CS260 – Advanced Systems Security

OS Responsibilities

April 7, 2025

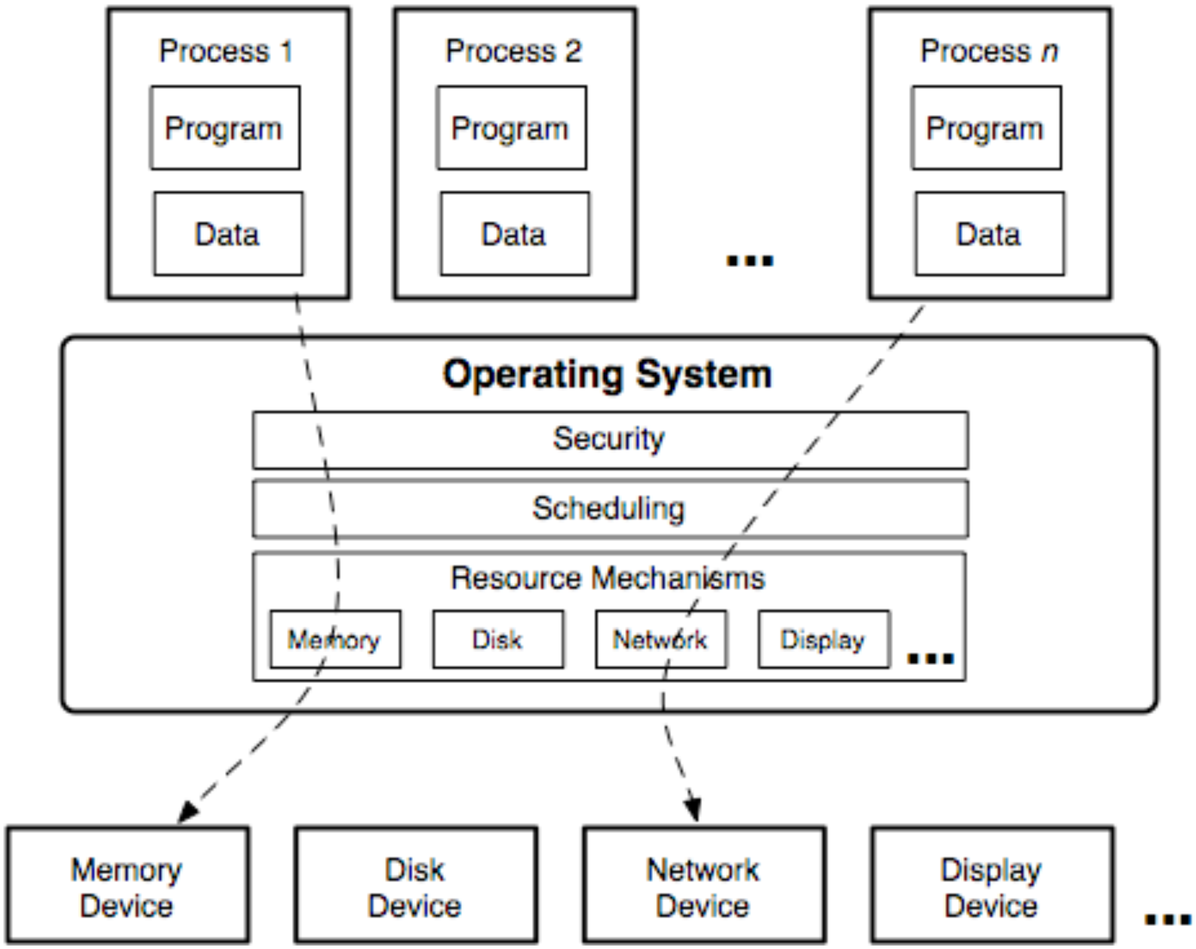
Who's Going To Prevent Attacks?

- While an adversary may
  - Trick a user into downloading and running bad code
  - Turn good code bad
  - Or trick good code into performing actions chosen by the adversary
- How are these threats going to be addressed?

Who's Going To Prevent Attacks?

- While an adversary may
  - Trick a user into downloading and running bad code
  - Turn good code bad
  - Or trick good code into performing actions chosen by the adversary
- How are these threats going to be addressed?
  - Programs?
  - Operating Systems?
  - Hardware?

