

9/30/26

COMP 435:



Computer Security Concepts

Lecture 17: Access Control, Capabilities



The University
of North Carolina
at Chapel Hill

<https://tinyurl.com/comp435-fa26>

Plan for Class Today



- 10:10-10:35 – Access Control
- 10:35-11:00 – Capabilities

Guiding Question 😊

How does the operating system create isolation and manage access?



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

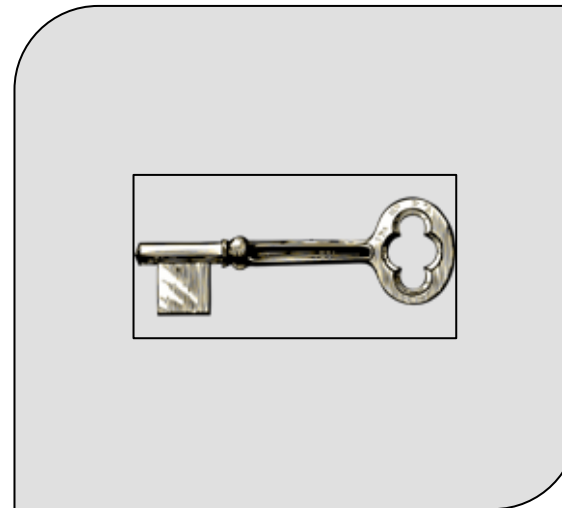
What

Mechanisms for Making Authorization Decisions



	BIBLIOG	TEMP	F	HELPTXT	C_COMP	LINKER	SYS_CLOCK	PRINTER
USERA	ORW	ORW	ORW	R	X	X	R	W
USERB	R	-	-	R	X	X	R	W
USERS	RW	-	R	R	X	X	R	W
USERT	-	-	-	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)~~

