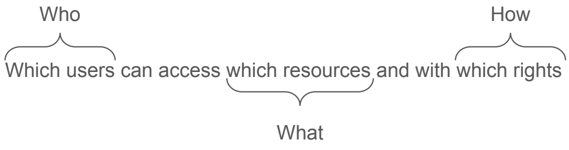
Access Control

COMP 435
Fall 2017
Prof. Cynthia Sturton

Access Control: enacting a security policy

Which users can access which resources and with which rights

Access Control: enacting a security policy



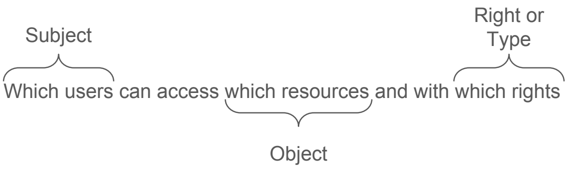
Who

How

Which users can access which resources and with which rights

What

Access Control: enacting a security policy



Subject

Right or
Type

Which users can access which resources and with which rights

Object

Subjects

- Users
- Processes

Objects

- Users
- Processes
- Files
- Memory
- I/O devices

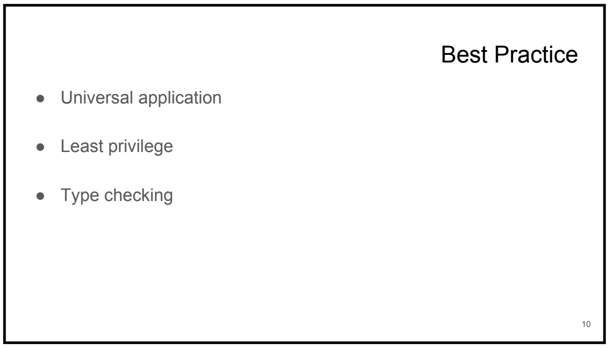
Objects

- Users } Subjects
- Processes }
- Files
- Memory
- I/O devices

Access Type

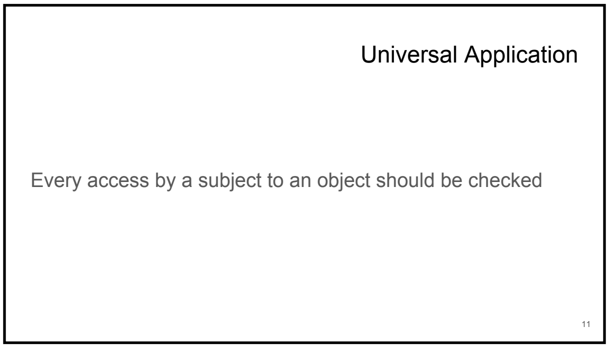
- Read
- Write
- Execute
- Create
- Transfer

Best Practices for Access Control



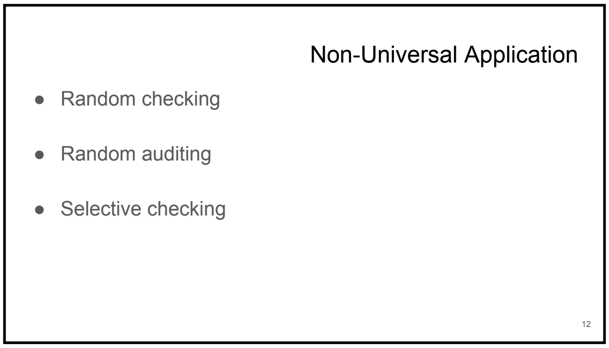
Best Practice

- Universal application
- Least privilege
- Type checking



Universal Application

Every access by a subject to an object should be checked



Non-Universal Application

- Random checking
- Random auditing
- Selective checking

Least Privilege

Every subject should be granted the least amount of access necessary to do its job

