Protection

John Criswell
University of Rochester

The Basics

Purpose of Protection

- Enforce information sharing and integrity policies
  - Professors can modify grades of students in class
  - Students can examine their grades from any class
  - Students cannot modify grades
- Limit damage caused by errant components
  - Buffer overflow in server doesn't permit grade change

Access Control Matrix

	Object 1	Object 2	Object 3
Subject 1	Read, Write		
Subject 2		Read	
Subject 3		Read, Write	Read, Own
Subject 4	Own	Read, Write, Own	

- Subject: Entity which performs an action
- Object: Entity upon which an action is performed
- Access: Read, Write, Delete, Send Signal, etc…
- Special Access: ***Own***

Why not use the access control matrix in real systems?

Two Problems with Access Control Matrix

- Too large to implement efficiently
- Cannot determine if an unsafe state can occur
  - Reduces to the halting problem

Access Control Lists

- Each object lists
  - Subjects that can access the object
  - What access the subject has to the object

Capabilities

- Each subject lists
  - Objects that the subject can access
  - What access the subject has to the object

Discretionary Access Control (DAC)

Owner decides which subjects can access the object

John

Jim

Mandatory Access Control (MAC)

Administrator decides which subjects can access the object

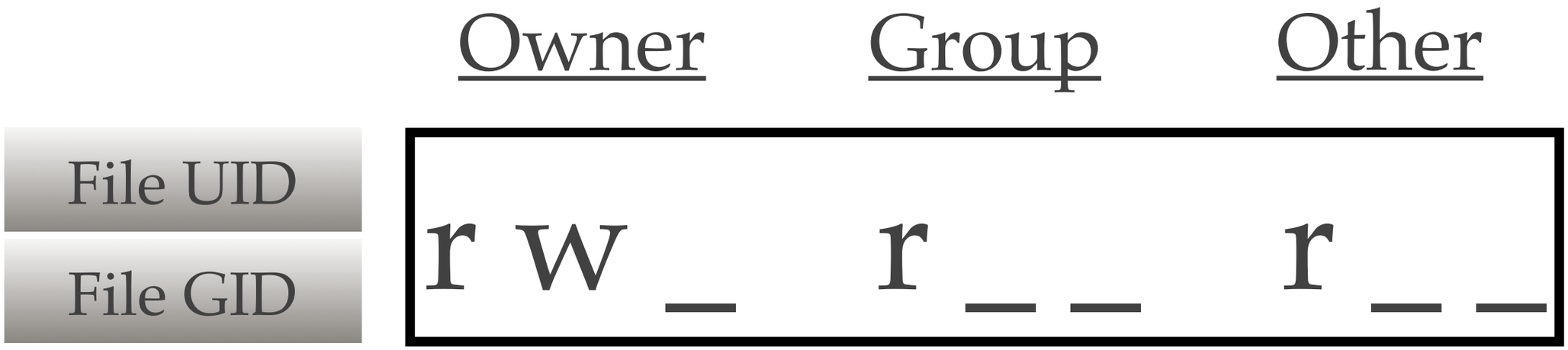
Real Access Controls

(Unprivileged) Unix File Access Control

Owner	Group	Other
r w _	r _ _	r _ _

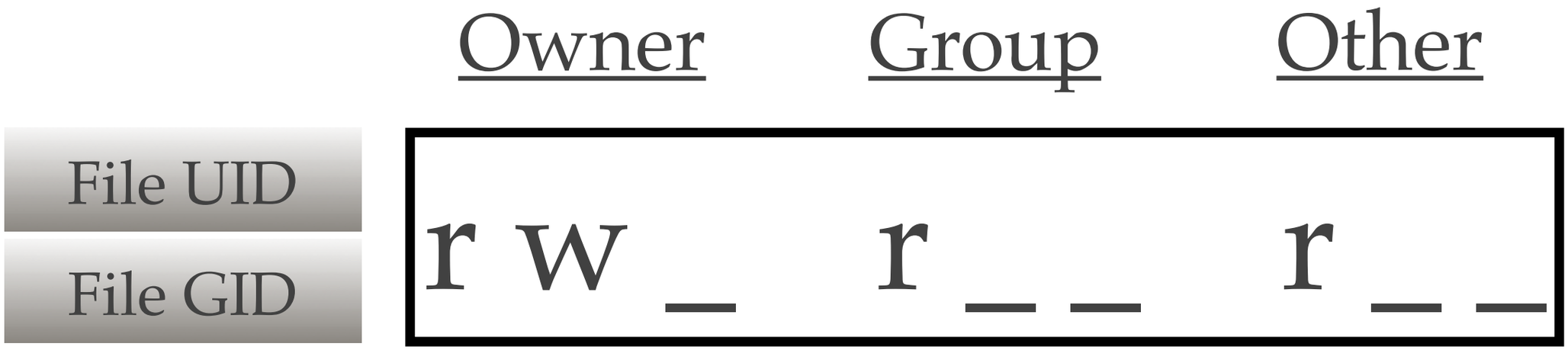
Permission Bit	What the Bit Allows
read	Can open for reading
write	Can open for writing
execute	Can execute or search

(Unprivileged) Unix File Access Control



- If Effective UID matches owner, use *Owner* permissions
- If Effective GID matches group, use *Group* permissions
- Otherwise, use *Other* permissions

(Unprivileged) Unix File Access Control



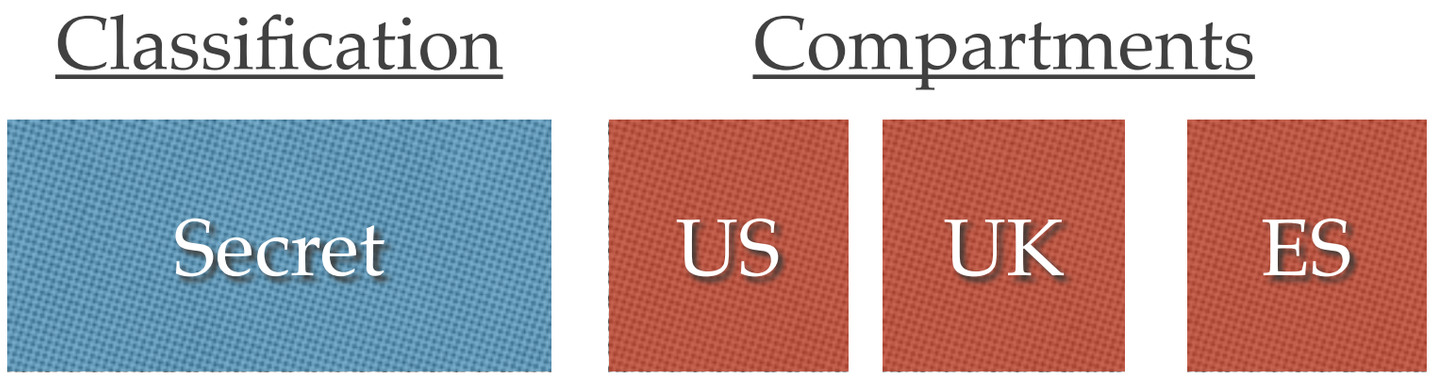
- Owner can modify file permissions to arbitrary value
- Owner can modify file's Group ID
- On some systems, Owner can modify file's Owner ID

setuid Execuables

- File permissions have a setuid bit
- When executed, process UID become file owner UID
- Saved UID is set to effective UID before execve()
- Examples
  - passwd
  - su, sudo
  - ssh

