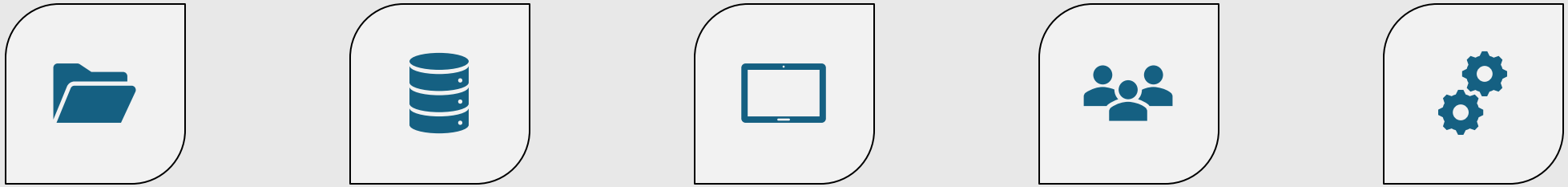
Subjects

- users
- processes





Objects



FILES MEMORY I/O DEVICES USERS PROCESSES

Objects

- files
- memory
- I/O devices
- users
- processes

subjects

Rights

- read
- write
- execute
- create
- transfer

BEST PRACTICES

Best Practice

- universal application
- least privilege
- type checking

Universal Application

Def’n: every access of an object by a subject should be checked

Ex: In practice: using cached results is a bad idea!

Also in practice… we use cached results all the time!

Universal Application

Def'n: every access of an object by a subject should be checked

Non-Universal Application

- random checking: randomly select a subset of accesses to check
- random auditing: log all accesses, then later randomly check some. Only provides detection, not prevention.
- selective checking: if a cheap-to-implement first glance seems suspicious, do a more thorough check

Least Privilege

Def'n: every subject should be granted the least amount of access necessary to do its job

Supports the idea of Failing Safe.

Limiting the amount of privilege a subject has, limits the damage they can do if they are compromised or turn out to be malicious.

Challenge: Need to know up front exactly what access each subject will need.

Result: Designers often over-privilege an entity (ex. A process) to ensure availability.

Type Checking

Def’n: operations should be meaningful for the object accessed

A type system, broadly speaking, defines the kinds of objects that will exist and the legal operations on those objects

MECHANISMS FOR MAKING AUTHORIZATION DECISIONS

Mechanisms for Making Authorization Decisions

	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

Capabilities

Is this process authorized to access this resource?

ACCESS CONTROL POLICIES

Access Control Policies

Discretionary Access Control (DAC)

Role-Based Access Control (RBAC)

Attribute-Based Access Control (ABAC)

Mandatory Access Control (MAC)