Type Checking

Operations should be meaningful for the object accessed

Access Control Policies

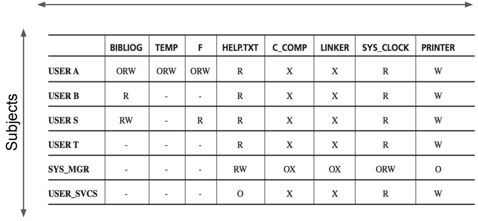
Access Control Policies

- Discretionary Access Control (DAC)
- Mandatory Access Control (MAC)
- Role-based Access Control (RBAC)
- Attribute-based Access Control (ABAC)

Discretionary Access Control (DAC)

Access Control Matrix

Objects



Subjects

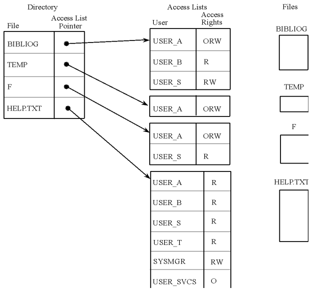
	BIBLIOG	TEMP	F	HELP.TXT	C_COMP	LINKER	SYS_CLOCK	PRINTER
USER A	ORW	ORW	ORW	R	X	X	R	W
USER B	R	-	-	R	X	X	R	W
USER S	RW	-	R	R	X	X	R	W
USER T	-	-	-	R	X	X	R	W
SYS_MGR	-	-	-	RW	OX	OX	ORW	O
USER_SVCS	-	-	-	O	X	X	R	W

From *Security in Computing, Fifth Edition*, by Charles P. Pfleeger, et al. (ISBN: 9780134085043). Copyright 2015 by Pearson Education, Inc. All rights reserved.

Access Control Matrix

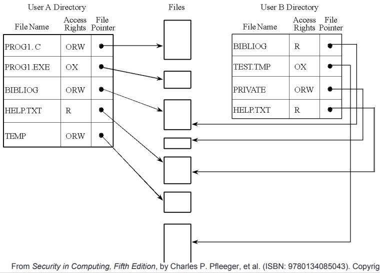
- + Single listing of all objects
  - + Eases revocation
  - + No aliasing
- - Sparse
- - Inefficient

Access Control List



From *Security in Computing, Fifth Edition*, by Charles P. Pfleeger, et al. (ISBN: 9780134085043). Copyright 2015 by Pearson Education, Inc. All rights reserved.

Access Control Directory



File Name	Access Rights	File Pointer
PROG1. C	ORW	●
PROG1.EXE	OX	●
BIBLIOG	ORW	●
HELP.TXT	R	●
TEMP	ORW	●

File Name	Access Rights	File Pointer
BIBLIOG	R	●
TEST.TMP	OX	●
PRIVATE	ORW	●
HELP.TXT	R	●

From *Security in Computing, Fifth Edition*, by Charles P. Pfleeger, et al. (ISBN: 9780134085043). Copyright 2015 by Pearson Education, Inc. All rights reserved.

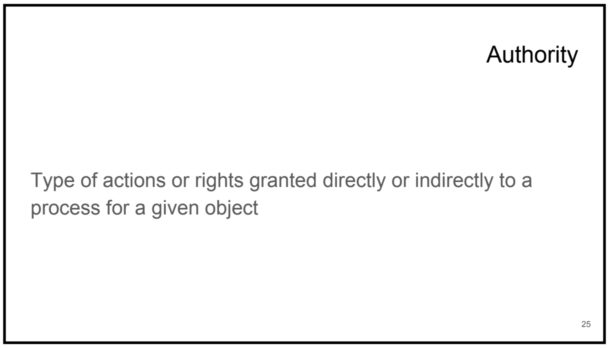
Access Control Directory

+ Easy to implement
+ Easy to understand
- Long lists
- Revoking access requires a search through every list
- Aliasing may cause ambiguous access rights

