Systems and Internet Infrastructure Security

Network and Security Research Center
Department of Computer Science and Engineering
Pennsylvania State University, University Park PA

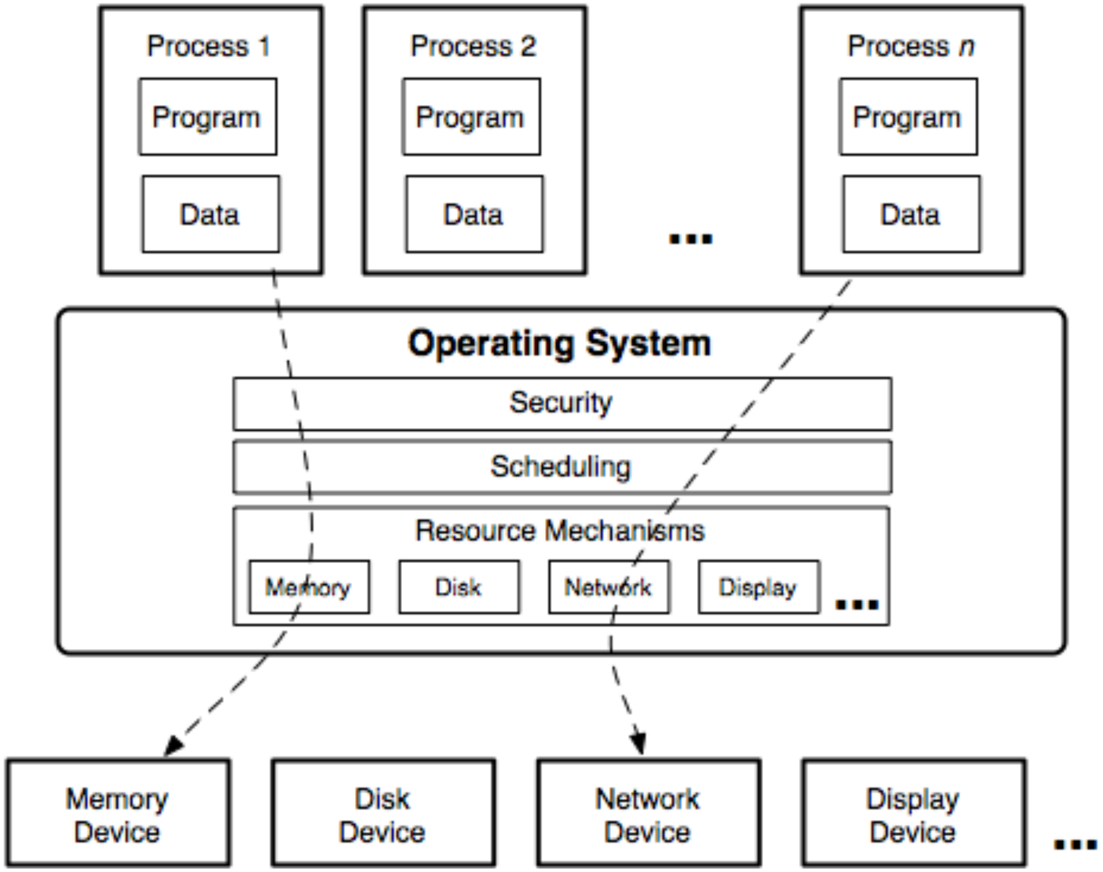
Advanced Systems Security: Introduction to OS Security

Trent Jaeger
Systems and Internet Infrastructure Security (SIIS) Lab
Computer Science and Engineering Department
Pennsylvania State University

Control Bad Code

- 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
- We still have operating systems security to protect the data and other processes on the host
  - Claim: Conventional OS security methods are insufficient
  - Why not?

Operating Systems



Control Bad Code

- What mechanism does an OS use to restrict the rights of processes (i.e., running code) from system resources?

WORRYING = WASTE OF TIME. GOOD AND BAD THINGS WILL HAPPEN IN LIFE. YOU JUST HAVE TO KEEP LIVING AND NOT STRESS OVER WHAT YOU CAN'T CONTROL.