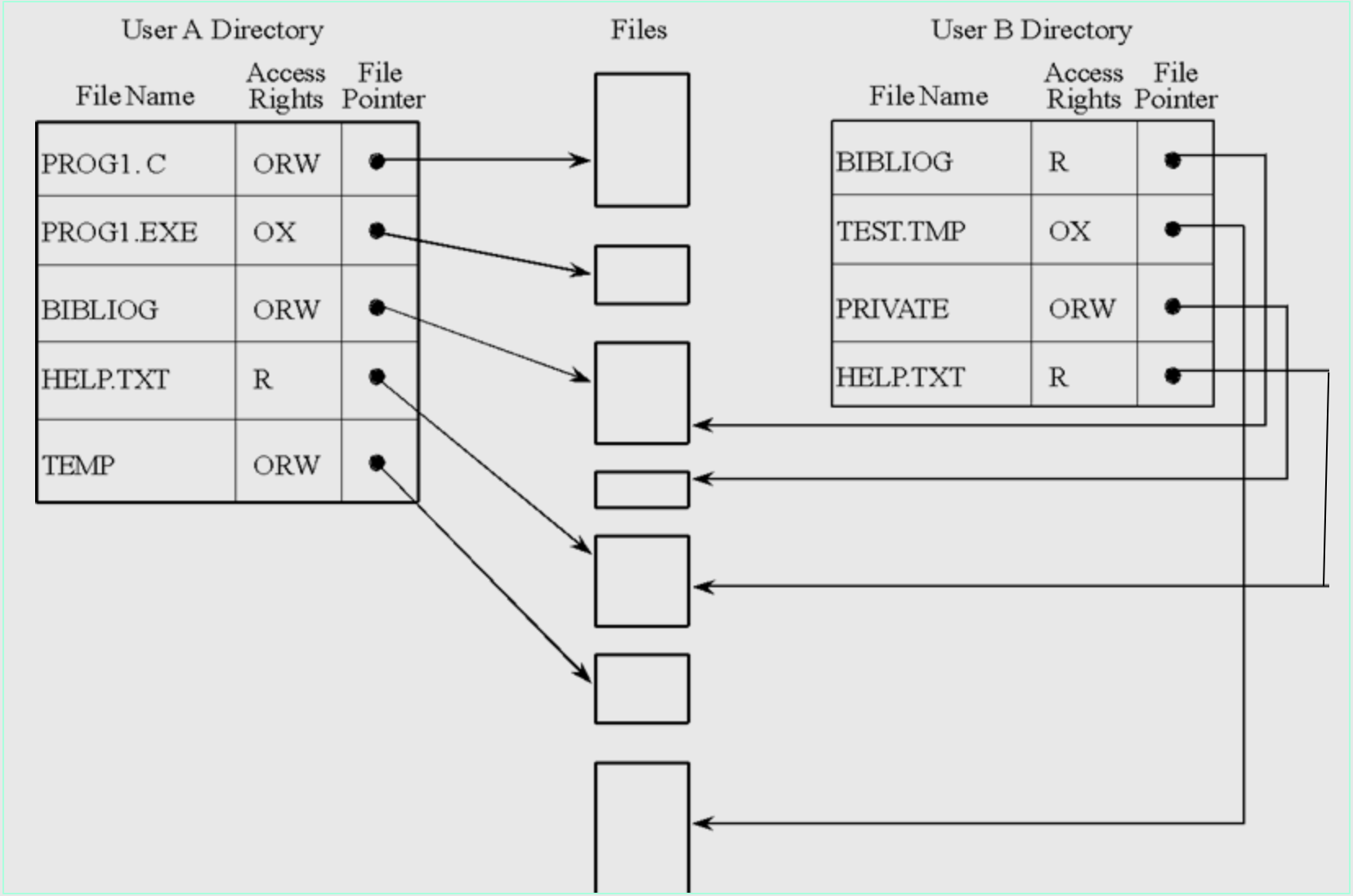
Access Control Matrix

Objects

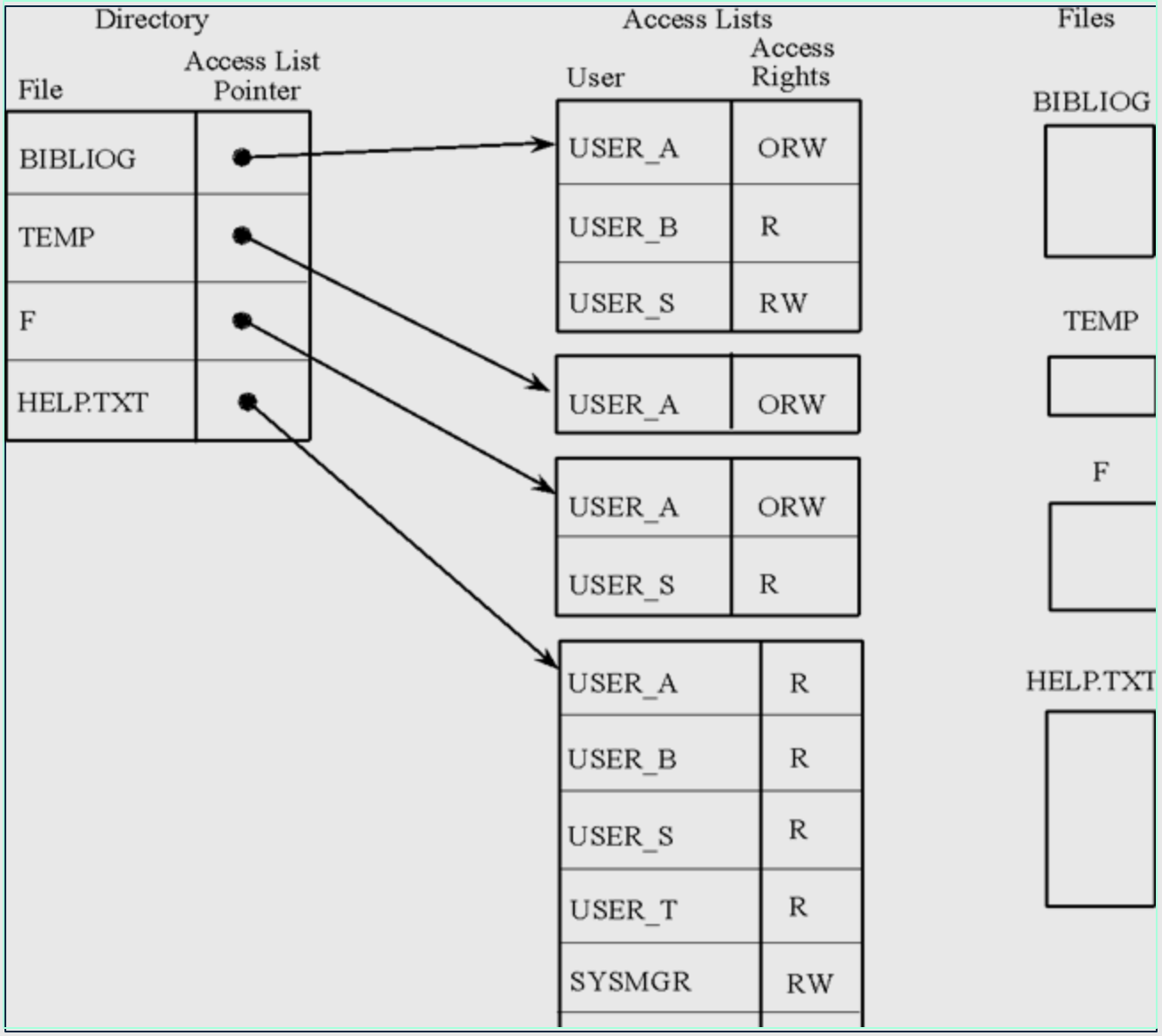
Subjects

	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 Directory



Access Control List



DAC Mechanisms

+ easy
+ revocation
+ no aliasing

- sparse matrix

Access Control Matrix

+ easy
+ fixes sparsity

- file revocation
- aliasing
- long lists

Access Control Directory

+ file revocation
+ no aliasing
+ default rights

- user revocation

Access Control List

Access Control Policies

Discretionary Access Control (DAC)

Role-Based Access Control (RBAC)

Attribute-Based Access Control (ABAC)

Mandatory Access Control (MAC)

Security Review with DAC