Permission vs. Authority

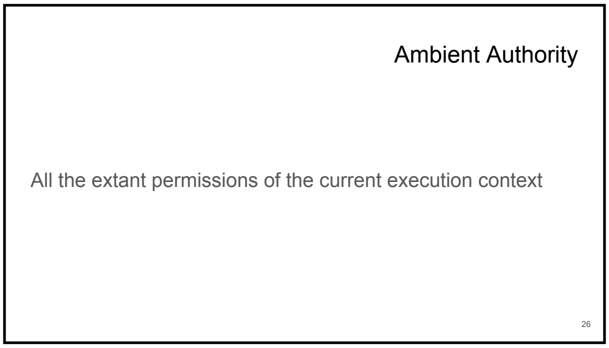
Permissions

Type of actions or rights granted directly to a process for a given object



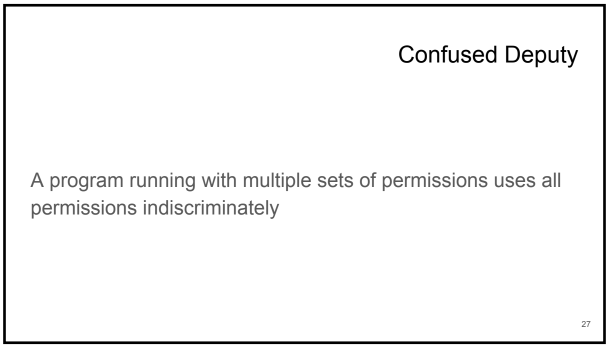
Authority

Type of actions or rights granted directly or indirectly to a process for a given object



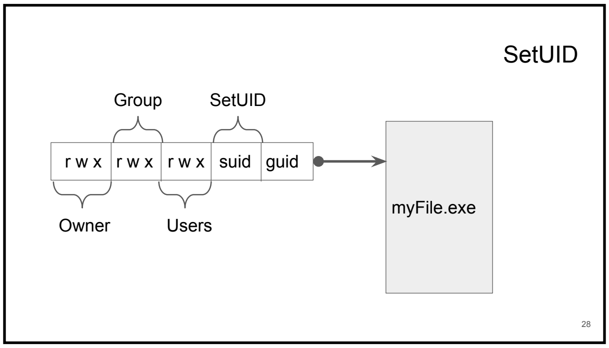
Ambient Authority

All the extant permissions of the current execution context



Confused Deputy

A program running with multiple sets of permissions uses all permissions indiscriminately



SetUID

```
\\gcc program
int main(int argc, char *argv[])
{
 //compile code
   ...
 //write to log
 FILE *fp = fopen(argv[2], "w");
 //write to fp
   ...
 //write out statistics:
 fp = fopen("/etc/compiler_stats", "a");
 //write to fp
   ...
}
```

```
$ gcc prog.c log.txt
```

```
\\gcc program
int main(int argc, char *argv[])
{
 //compile code
   ...
 //write to log
 FILE *fp = fopen(argv[2], "w");
 //write to fp
   ...
 //write out statistics:
 fp = fopen("/etc/compiler_stats", "a");
 //write to fp
   ...
}
```

```
$ gcc prog.c log.txt

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

Analogy:
Confused Valet

Capabilities

- Unforgeable token
- Possession of the token grants access rights
- Directly ties access right to object
- Think physical key

```
\\compiler program
int main(int argc, char *argv[])
{
 //compile code
   ...
 //write to log
 FILE *fp = fopen(argv[2], user_cap);
 //write to fp
   ...
 //write out statistics:
 fp = fopen("/etc/compiler_stats",
system_cap);
 //write to fp
   ...
}
```

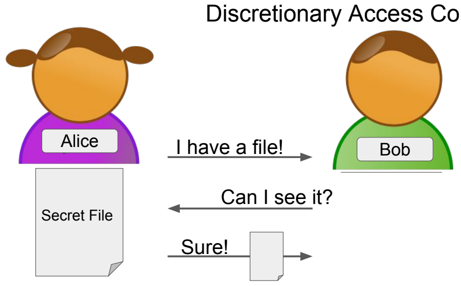
```
$ gcc prog.c log.txt