- matrix X
- d's
- list

Access Control Matrix

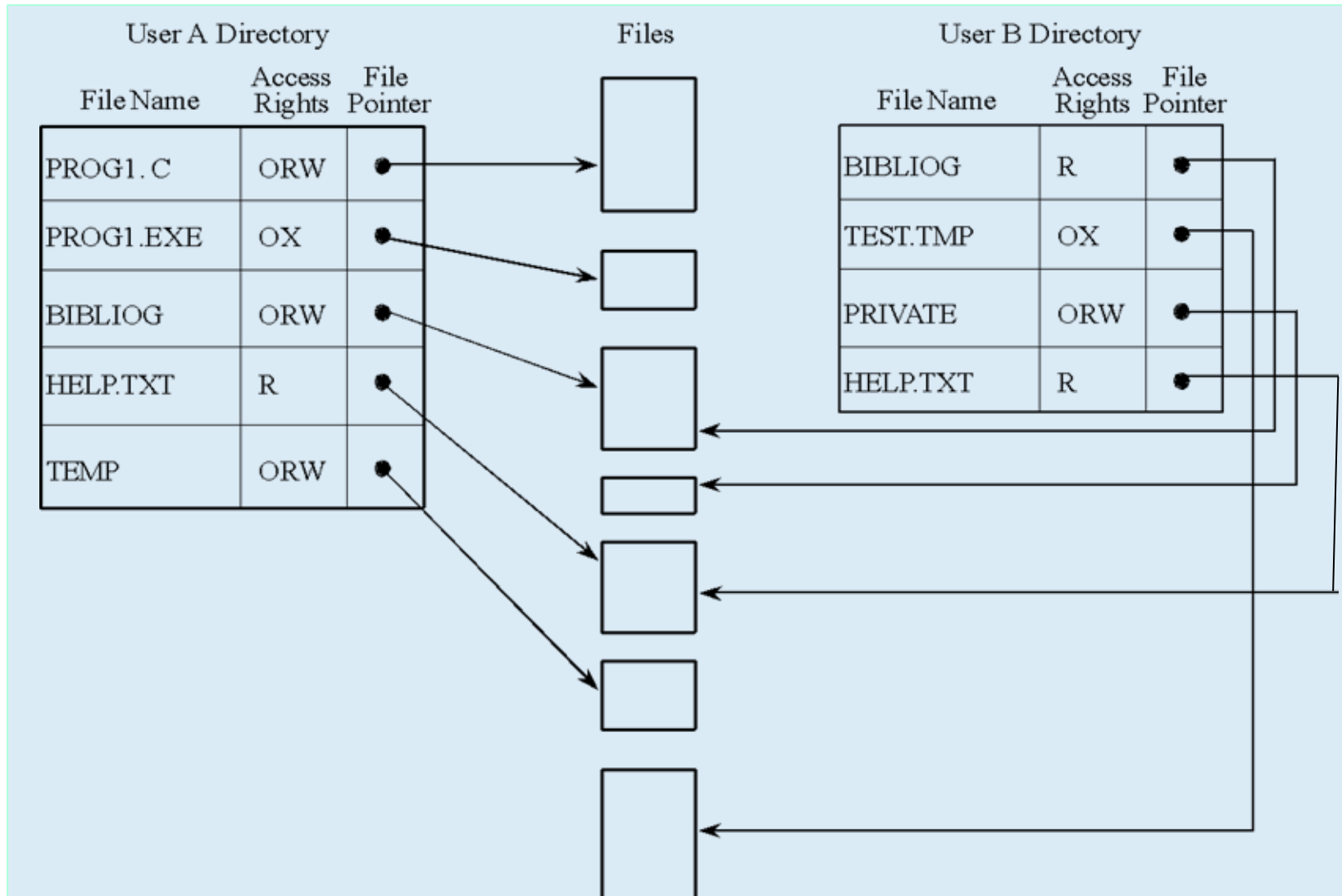


Objects

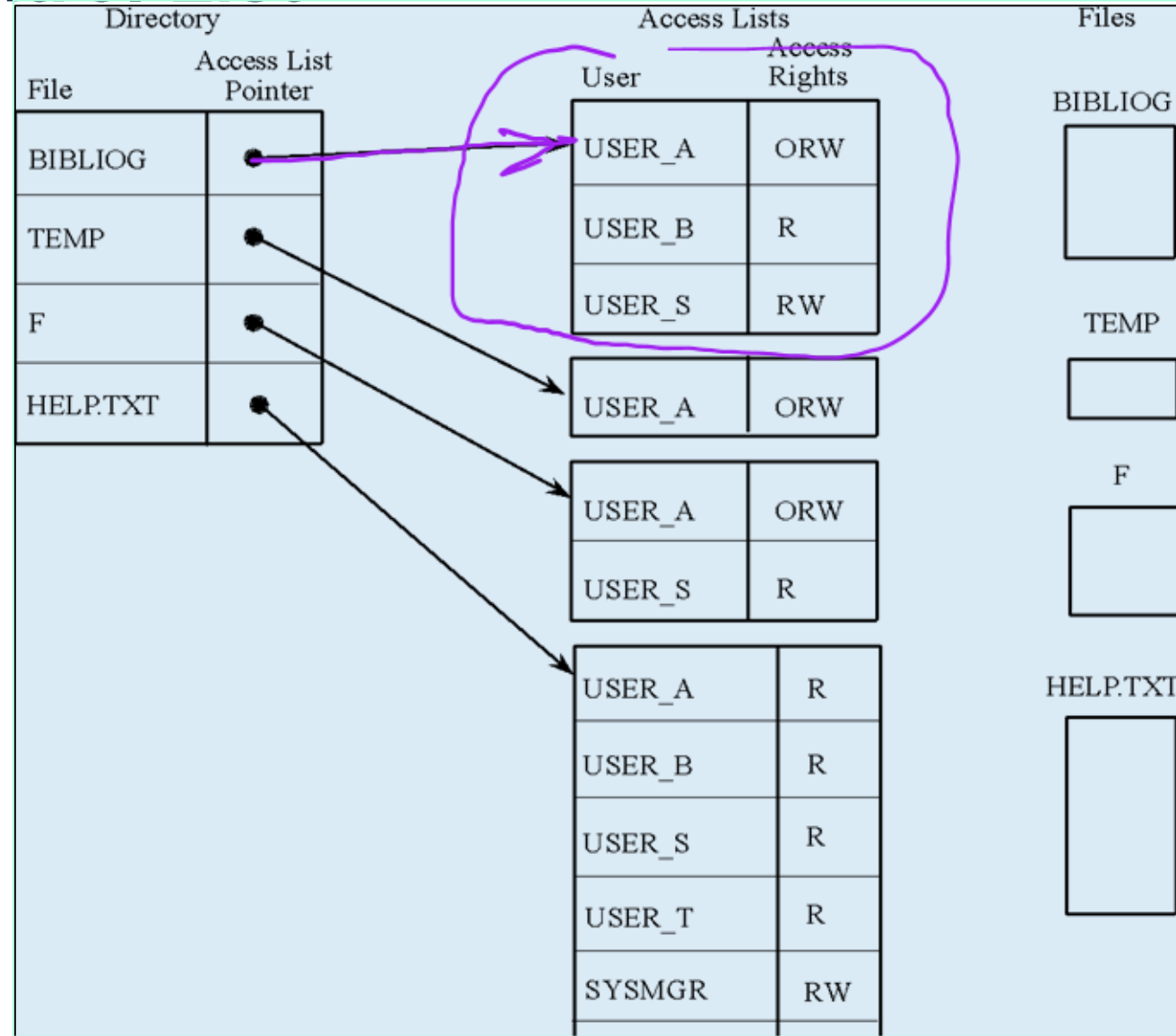
Subjects

	BIBLIOG	TEMP	F	HELP.TXT	C_COMP	LINKER	SYS_CLOCK	PRINTER
USER A	<u>ORW</u>	<u>ORW</u>	<u>ORW</u>	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