	topSecret.txt	printer	unrestricted.txt
Alice	RW	W	RW
Bob	--	W	R

Mandatory Access Control

- Goal: understand and control how information flows through a system
- Key Features:
  - *multi-level security (MLS) system*
  - *reference monitor*

Security Levels

top secret

secret

confidential

restricted

unclassified

high

low

Security Levels

top secret

secret

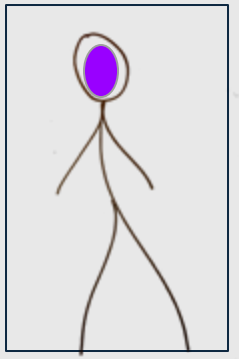
confidential

restricted

unclassified

high

low



Clearance: secret

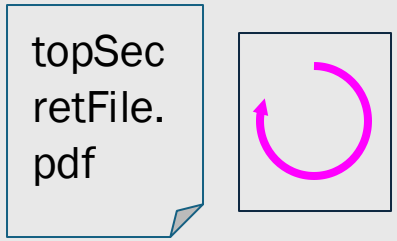
Classification: confidential

Mandatory Access Control Policies

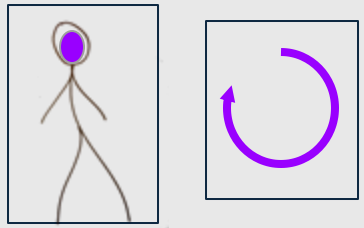
- Bell-LaPadula model
- Biba integrity model