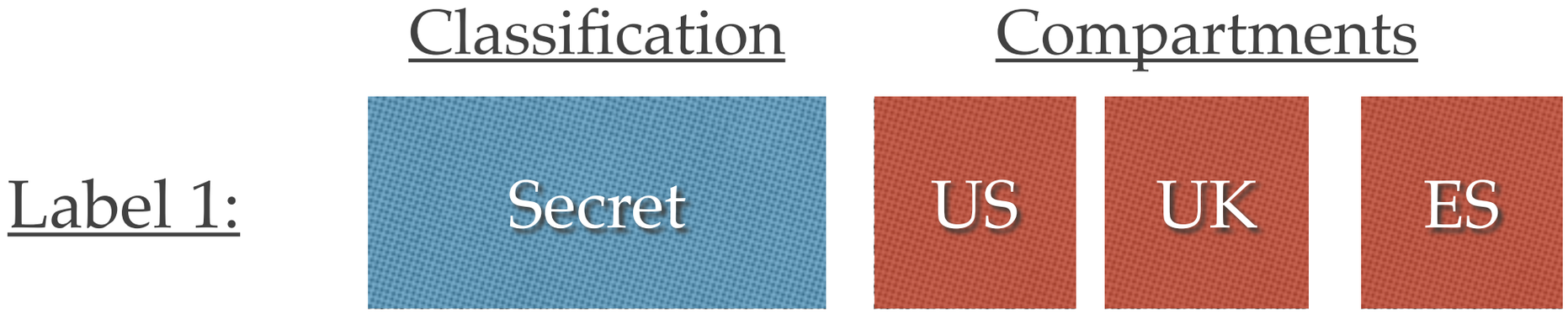
Is Unix access control mandatory or discretionary?

Bell LaPadula



- Attach labels to Subjects and Objects
- Classification: an integer representing secrecy level
- Compartments: bit array representing subsets of data
- Human-readable names associated with classifications and compartments