Operating Systems



Operating Systems and Security

- Have historically been responsible for "security"
  - Enable the execution of multiple programs
  - On CPU architectures
  - With a multitude of devices
- Goal: Keep programs from interfering with each other
  - Regardless of the hardware/devices used
  - And whatever the programmers may do
- Not easy to achieve in general
  - And especially not against a malicious adversary

The Security Challenge

- Today, we are going to examine the challenges facing an operating system when enforcing security against a malicious adversary

Dealing with Bad Code

- What do we want to do to prevent bad code from compromising our system?

Dealing with Bad Code

- What do we want to do to prevent bad code from compromising our system?
  - Limit communication with other processes

Dealing with Bad Code

- What do we want to do to prevent bad code from compromising our system?
  - Limit communication with other processes
- Systems consist of many resources that enable processes to interact
  - Files
  - IPCs
  - Shared memory
  - Network, etc
- How do we limit access to these?

Access Control

- System makes a decision to grant or reject an access request
  - from an already authenticated subject
  - based on what the subject is authorized to access
- Access request
  - Object: System resource
  - Operations: One or more actions to be taken
  - Subject: Process that initiated the request
- Access Control Mechanisms enforce Access Control Policies to make such decisions

Access Control Policy

- How is an access control policy expressed and managed?
- Protection System
  - It describes what operations each subject (via their processes) can perform on each object
- Consists of
  - **State:** *Protection state*
    - I.e., The access control policy
  - **State Ops:** *Protection state operations*
    - I.e., How the policy can be changed

Access Matrix

- Lampson formalizes the model of access control in his 1970 paper "Protection"
- Called Access Matrix
  - Rows are subjects
  - Columns are objects
  - Authorized operations listed in cells
- To determine if S_i has right to object O_j, compare the ops to the appropriate cell

	O	O	O
S	Y	Y	N
S	N	Y	N
S	N	Y	Y

Access Matrix

- Using the Access Matrix
- (1) Suppose J wants to prevent other users' processes from reading/writing her private key (object O_1)
- (2) Suppose J wants to prevent other users' processes from writing her public key (object O_2)
- Design the access matrix
- Are these the rights on your host to your SSH public and private keys?

	O_1	O_2	O_3
J	?	?	?
S	?	?	?
S	?	?	?

UNIX Access Control

- On Files
  - All objects are files
  - Not exactly true
- Classical Protection System
  - Access matrix
  - Discretionary protection state operations
- Practical model for end users
  - Still involves some policy specification

