Capability Based Systems

Chester Rebeiro
IIT Madras

https://homes.cs.washington.edu/~levy/capabook/Chapter1.pdf

Confused Deputy Problem

- A computer program that is fooled into misusing authority leading to a privilege escalation

- Fortran Compiler Installed in a directory say SYSX
- Writes billing to a file called SYSX/BILL
- Writes statistics to a file called SYSX/STATS
- The SYSX directory is privileged and cannot be written into by other programs (only the compiler can write into it because it had a LISENCE file)

- Usage of the Fortran compiler will look like this:
  SYSX/FORT file_to_be_compiled output_file

MISUSE BY USER
- SYSX/FORT file_to_be_compiled SYSX/BILL

Bill file is overwritten

Confused Deputy Problem

- **Who is to blame?**

- Compiler?
    - Should the compiler check if for the directory / output file name and prevent access to it?

      (No, the name SYSX was not invented at the time of writing the code; BILL is not the only sensitive file in SYSX)

FIXING THE PROBLEM – SWITCH HATS

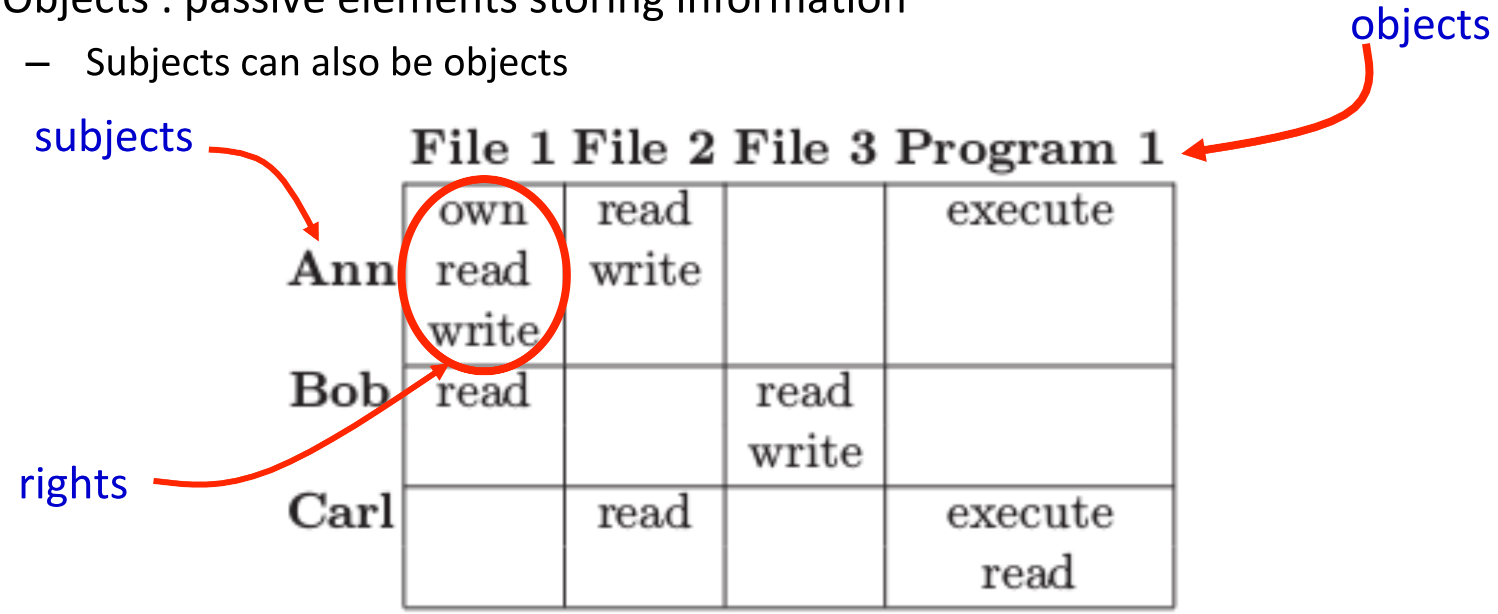
- The compiler wears two hats
  One hat when sensitive information like the file BILL was written into
  Other hat was based on user's privileges to write user file