Bell-LaPadula Model

- confidentiality
- no read up



classification: top secret

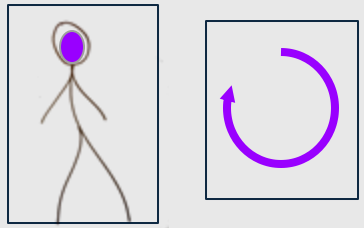


clearance: secret

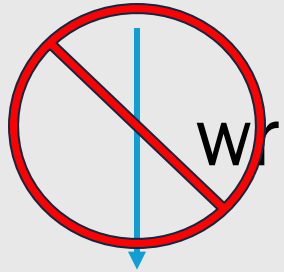
Bell-LaPadula Model

- confidentiality
- no read up
- no write down

classification: top secret



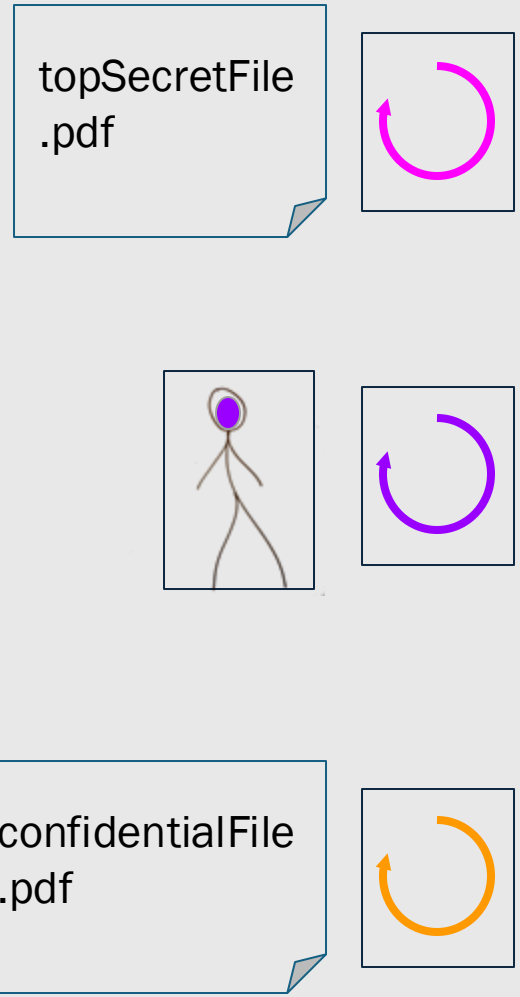
clearance: secret



classification: confidential

Biba Integrity Model

- integrity
- no write up
- no read down

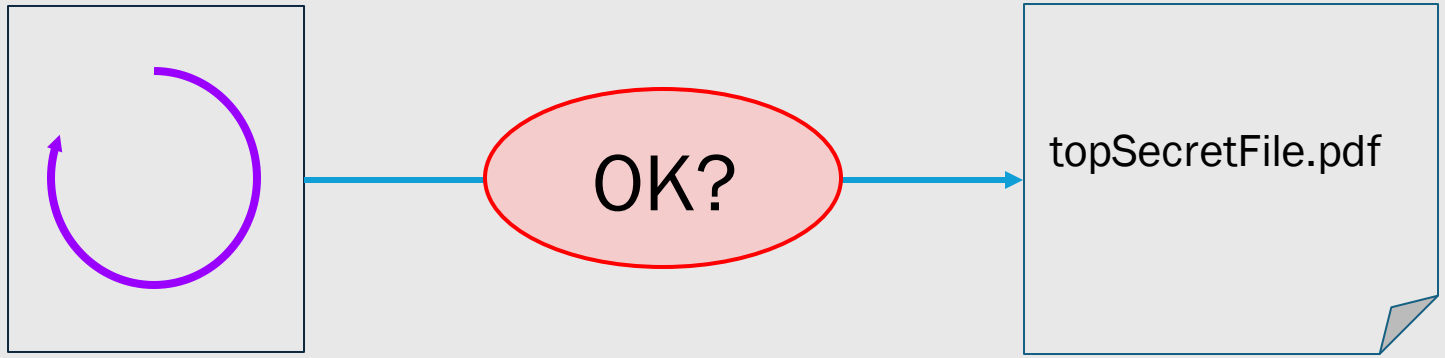


Enforcing the MAC Policy: Reference Monitor

- complete mediation
- tamperproof
- verifiable

Reference monitor: monitors execution and enforces the access control policy.

Complete mediation: every single access should pass through the reference monitor (also universal application!)