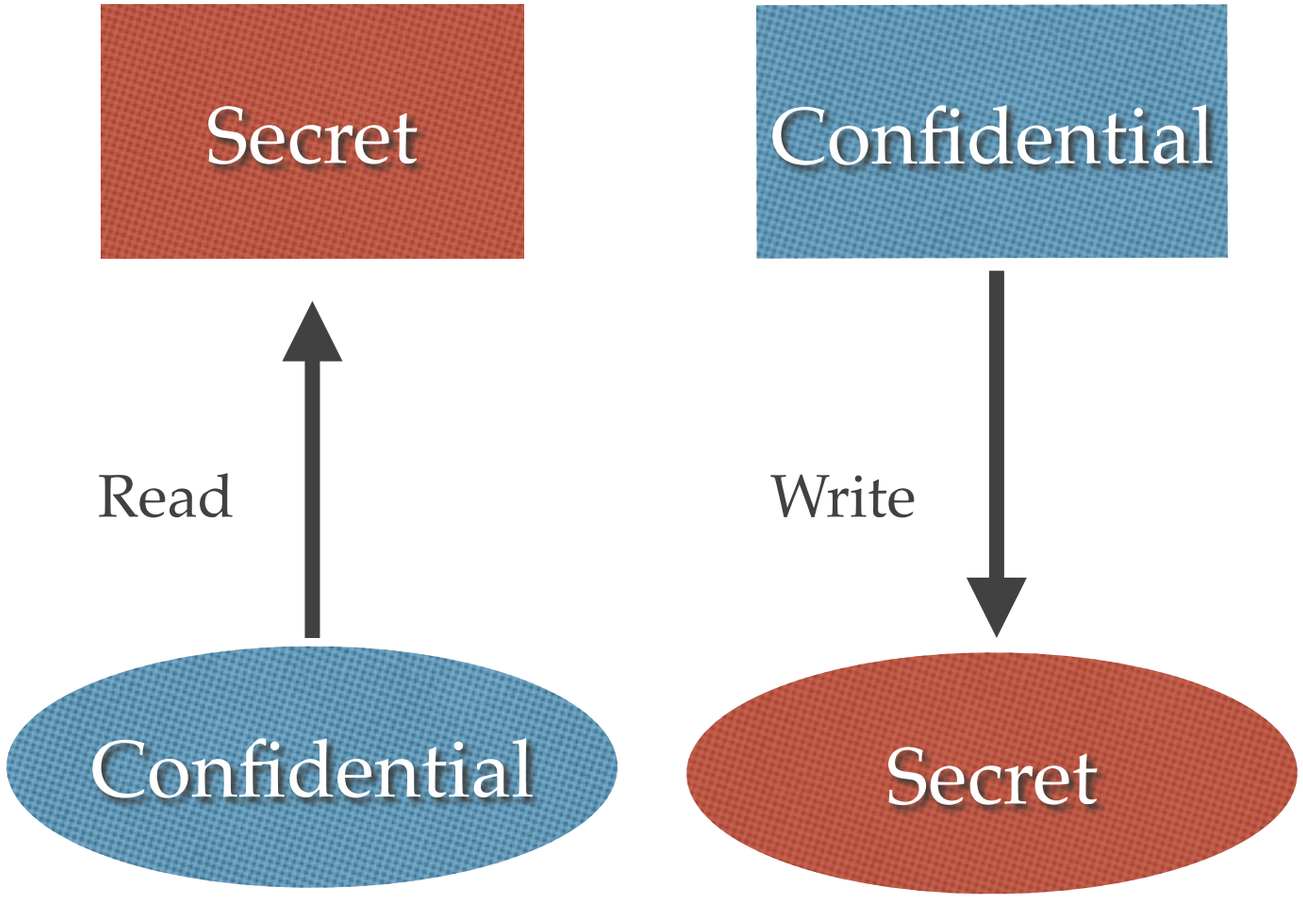
Bell LaPadula: Domination



- Classification2 <= Classification1
- Compartments2 is a subset of Compartments1
- Label1 dom Label2

Bell LaPadula: Access Controls

- Read: $\text{Label}_{\text{Subject}}$ dom $\text{Label}_{\text{Object}}$
- Write: $\text{Label}_{\text{Object}}$ dom $\text{Label}_{\text{Subject}}$



What label should a newly created object have?

What label should a newly created object have?

Bell LaPadula Proof

- Proved that information does not flow from high to low
  - Shows that system cannot enter unsafe state
  - Assumes no privileges to bypass rules
- Proof created a famous controversy
  - McLean questioned how security is defined
  - Controversy led to the creation of a conference
    - Computer Security Foundations (CSF)

Decentralized Information Flow Control

- In Bell-LaPadula, labels created by administrator
- It would be nice to have applications create labels
  - Temporary session IDs
  - Subset of users that are logged in
- Applications create labels
- OS kernel propagates and enforces label policy
- E.g., AsbestOS

Other Access Controls

- Biba Integrity Labels
- Role-based Access Control
- Domain Type Enforcement
- … and many, many more

Privileges

Rules are Made to be Broken

- Real systems need to bypass access control
  - Installing new software
  - Change of policy
  - Change of ownership
  - Fix incorrect configurations
  - Help users solve problems

Privileges

- Override access controls
- Usually a process attribute
  - Note: I think this is a bad idea
- Coarse-grained: User ID 0 (root user)
- Fine-grained: Bit-field of privileges