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

Mandatory Access Control

Discretionary Access Control

Access List for secretFile.pdf	
Alice	R
Bob	-

Discretionary Access Control



Alice

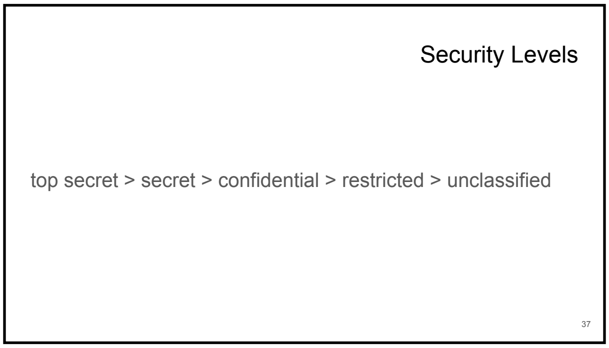
Bob

I have a file!

Can I see it?

Sure!

Secret File

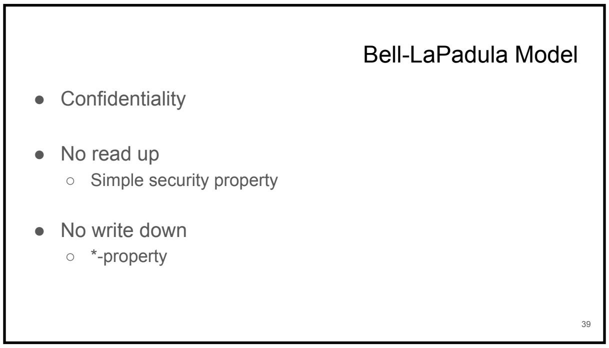


Security Levels

top secret > secret > confidential > restricted > unclassified

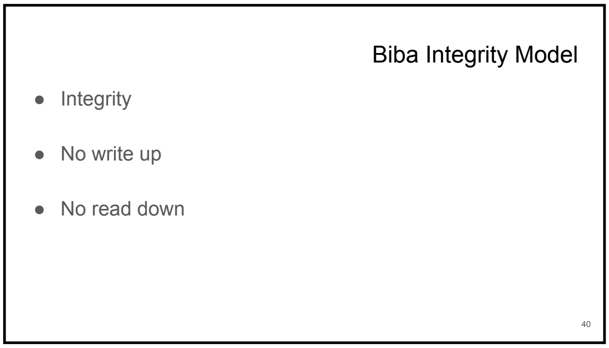
Bell-LaPadula Model

- Confidentiality
- No read up
  - Simple security property



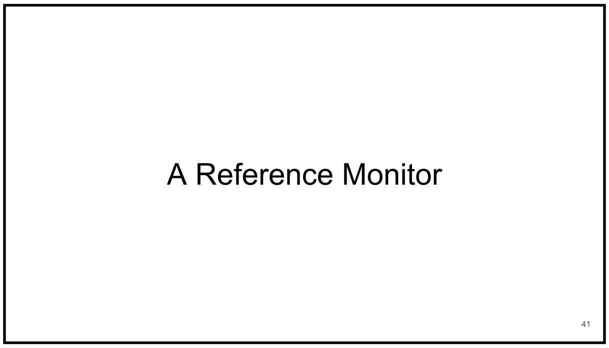
Bell-LaPadula Model

- Confidentiality
- No read up
  - Simple security property
- No write down
  - *-property

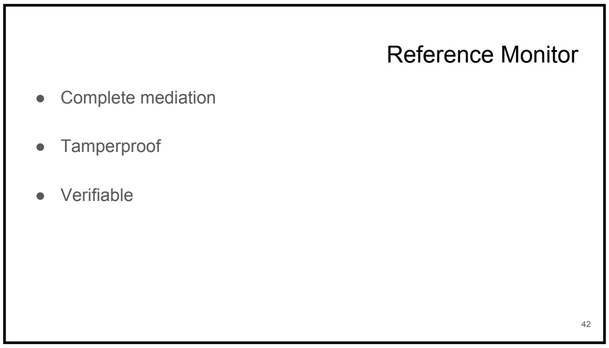


Biba Integrity Model

- Integrity
- No write up
- No read down



A Reference Monitor



Reference Monitor

- Complete mediation
- Tamperproof
- Verifiable