(However this approach cannot be easily generalized – a program may require multiple hats)

http://people.csail.mit.edu/alinush/6.858-fall-2014/papers/confused-deputy.pdf

Discretionary Access Control

- By Butler Lampson, 1971 (Earliest Form)
- Subjects : active elements requesting information
- Objects : passive elements storing information
  - Subjects can also be objects


subjects
objects
rights

	File 1	File 2	File 3	Program 1
Ann	own read write	read write		execute
Bob	read		read write	
Carl		read		execute read

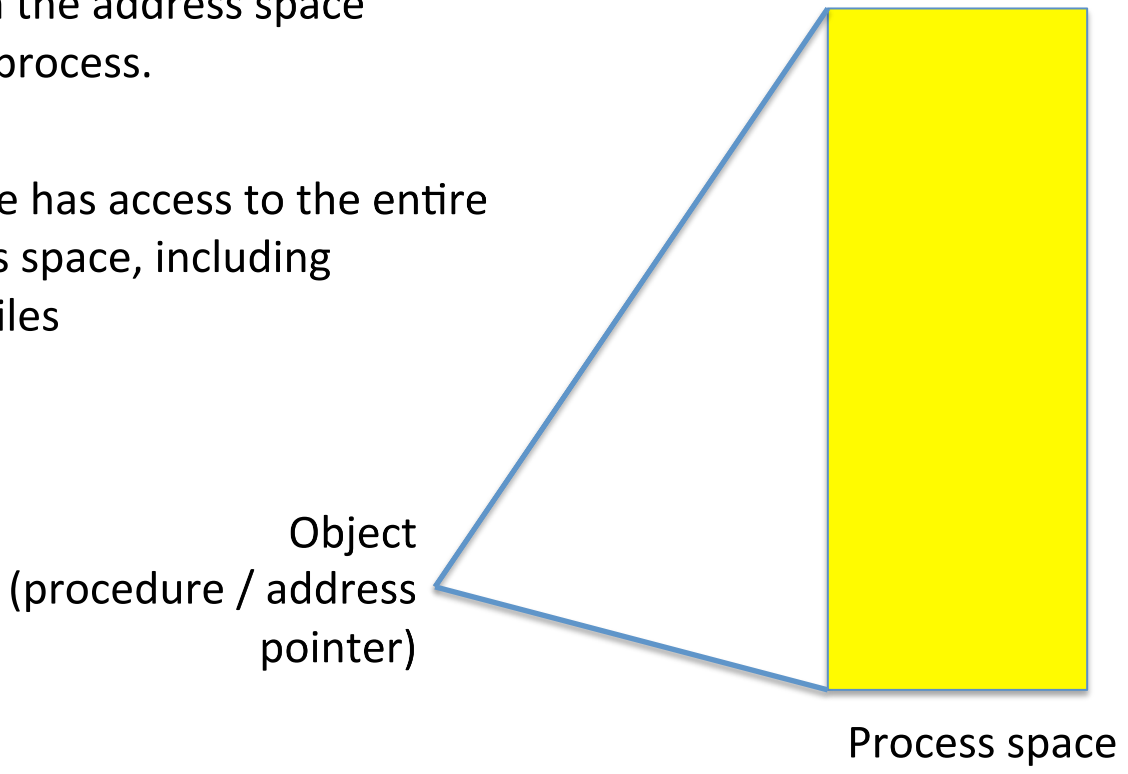
Other actions : ownership (property of objects by a subject),
control (father-children relationships between processes)