Coarse-Grained Privileges

- Unix
  - All or nothing: Root UID overrides all access controls

Medium-Grained Privileges

- Linux
  - CAP_CHOWN
  - CAP_DAC_OVERRIDE
  - CAP_DAC_READ_SEARCH
  - CAP_FOWNER
  - CAP_SETUID and CAP_SETGID

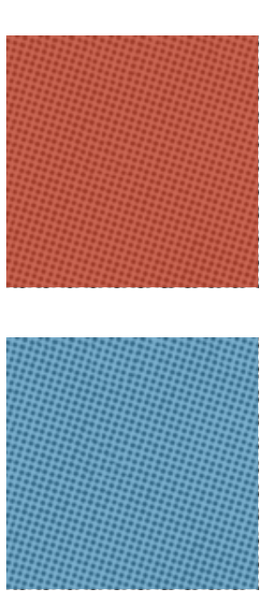
Fine-Grained Privileges: Argus PitBull

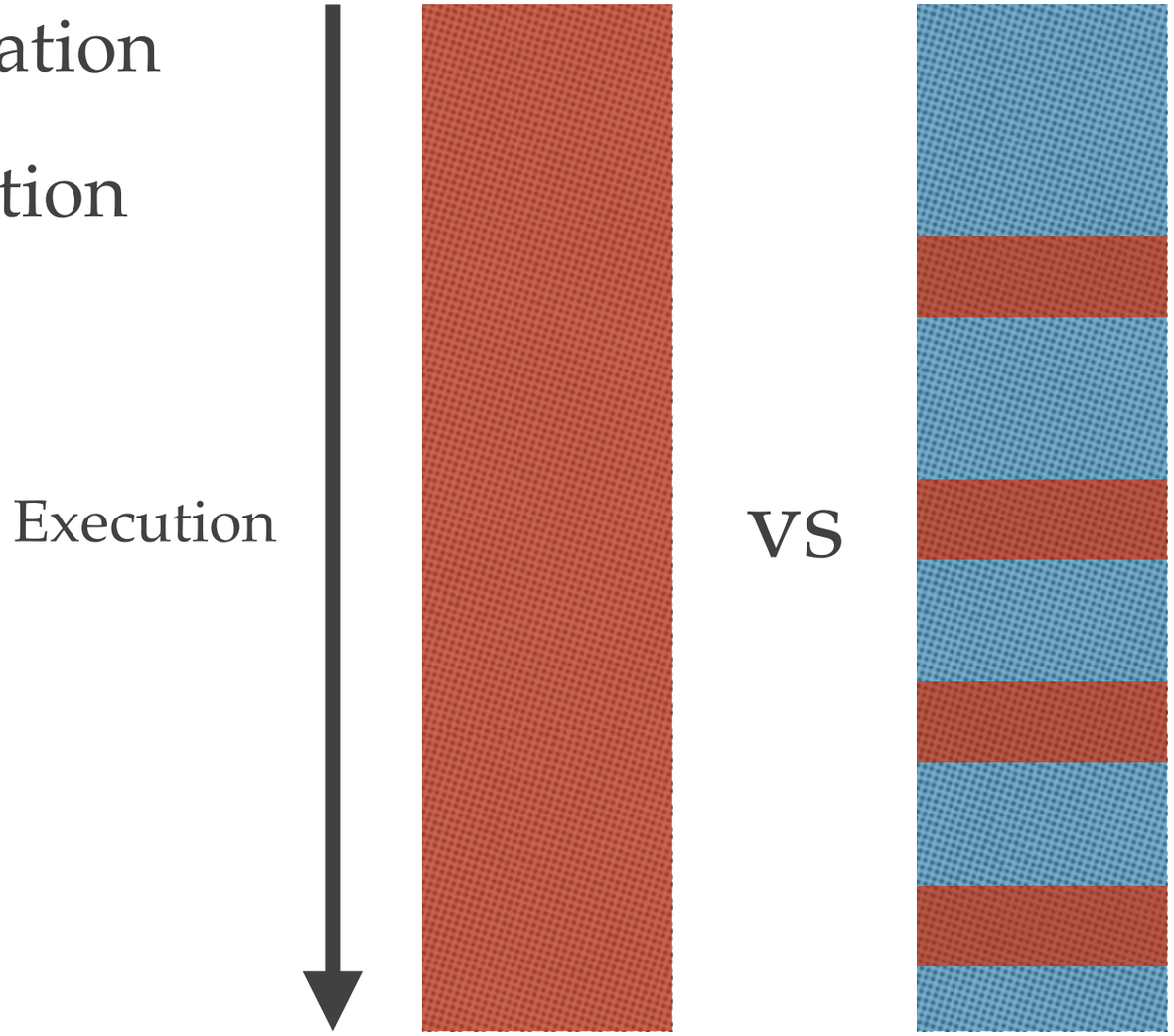
- Hierarchal tree: Top privilege is superset of sub-tree
  - PV_ROOT
    - PV_MAC
      - PV_MAC_READ
      - PV_MAC_WRITE
    - PV_DAC
      - PV_DAC_READ
      - PV_DAC_WRITE
- Separate privileges for overriding read, write, execute
- Separate privilege classes for MAC and DAC override

What is the value of fine-grained privileges?