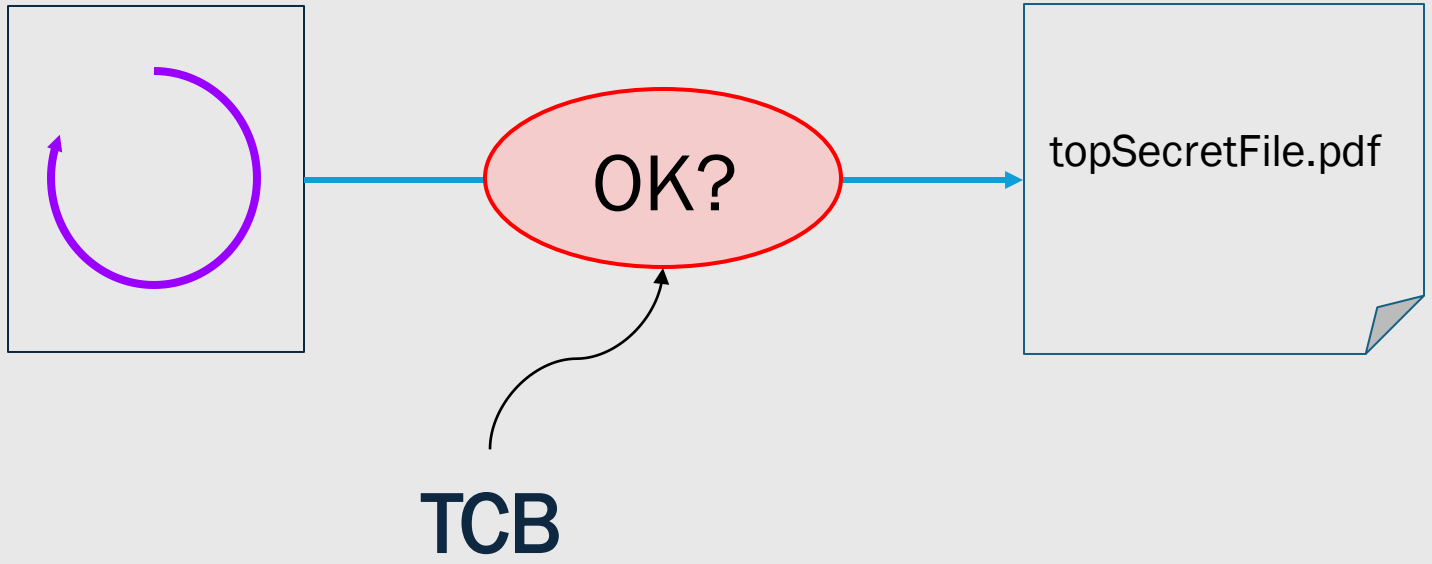


Enforcing the MAC Policy: Reference Monitor

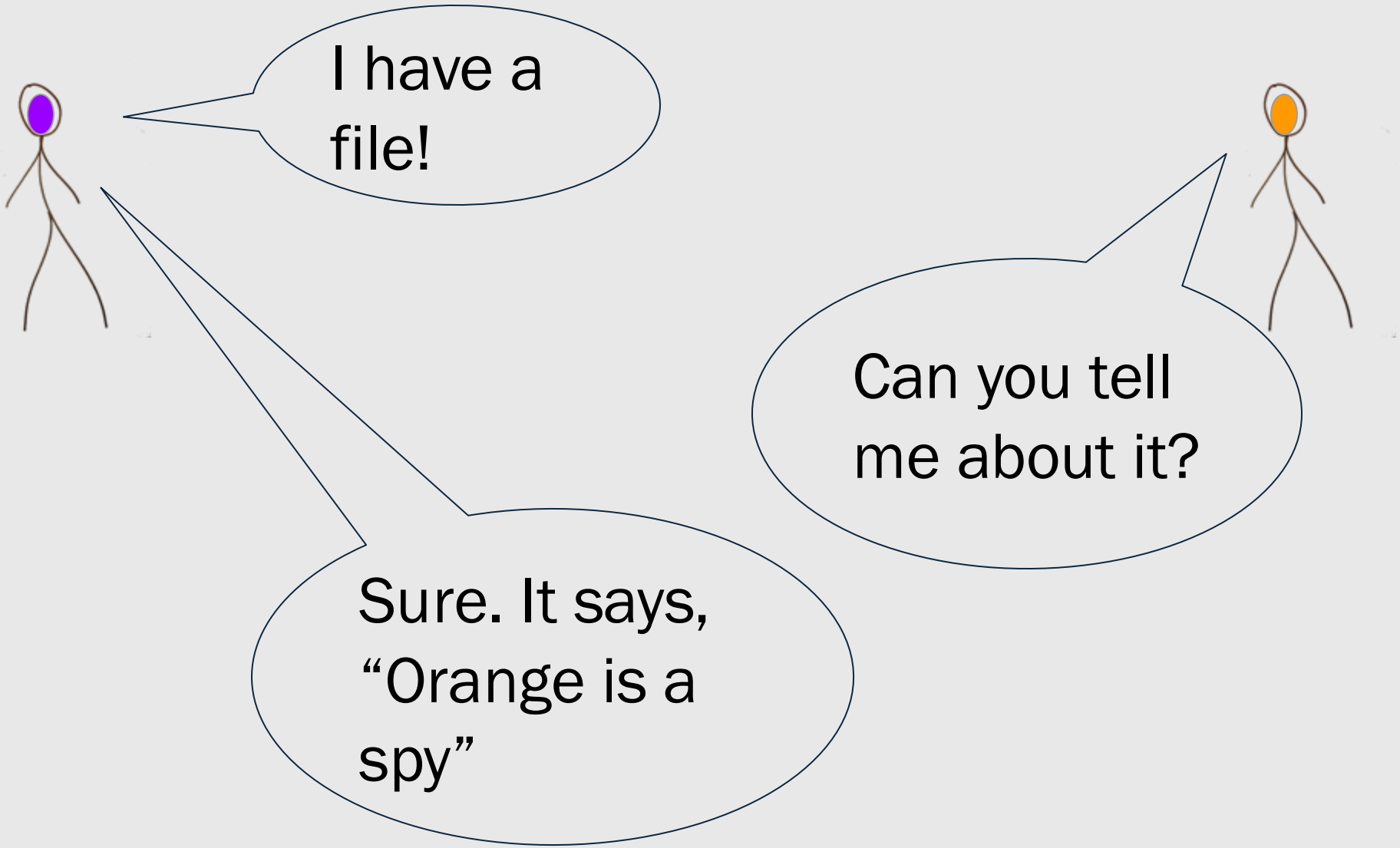
- complete mediation
- tamperproof
- verifiable

OSes don't use reference monitors these days, but a real-world example of a reference monitor is a firewall on a network.

All traffic coming from outside the network must pass through the firewall



The Analog Hole



WS Questions!