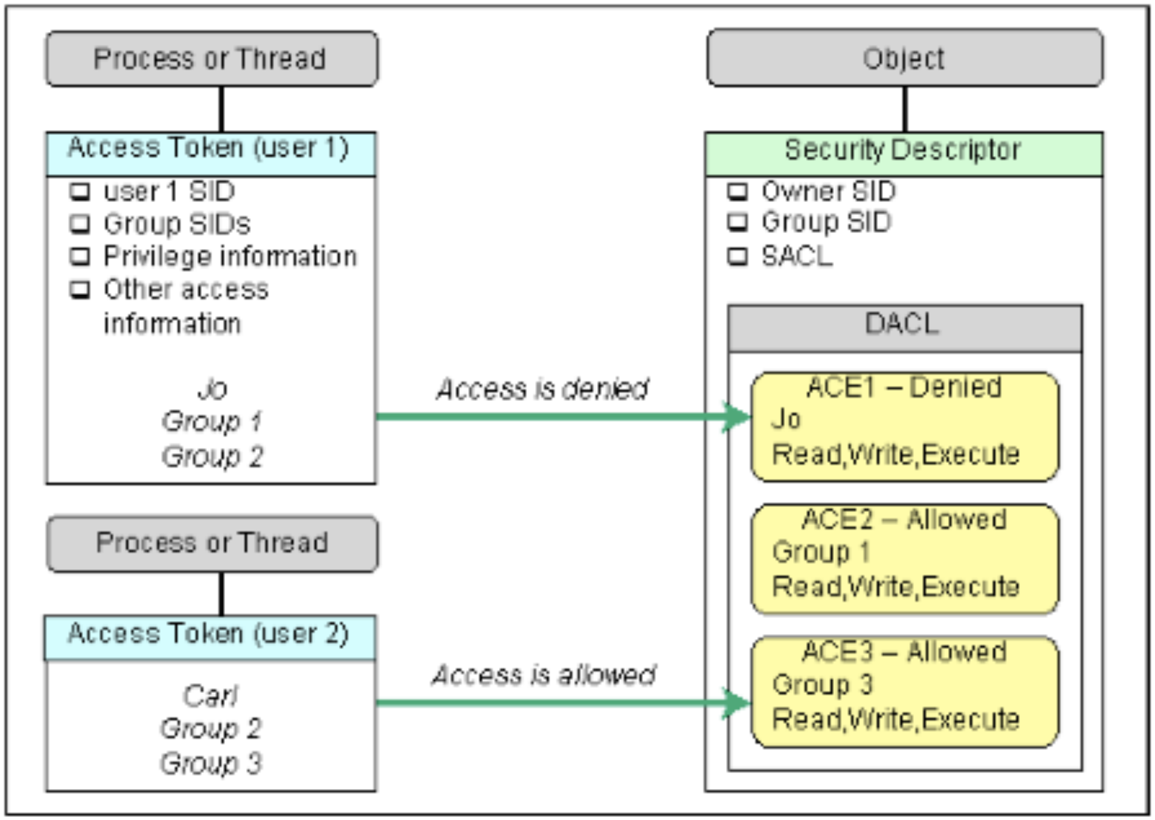
UNIX Mode Bits

-rw-rw-r--	1 pbg	staff	31200	Sep 3 08:30	intro.ps
drwx------	5 pbg	staff	512	Jul 8 09.33	private/
drwxrwxr-x	2 pbg	staff	512	Jul 8 09:35	doc/
drwxrwx---	2 pbg	student	512	Aug 3 14:13	student-proj/
-rw-r--r--	1 pbg	staff	9423	Feb 24 2003	program.c
-rwxr-xr-x	1 pbg	staff	20471	Feb 24 2003	program
drwx--x--x	4 pbg	faculty	512	Jul 31 10:31	lib/
drwx------	3 pbg	staff	1024	Aug 29 06:52	mail/
drwxrwxrwx	3 pbg	staff	512	Jul 8 09:35	test/

Windows Access Control

- On Objects
  - Arbitrary classes can be defined
  - New classes can be defined (Active Directory)
- Classical Protection System
  - Full-blown ACLs (even negative ACLs)
  - Discretionary protection state operations
- Not so usable
  - Few people have experience

Windows Access Control



Access Matrix

- Using the Access Matrix
- (1) Suppose J wants to protect a private key (object O_1) from being leaked to or modified by others
- (2) Suppose J wants to prevent a public key (object O_2) from being modified by others
- Design the access matrix
- Will this access matrix protect the keys' secrecy and integrity?

	O_1	O_2	O_3
J	?	?	?
S	?	?	?
S	?	?	?

Consider Bad Code Again

- Claim: Any code you run may be able to compromise either of the key files
- For the private key
  - Any process running under your user id can read and leak your private key file
- For the public key
  - Any process running under your user id may modify the public key file
    - Often people make the public key file read-only even to the owner
    - Is that enough?

Consider Bad Code Again

- Claim: Any code you run may be able to compromise either of the key files
- For the private key
  - Any process running under your user id can read and leak your private key file
- For the public key
  - Any process running under your user id may modify the public key file
    - Often people make the public key file read-only even to the owner
    - No. Processes running on behalf of the owner may change perms

Bad Code - Examples

- Suppose you download and run adversary-controlled code (e.g., Trojan horse)
  - It will run with all your permissions
  - Even can modify the permissions of any files you own
- Suppose you run benign code that is compromised by an adversary – becoming bad
  - Is effectively the same as above - if adversary can choose code to execute (e.g., return-oriented attack)
  - Adversaries can also trick victims into performing operations on their behalf (e.g., confused deputy attack)

Protection vs. Security

- Protection
  - All security goals met under *benign* processes
  - Protects against an error by a non-malicious entity
- Security
  - All security goals met under *malicious* processes
  - Enforces requirements even if adversary is in complete control of the process
- Hence, for J: Non-malicious processes shouldn't leak the private key by accident to a specific file owned by others
- A potentially malicious process may contain a Trojan horse that can write the private key to files chosen by adversaries

Fundamentally Flawed

- Conventional operating system mechanisms enforce protection rather than security
  - Protection is fundamentally incapable of defending from an active and determined adversary

Security

- Integrity
  - High integrity processes and objects will not be modified by an adversary
    - High integrity processes will remain high integrity
    - High integrity objects will only contain high integrity data
- Secrecy
  - High secrecy data will not be leaked to adversaries
    - High secrecy processes will not leak data
    - High secrecy data will remain in high secrecy objects
- Even when a system may contain malicious processes