Security Review with DAC (WS Q2)



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

Access Control Policies



Discretionary
Access
Control (DAC)

Role-Based
Access
Control
(RBAC)

Attribute-Based
Access
Control
(ABAC)

Mandatory
Access
Control (MAC)



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

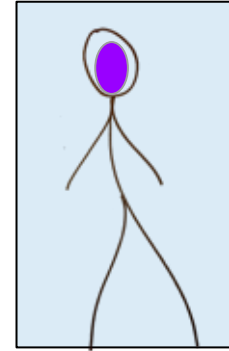
restricted

unclassified

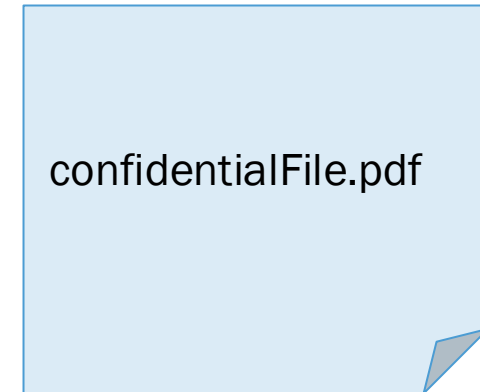


high

low



Clearance: secret



Classification: confidential

Mandatory Access Control Policies



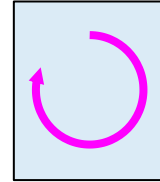
- Bell-LaPadula model
- Biba integrity model

Bell-LaPadula Model

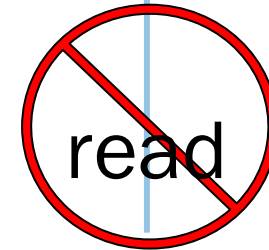


- confidentiality
- no read up

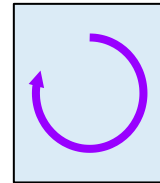
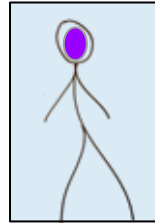
topSec
retFile.
pdf



classification: top secret



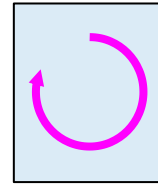
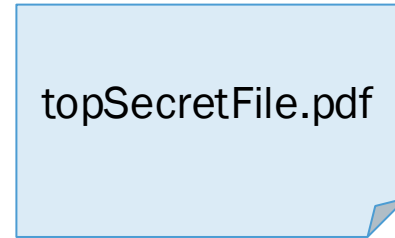
clearance: secret



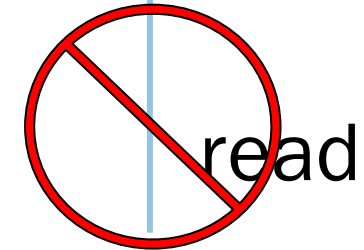
Bell-LaPadula Model



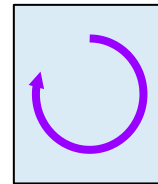
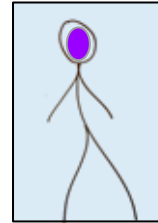
- confidentiality
- no read up
- no write down



classification: top secret



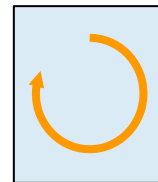
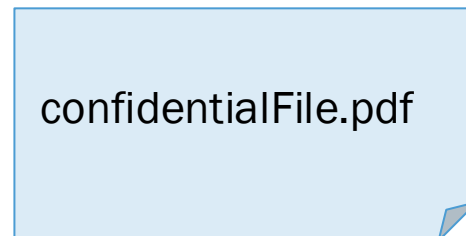
read