Clearance Level: Top Secret

secretFile.pdf

Security Classification: Secret

Clearance Level: Confidential

CAPABILITIES

Mechanisms for Making Authorization Decisions

	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

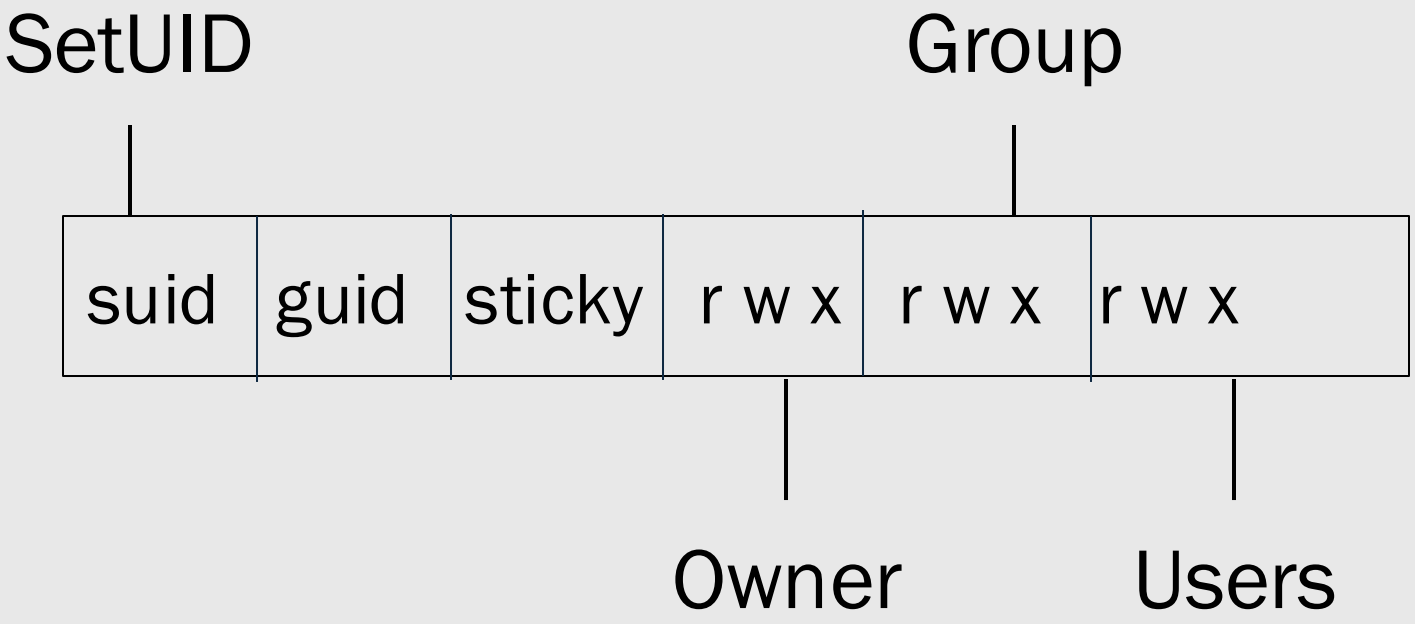
Capabilities

Is this process authorized to access this resource?

MOTIVATING CAPABILITIES: *THE CONFUSED DEPUTY ATTACK*

SetUID

When the setUID bit is set on an executable file and a user executes the file → the effective UID of that user switches to that of the creator of the executable.



myFile.exe

```
(1) int main(int argc, char *argv[]) {
    /*compile code*/

 //write out binary executable
(2) FILE *fp = fopen(argv[2], "w");
    /*write to fp*/

 //write out statistics
(3) fp = fopen("/etc/compiler_stats", "a");
    /*write to fp*/
 }
```

gcc code

```
(1) int main(int argc, char *argv[]) {
    /*compile code*/

 //write out binary executable
(2) FILE *fp = fopen(argv[2], "w");
    /*write to fp*/

 //write out statistics
(3) fp = fopen("/etc/compiler_stats", "a");
    /*write to fp*/
 }
```

```
$ gcc prog.c -o prog
```

```
(1) int main(int argc, char *argv[]) {
    /*compile code*/

 //write out binary executable
(2) FILE *fp = fopen(argv[2], "w");
    /*write to fp*/

 //write out statistics
(3) fp = fopen("/etc/compiler_stats", "a");
    /*write to fp*/
 }
```

```
$ gcc prog.c -o prog
$ gcc prog.c -o "/etc/passwd"
```

Ambient Authority

Def’n: all the extant authority of the current execution context

Problem: Authority can accrue as a program runs.