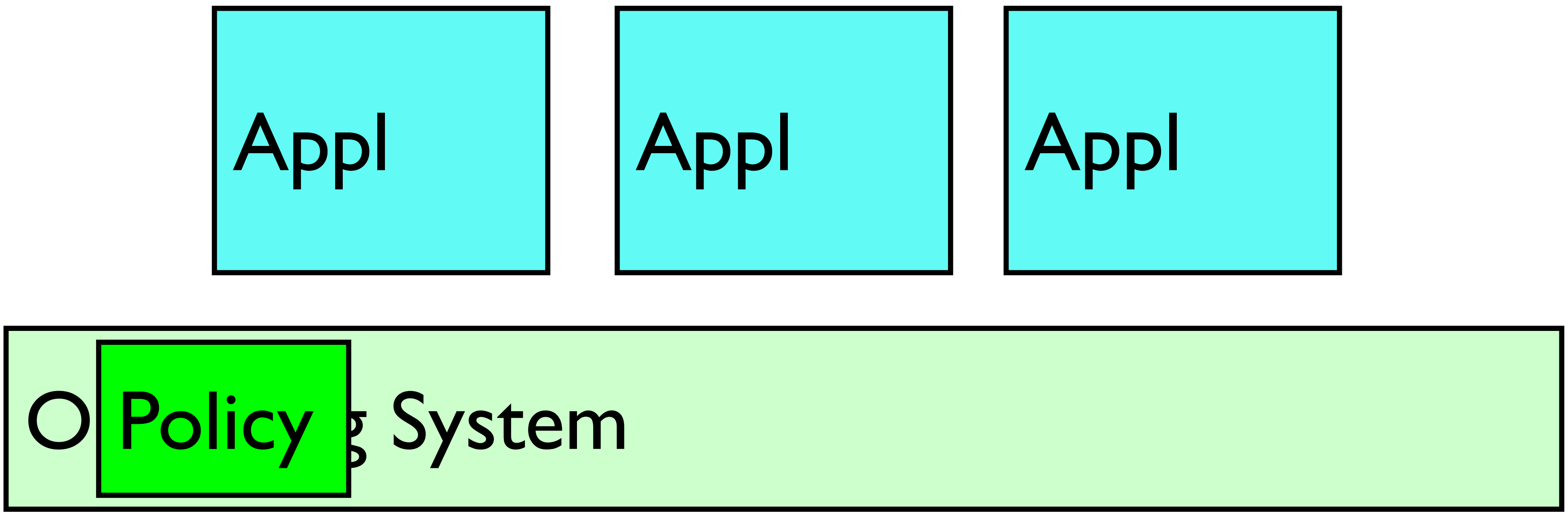
Integrity

- Process integrity requires that the process not depend on adversary input
  - What does “depend on” mean?
  - This is a very difficult requirement to meet
- Suppose a benign process can read from a file controlled by an adversary
- Unless the process is trusted to contain no vulnerabilities then the process could be compromised (is *potentially malicious*)