clearance: secret



write

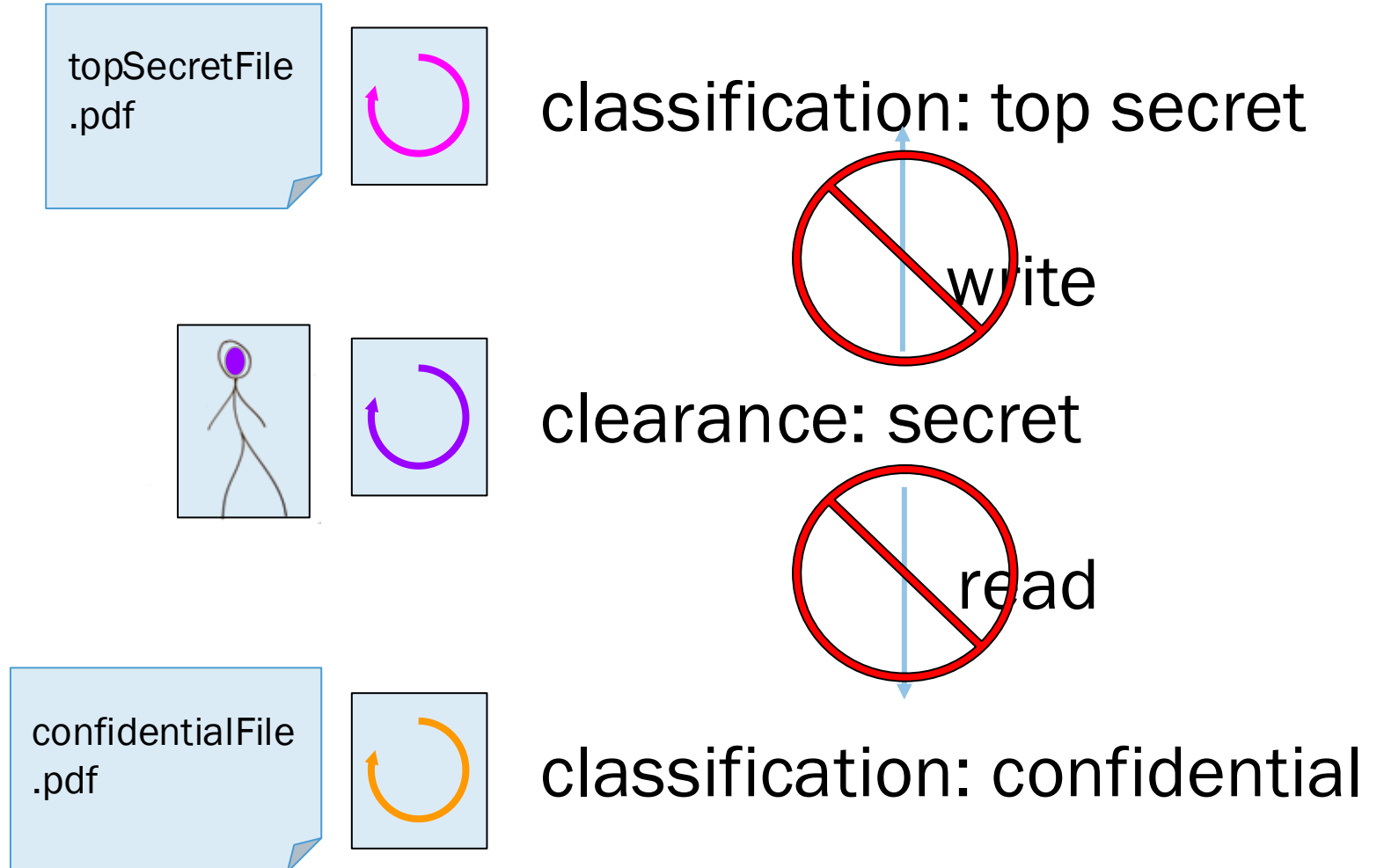


classification: confidential

Biba Integrity Model



- integrity
- no write up
- no read down



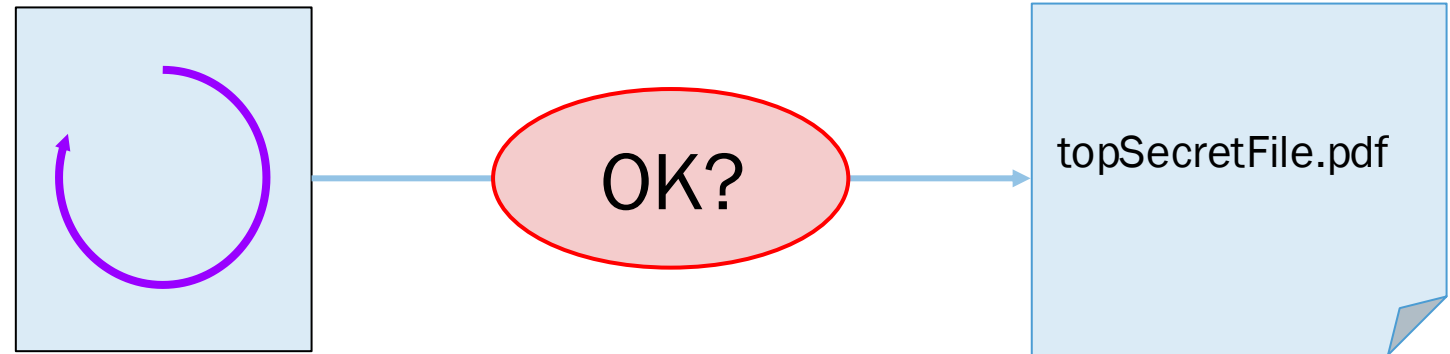
Enforcing the MAC Policy: Reference Monitor