Butler Lampson, “Protection”, 1971

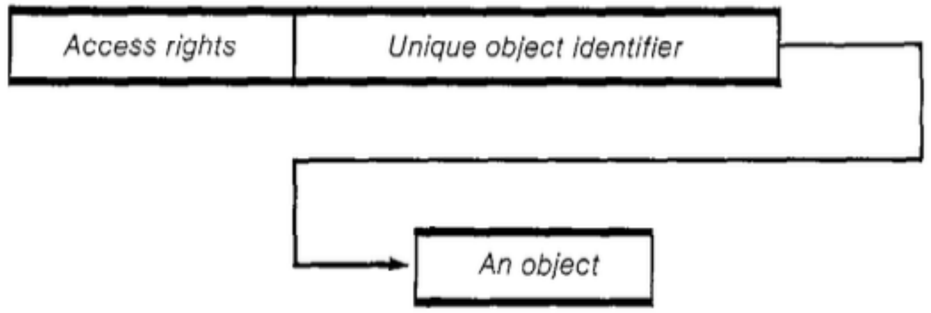
Unix Processes

Every procedure called by a program executes within the address space defined by the process.

Every procedure has access to the entire process address space, including segments and files



Capability Based Systems



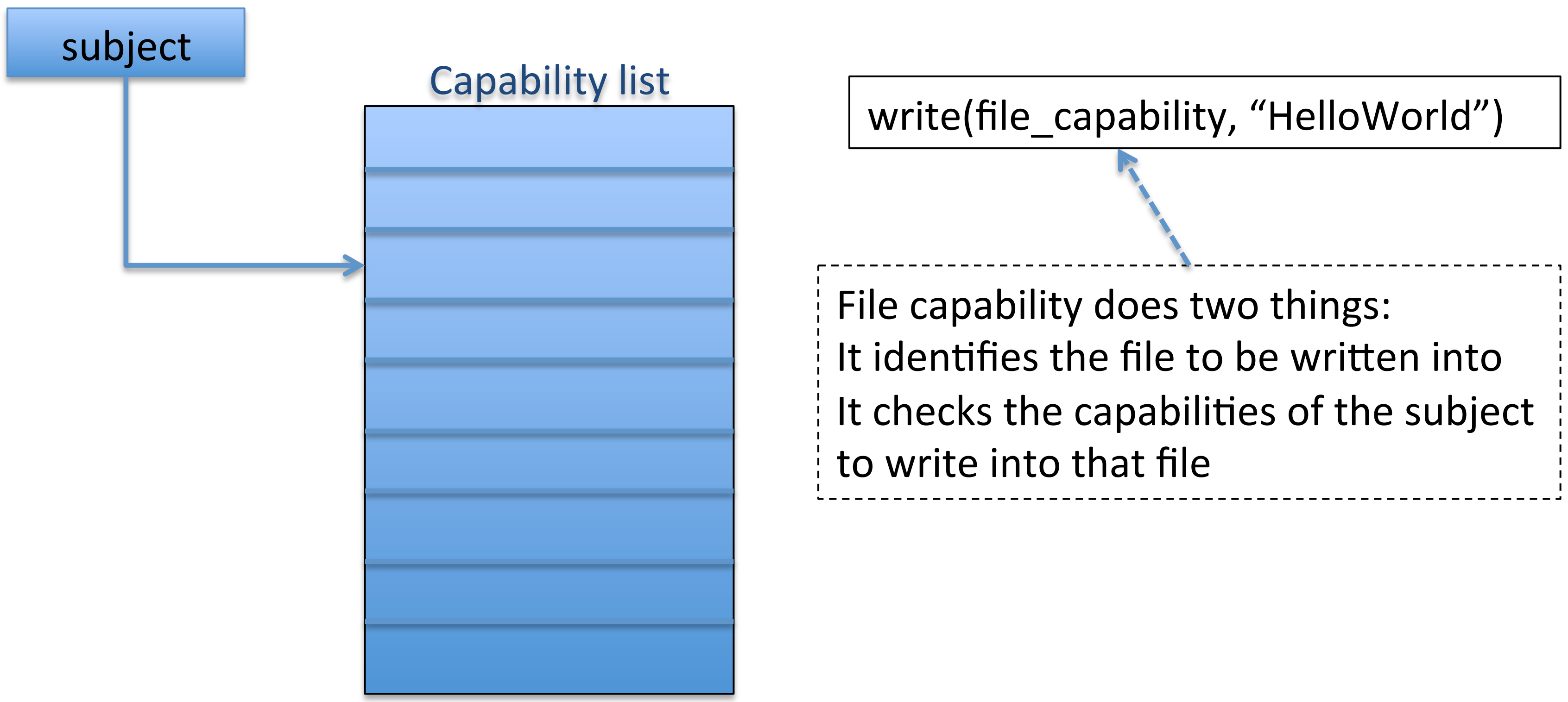
unforgeable

Objects can be any logical entity or physical entity: such as a segment of Memory, an array, a file, IO port

Access rights define the operations that can be performed on the object

Capability Based Systems