Secrecy

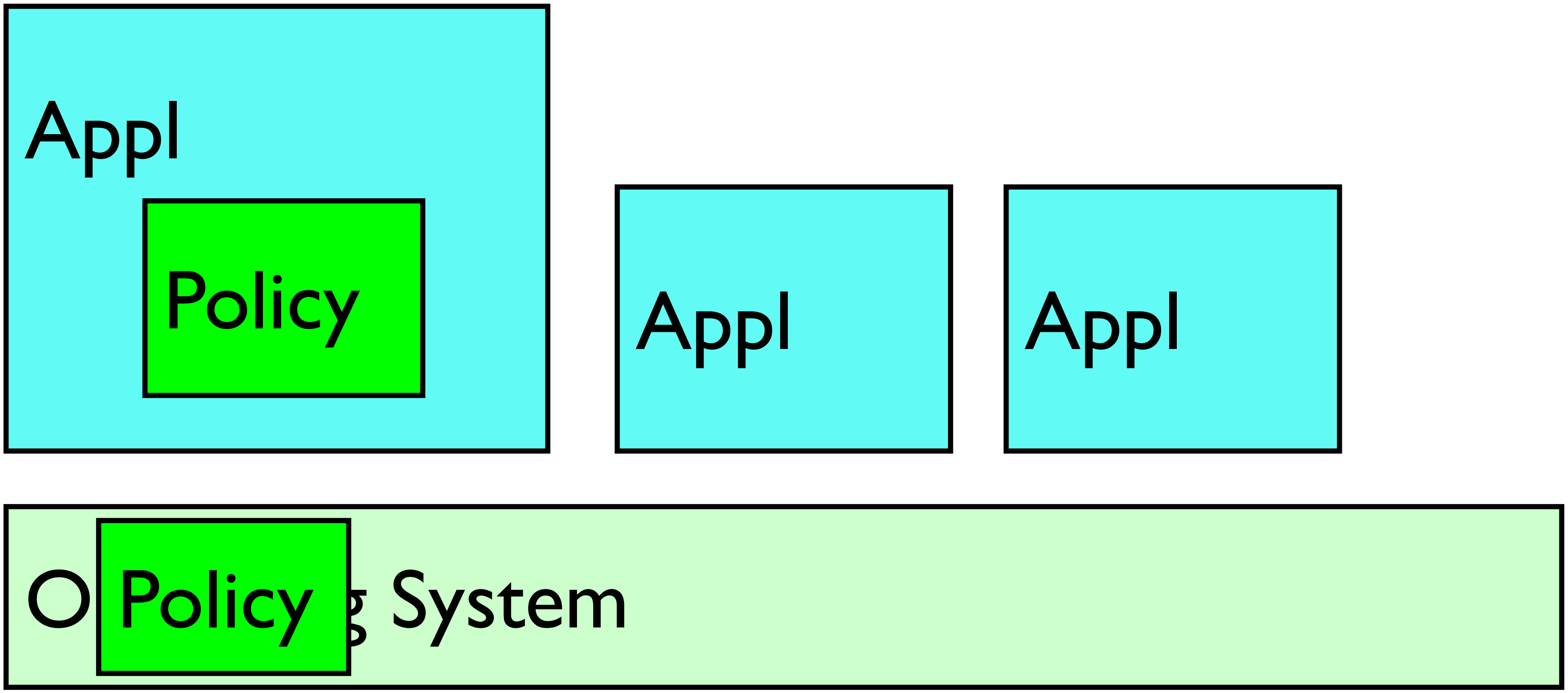
- Process secrecy requires that the process not communicate with unauthorized parties
  - But what about a process that services requests?
  - This is a very difficult requirement to meet
- Suppose a benign process can write to a file controlled by an adversary
- Unless the process is trusted to contain no vulnerabilities then the process could be compromised (is potentially malicious)

Trusted Computing Base



- Historically, OS treats applications as black boxes
  - OS controls flows among applications
  - OS is the Trusted Computing Base