- 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!)



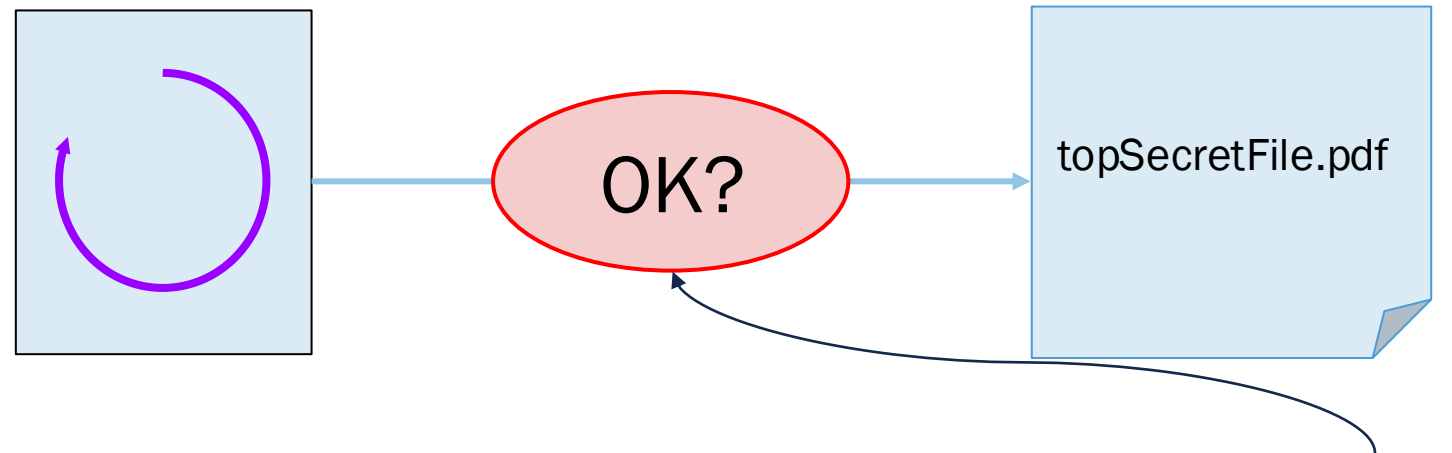
Enforcing the MAC Policy: Reference Monitor