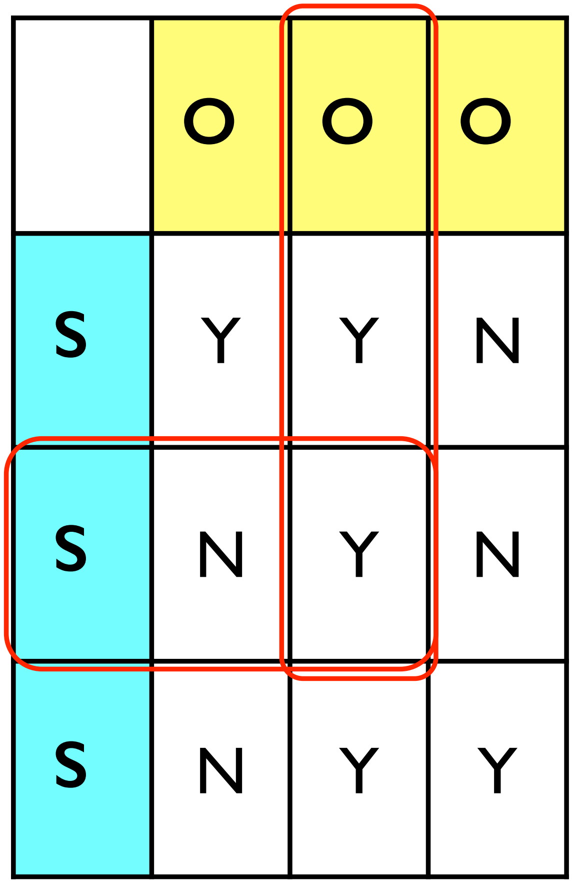
KUSHANDWIZDOM

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 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 access object O_j, compare the request ops to the appropriate cell



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

Access Matrix

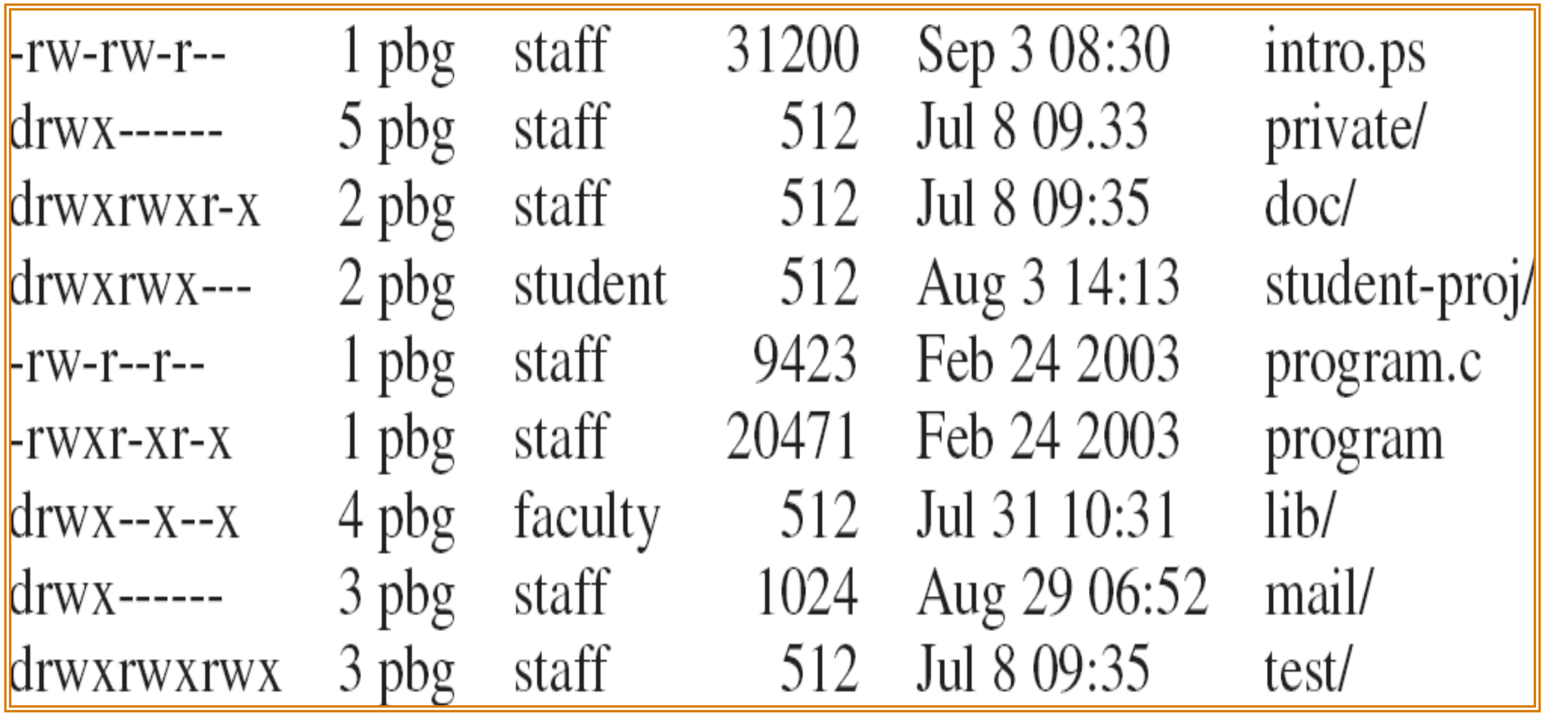
- Using the Access Matrix
- (1) Suppose J wants to prevent other users' processes from reading/writing her private key (object O_1)
- (2) Suppose J wants to prevent other users' processes from writing her public key (object O_2)
- Design the access matrix
- Are these the rights on your host to your SSH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