Policy Enforcement in Apps

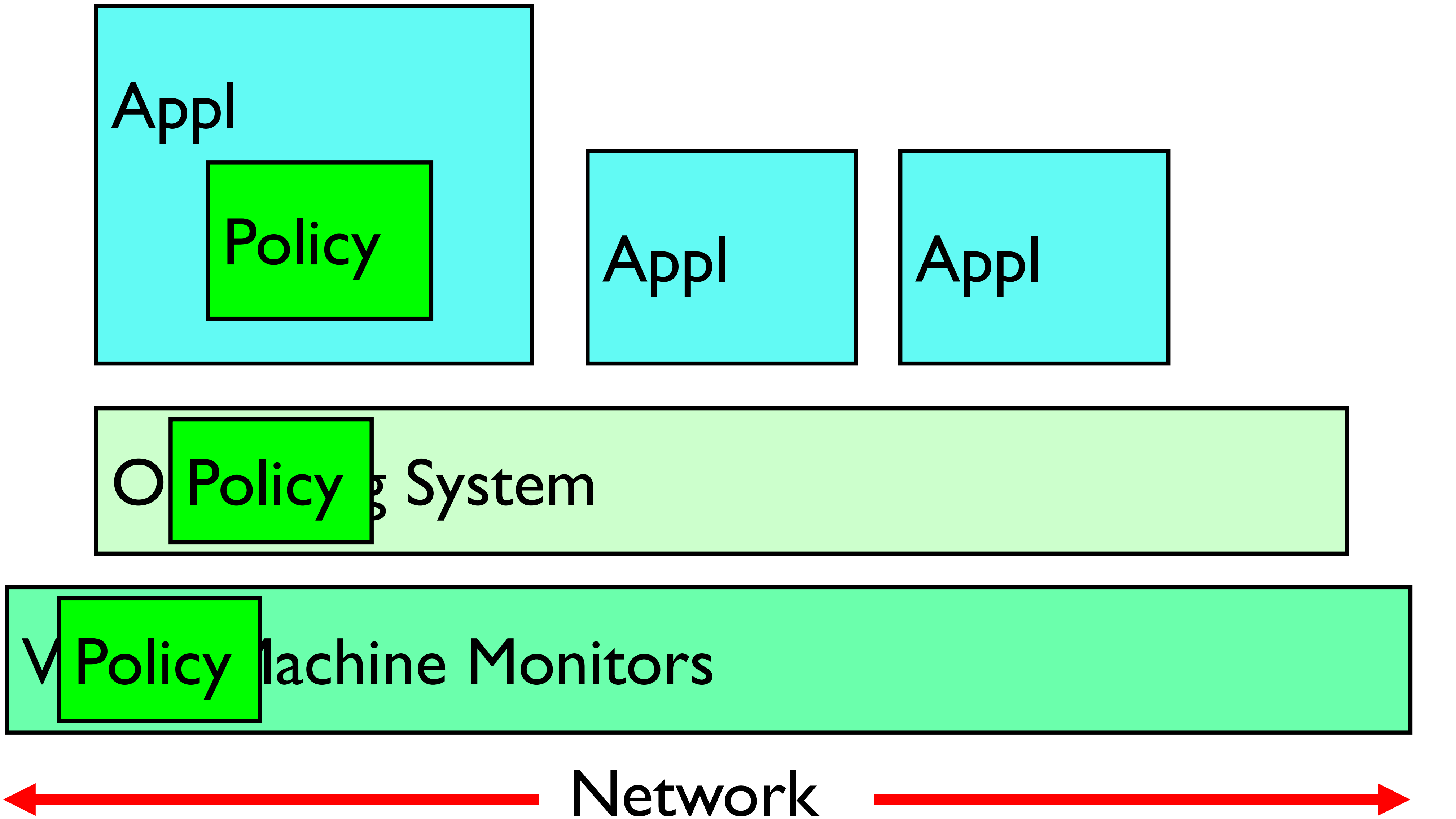


- Application policy enforcement: databases, JVM, X Windows, daemons, browsers, email clients, servers

Application Layer in TCB?

- Do not trust applications
  - Why not?
- But, we need to depend on some application enforcement
  - Some root/privileged processes
  - Have more semantics
  - May be able to break system
- May need to trust apps in a partial way

Multi-Layered Enforcement

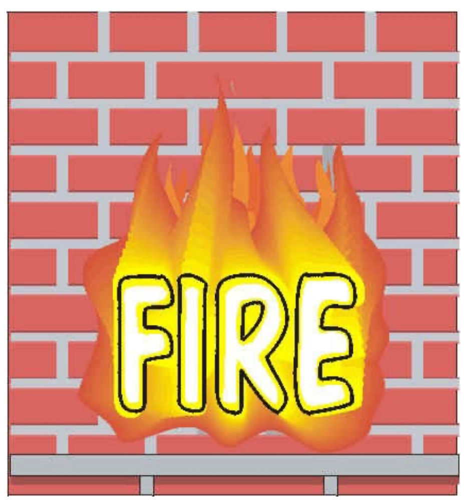


Virtual Machine Layer

- Key technology: Isolation
  - Each VM is a protection domain
- Problem: VM internals are not homogeneous
  - There are security-critical apps
  - There are untrusted inputs and less-critical apps
- How to use VM isolation and control of flows among VMs to achieve security goals?

Network Layer

- Network Access Control == Firewall
  - Protect a network from external malice
    - This is a beneficial view of the world
  - But, is the internal network (hosts) ready for the approved (but untrusted) messages?



Questions for This Class

- How do we keep benign code from becoming bad code?
- How do we prevent benign code from being tricked into being a confused deputy?
- How do we restrict code that may be/go bad from propagating damage?
- How can we leverage the myriad of system defenses to control code efficiently?
- How do we know what we configured works?

Take Away

- Traditional OS access control
  - Is for protection, not security
- So it cannot confine an active adversary
  - Build attacks that work despite access control
  - They can change the access control policies
- Access control is enforced in many places now
  - Can we utilize them comprehensively and efficiently?

Questions