- complete mediation
- tamperproof
- verifiable

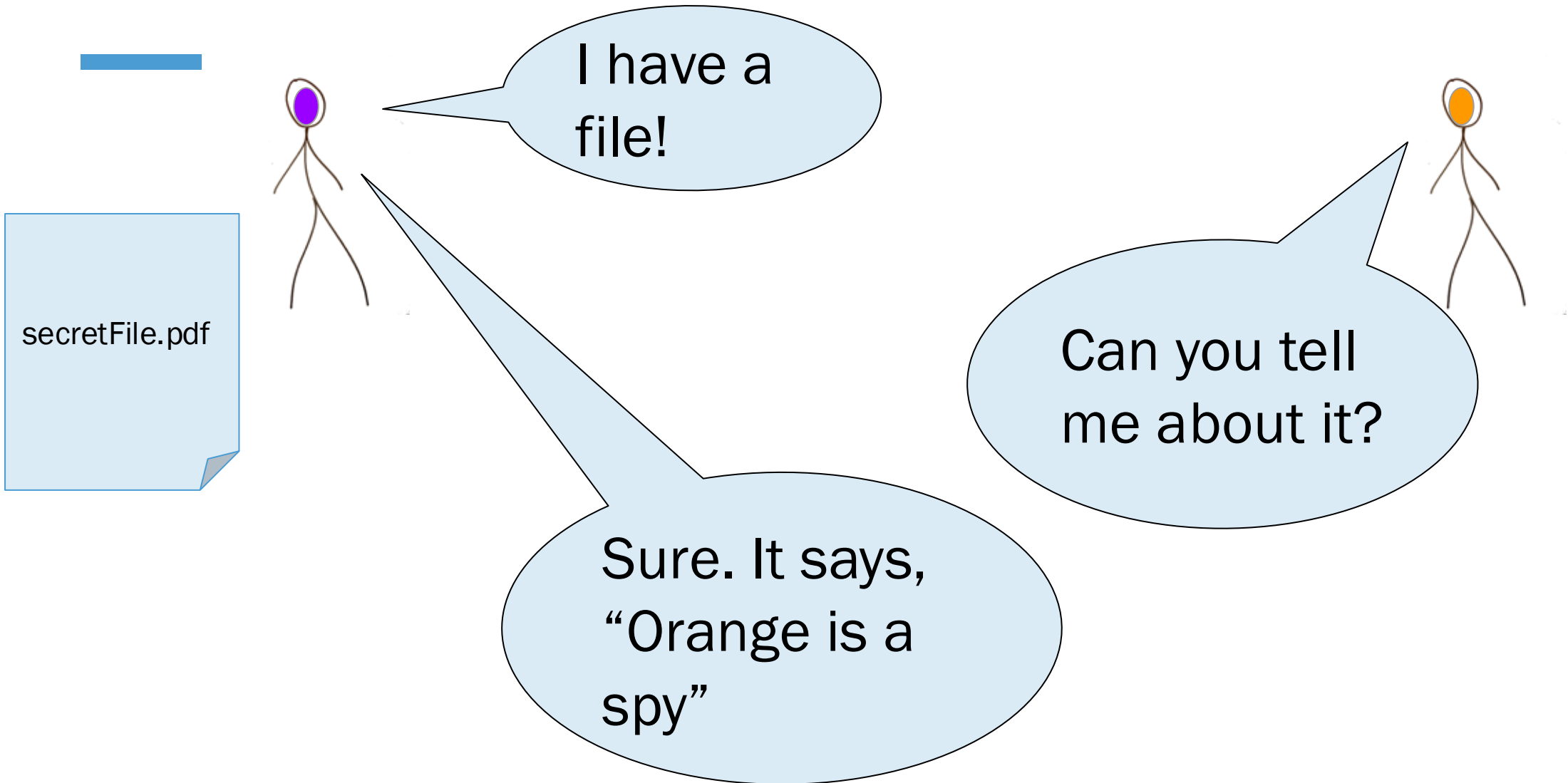
OSes don't use reference monitors these days, but a real-world example of a reference monitor is a firewall on a network.

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



TCB

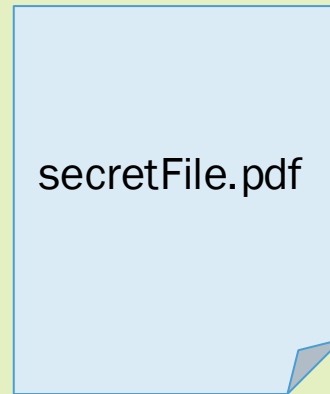
The Analog Hole



WS Questions!



Clearance
Level:
Top Secret



Security
Classification:
Secret



Clearance
Level:
Confidential



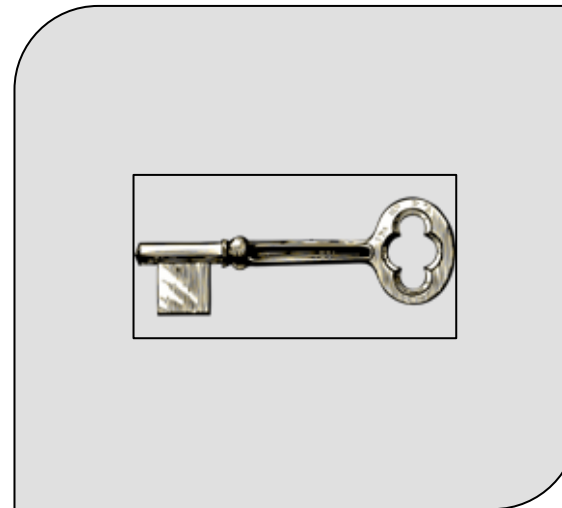
Capabilities

Mechanisms for Making Authorization Decisions



	BIBLIOG	TEMP	F	HELPTXT	C_COMP	LINKER	SYS_CLOCK	PRINTER
USERA	ORW	ORW	ORW	R	X	X	R	W
USERB	R	-	-	R	X	X	R	W
USERS	RW	-	R	R	X	X	R	W
USERT	-	-	-	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?