	O	O	O
J	?	?	?
S	?	?	?
S	?	?	?

UNIX Access Control

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

UNIX Mode Bits

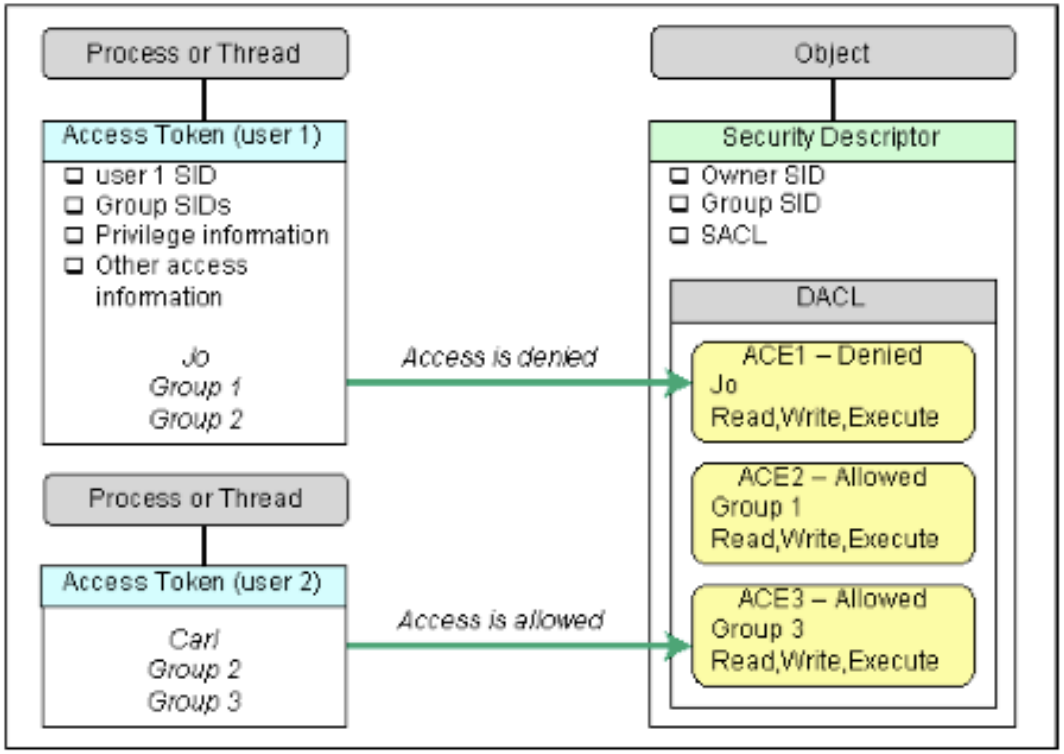


```
-rw-rw-r--   1 pbg  staff    31200  Sep 3 08:30   intro.ps
drwx------   5 pbg  staff      512  Jul 8 09.33   private/
drwxrwxr-x   2 pbg  staff      512  Jul 8 09:35   doc/
drwxrwx---   2 pbg  student    512  Aug 3 14:13   student-proj/
-rw-r--r--   1 pbg  staff     9423  Feb 24 2003   program.c
-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



Process or Thread

Access Token (user 1)

- ❑ user 1 SID
- ❑ Group SIDs
- ❑ Privilege information
- ❑ Other access information

Jo
Group 1
Group 2