- **Subjects:** users, programs, functions, pointers
- Each subject has access to a list of capabilities, which specifies objects that can be accessed



write(file_capability, "HelloWorld")

File capability does two things:
It identifies the file to be written into
It checks the capabilities of the subject to write into that file

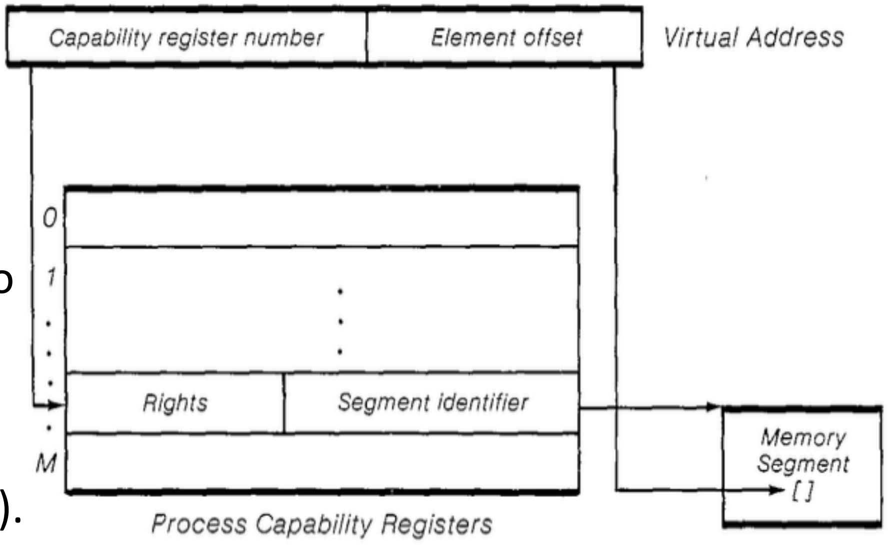
Capability Based Systems

unforgeable

Programs cannot directly modify the capability list

New capabilities can be obtained by requesting the OS or by special hardware instructions