A given execution context will have multiple sets of permissions at any given time!

Confused Deputy Attack

A program running with multiple sets of permissions uses its ambient authority indiscriminately

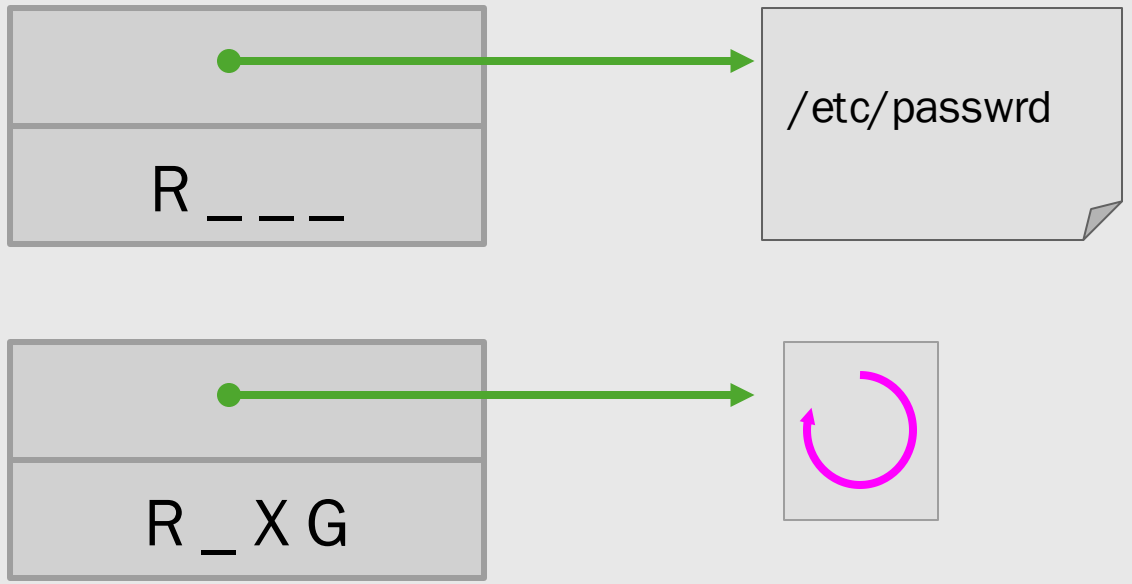
Confused Deputy Attack

A program running with multiple sets of permissions uses its ambient authority indiscriminately

Access Control cannot prevent the confused deputy attack

Capabilities

- object reference + access right
- “fat pointer”



```
(1) int main(int argc, char *argv[]) {
    /*compile code*/

 //write out binary executable
(2) cap *fcap = get_cap(argv[2], user_dir_cap);
(3) FILE *fp = cap_fopen(fcap);
    /*write to fp*/

//write out statistics
(4) fcap = get_cap("/etc/compiler_stats", sys_dir_cap);
(5) fp = cap_fopen(fcap);
    /*write to fp*/
 }
```

```
(1) int main(int argc, char *argv[]) {
    /*compile code*/

 //write out binary executable
(2) cap *fcap = get_cap(argv[2], use
(3) FILE *fp = cap_fopen(fcap);
    /*write to fp*/

//write out statistics
(4) fcap = get_cap("/etc/compiler_stats", sys_dir_cap);
(5) fp = cap_fopen(fcap);
    /*write to fp*/
 }
```

```
$ gcc prog.c -o prog
$ gcc prog.c -o "/etc/passwd"
> ERROR: no capability for
passwd file!
```

Capabilities

Properties:

- unforgeable token
- token directly ties access right to object
- no designation without authority

Provides:

- no ambient authority
- supports principle of least privilege (POLA in cap world)