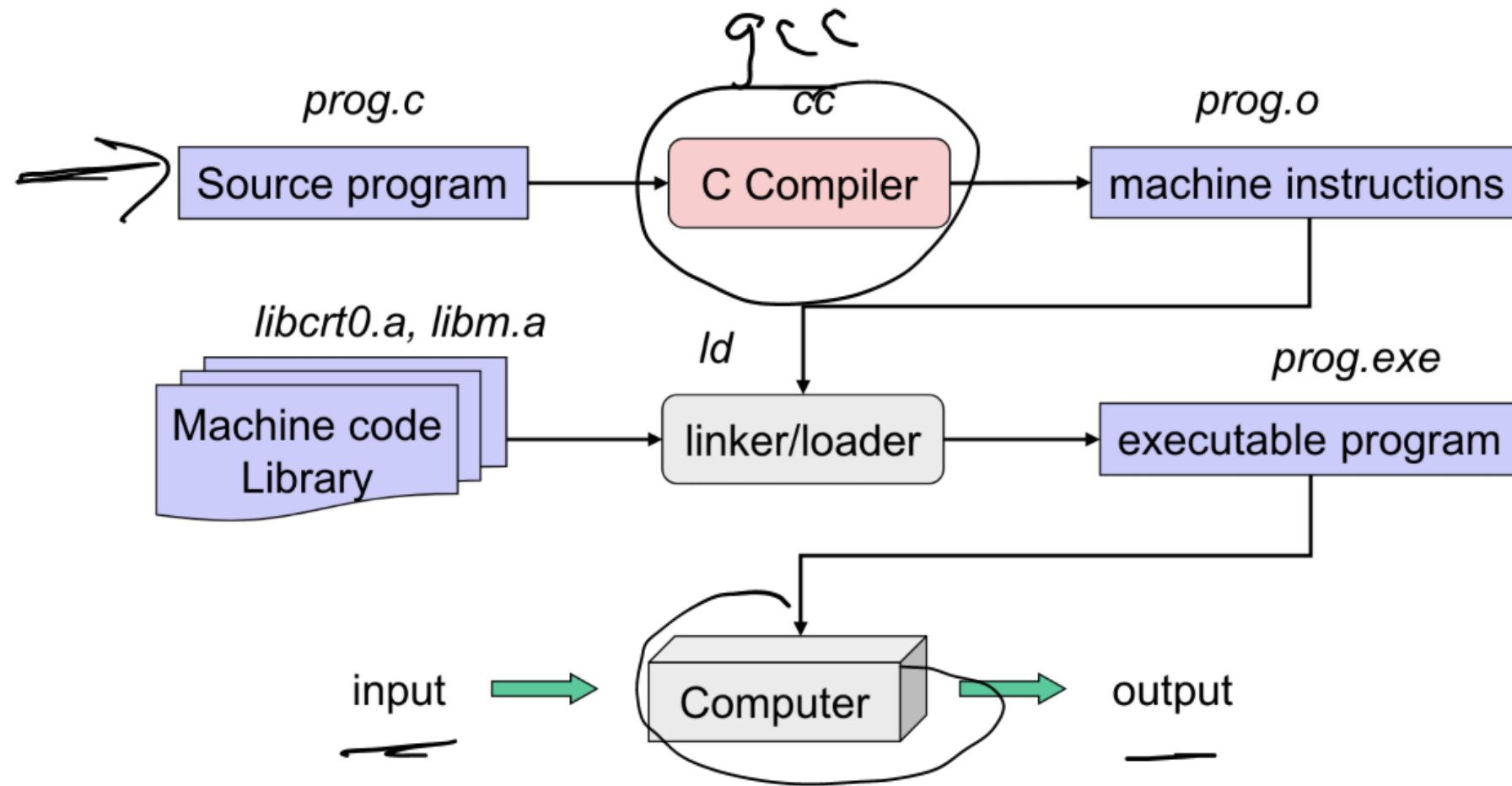
Motivating Capabilities: *The Confused Deputy Attack*

Quick Background: EUID



- Effective user ID (EUID): the user ID that the OS uses to determine **what permissions the program** has when it runs
- There is also Real user ID (RUID): ID of user who started the process.
- Most of the time EUID == RUID.
- When the `setuid` bit is set and a user runs that program, EUID becomes the file owner's UID (usually root), even though the RUID remains the users
 - This lets certain programs perform privileged ops on behalf of ordinary users