Privilege Bracketing

- Enable privileges before operation
- Disable privileges after operation





Unix Privilege Bracketing

Real UID	23
Effective UID	23
Saved UID	0

seteuid(getruid());

seteuid(0);

open (“/dev/hd”);

seteuid(getruid());

Unix Privilege Bracketing

Real UID	23
Effective UID	0
Saved UID	0

seteuid(getruid());

seteuid(0);

open (“/dev/hd”);

seteuid(getruid());

Unix Privilege Bracketing

Real UID	23
Effective UID	23
Saved UID	0

seteuid(getruid());

seteuid(0);

open (“/dev/hd”);

seteuid(getruid());

Argus PitBull Privilege Bracketing

Maximum Privilege Set

PV_DAC_R, PV_DAC_W, PV_DAC_X

priv_raise (PV_DAC_R);

Effective Privilege Set

open (“/dev/hd”, O_RDONLY);

priv_lower (PV_DAC_R);

Argus PitBull Privilege Bracketing

Maximum Privilege Set

PV_DAC_R,
PV_DAC_W,
PV_DAC_X

priv_raise (PV_DAC_R);

Effective Privilege Set

PV_DAC_R

open (“/dev/hd”, O_RDONLY);

priv_lower (PV_DAC_R);

Argus PitBull Privilege Bracketing

Maximum Privilege Set

PV_DAC_R, PV_DAC_W, PV_DAC_X

priv_raise (PV_DAC_R);

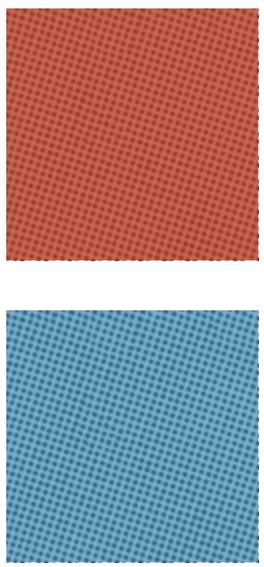
Effective Privilege Set

open (“/dev/hd”, O_RDONLY);

priv_lower (PV_DAC_R);

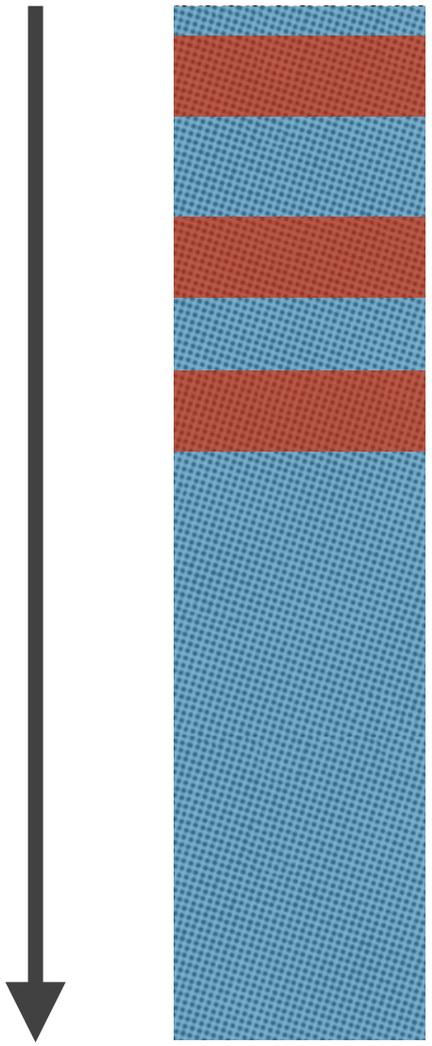
Privilege Dropping

- Remove privilege permanently when no longer needed



Privileged Execution

Non-privileged Execution



What is the value of privilege bracketing?

Open Research Questions

- How to design access controls that are ***usable***?
  - SELinux and PitBull too difficult to use
    - Requires significant system integration effort
- Retrofitting access controls to existing systems
  - Causes very confusing (but correct) system behavior
- Can tools configure access controls to enforce policies?

Open Research Questions

- How much does better privilege handling help?
- How fine-grained do privileges need to be?
  - Answer may lie in bounded model checking
- Programming patterns that reduce privilege use