Capability Based Processes

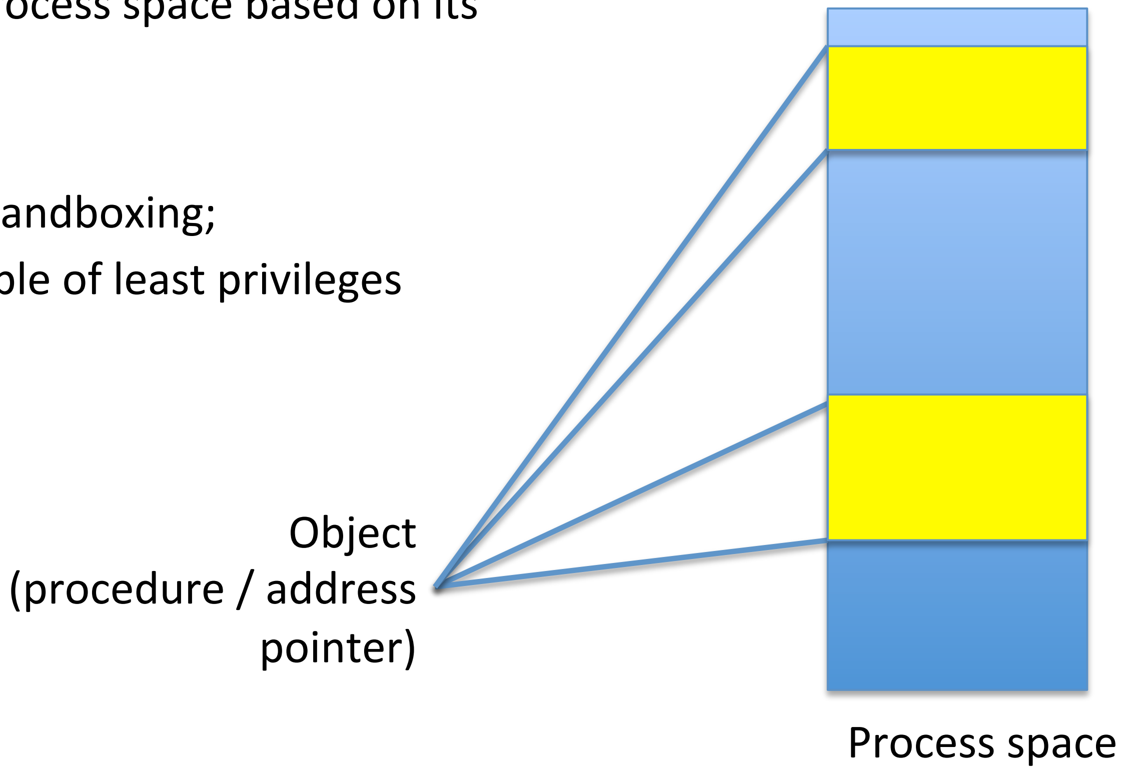


- Process capability registers instead of segments
- A segment of memory is only accessible if a capability of a segment is loaded into a capability register
- Loading a capability register is not a privilege (does not require OS support). However, modifying the capability requires support from the OS.
- Address space dynamically changes depending on capability registers. Done by changing capability registers
- A capability does not have to be local to a process. That is, a segment addressed by a capability is independent of a process. (Easily implement shared libraries)

Capability Processes

A procedure called by a program has access to the process space based on its capabilities.

Easily support sandboxing;
Achieves principle of least privileges easily.

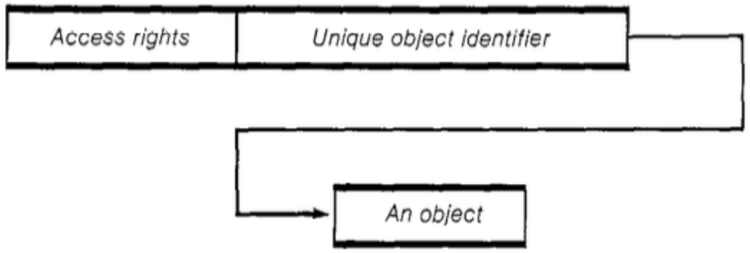


Address Contexts`

- Each object identifier is unique and persistent

- It is used by the OS to locate an object

- The identifier is assigned after the object is created and that identifier is never resused even after the object is deleted.



- Unlike conventional addressing schemes, where addresses are valid within a process, in capability systems, the object identifiers are valid throughout the system.
    - Adv. Capabilities can be freely passed from one process to another and used to access shared data
- Files in secondary devices are also referred to by their object identifiers.

Capability Based Systems

- Hydra, L4 micro kernel, Cambridge CAP processor
- EROS
- Google Fuchsia
- CHERI
- Intel iAPX 432