Background: The Compiler



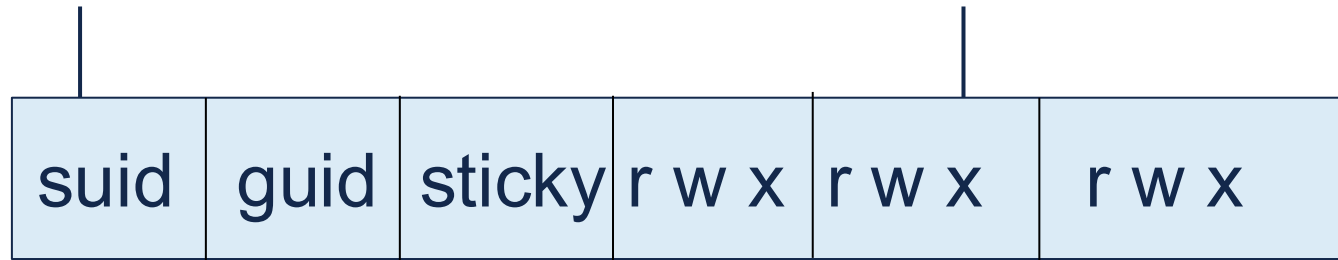
SetUID

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.



SetUID

Group



Owner

Users

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*/  
}
```

Some pseudocode for
the GCC compiler.



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

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

```
    // 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*/
```

```
}
```

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

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

```
// 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*/
```

```
}
```

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

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

```
// write
```

```
(2) FILE
```

```
/*write
```

```
// write
```

```
(3) fp
```

```
/*write to fp*/
```

```
}
```

This is a problem if the compiler is running with root privileges or as an administrator!

It will use its *own privileges* to open /etc/passwd for writing and overwrite the password file! Even if the user who ran the gcc command didn't have permission to do so!

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

In our example:

Role	Description
Deputy	The compiler, has authority to write to system files
Attacker	Unprivileged user, who provides etc/passwd as an output file name
Victim	The system or admin who trusts the compiler
Confusion??	The compiler doesn't realize it's acting on behalf of a user when it opens /etc/passwd for appending. It just uses its own privilege

Confused Deputy Attack



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

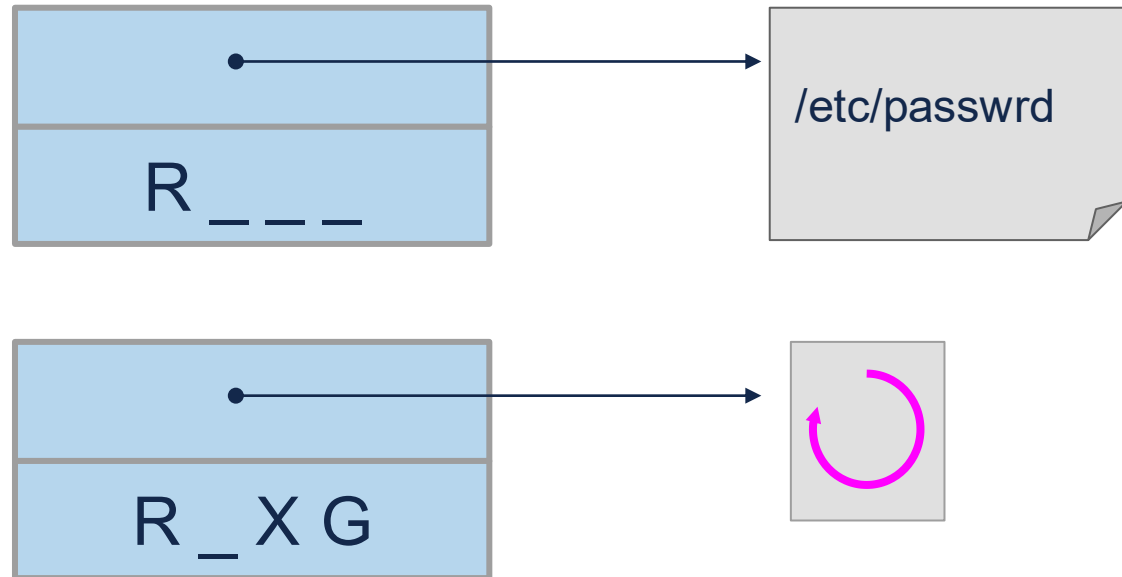
Access Control cannot
prevent the confused
deputy attack

Capabilities



- object reference + access right
- “fat pointer”

READ, WRITE, EXECUTE,
GRANT



```
(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], 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*/
```

```
}
```

```
$ 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



Another Confused Deputy Example: WS



Chromium: Open-source web browser

Node.js: a JavaScript runtime environment that enables developers to execute JavaScript code outside of a web browser

Electron: a framework for building apps w/ HTML, CSS, JS, packaged together with Chromium & Node.JS. Slack, VSCode & Discord are examples of apps built using Electron!