Access is denied

Process or Thread

Access Token (user 2)

Carl
Group 2
Group 3

Access is allowed

Object

Security Descriptor

- ❑ Owner SID
- ❑ Group SID
- ❑ SACL

DACL

ACE1 – Denied
Jo
Read,Write,Execute

ACE2 – Allowed
Group 1
Read,Write,Execute

ACE3 – Allowed
Group 3
Read,Write,Execute

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	O	O
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
  - Secrecy and integrity met under *benign* processes
  - Protects against an error by a non-malicious entity
- Security
  - Security goals met under *potentially malicious* processes
  - Enforces requirements even if adversary is in complete control of the process
- Hence, for J: Non-malicious process 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

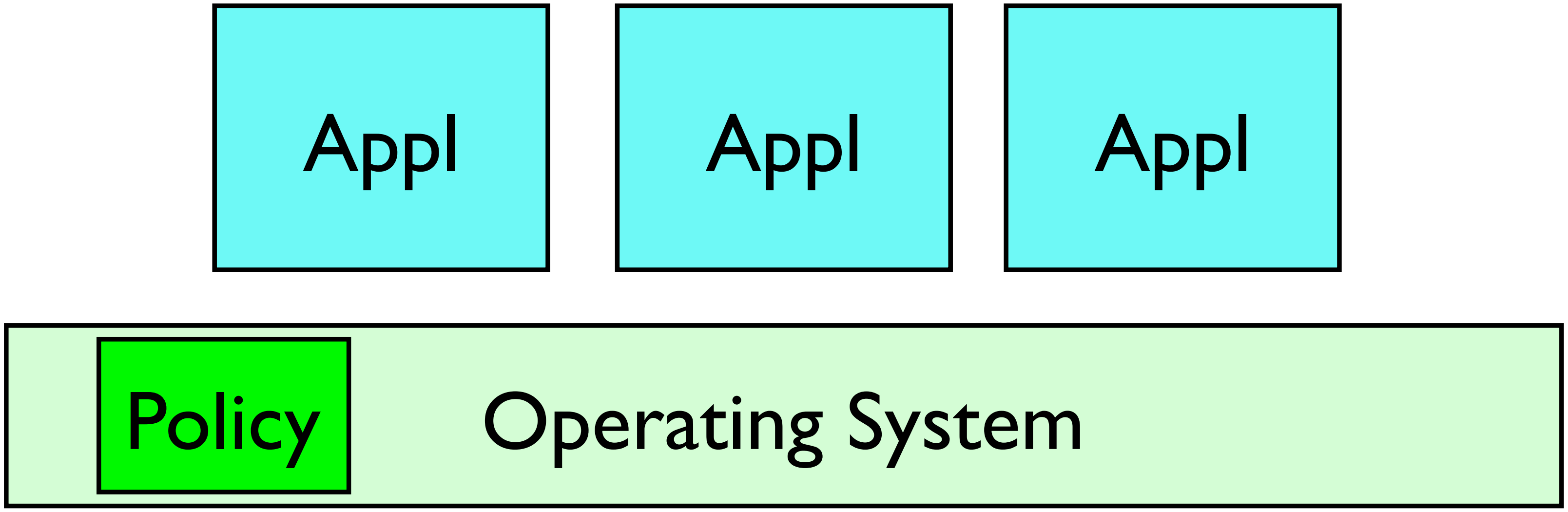
Integrity

- 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

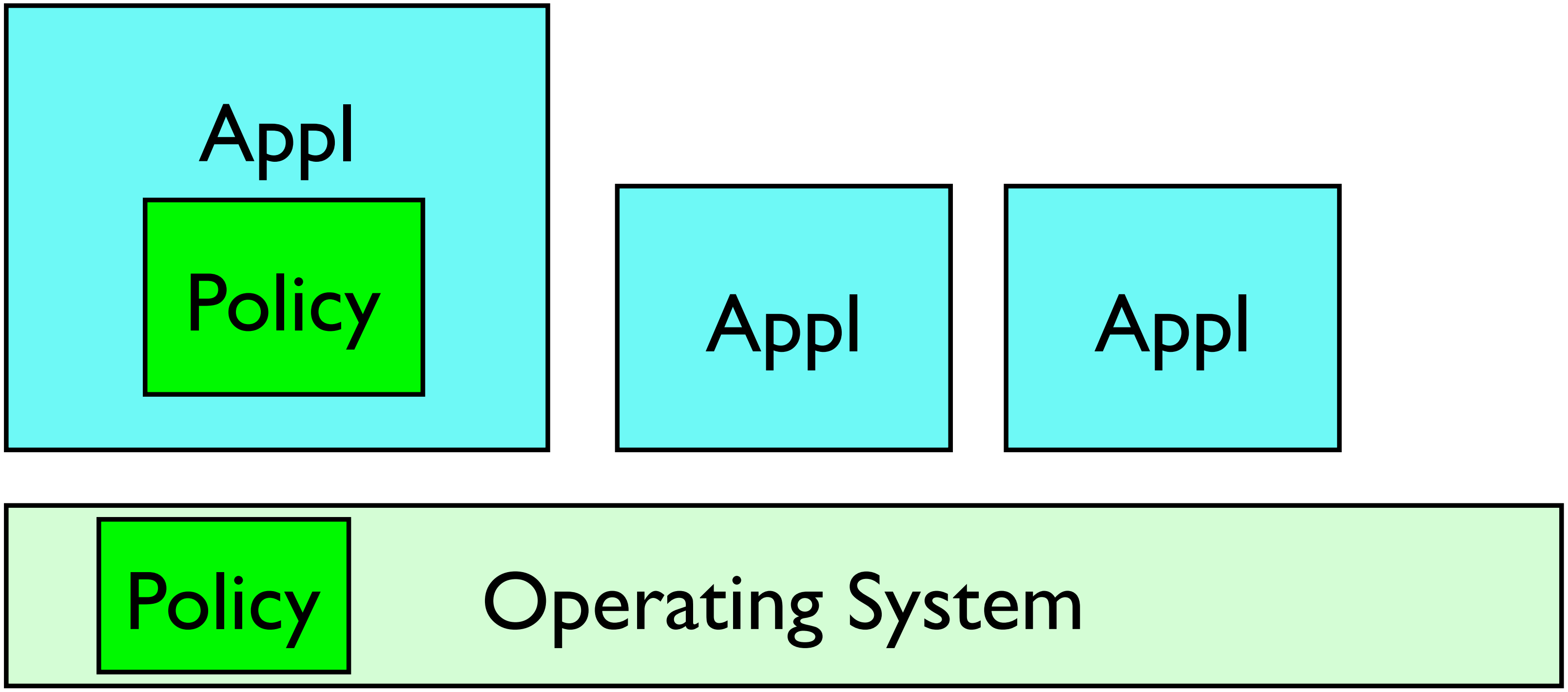
- Process secrecy requires that the process not communicate with unauthorized parties
  - But what about a process that services requests?
  - This is a very difficult requirement to meet
- Suppose a benign process can write to a file controlled by an adversary
- Unless the process is trusted to contain 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
    - Security requirements determined by allowed flows

Policy Enforcement in Apps

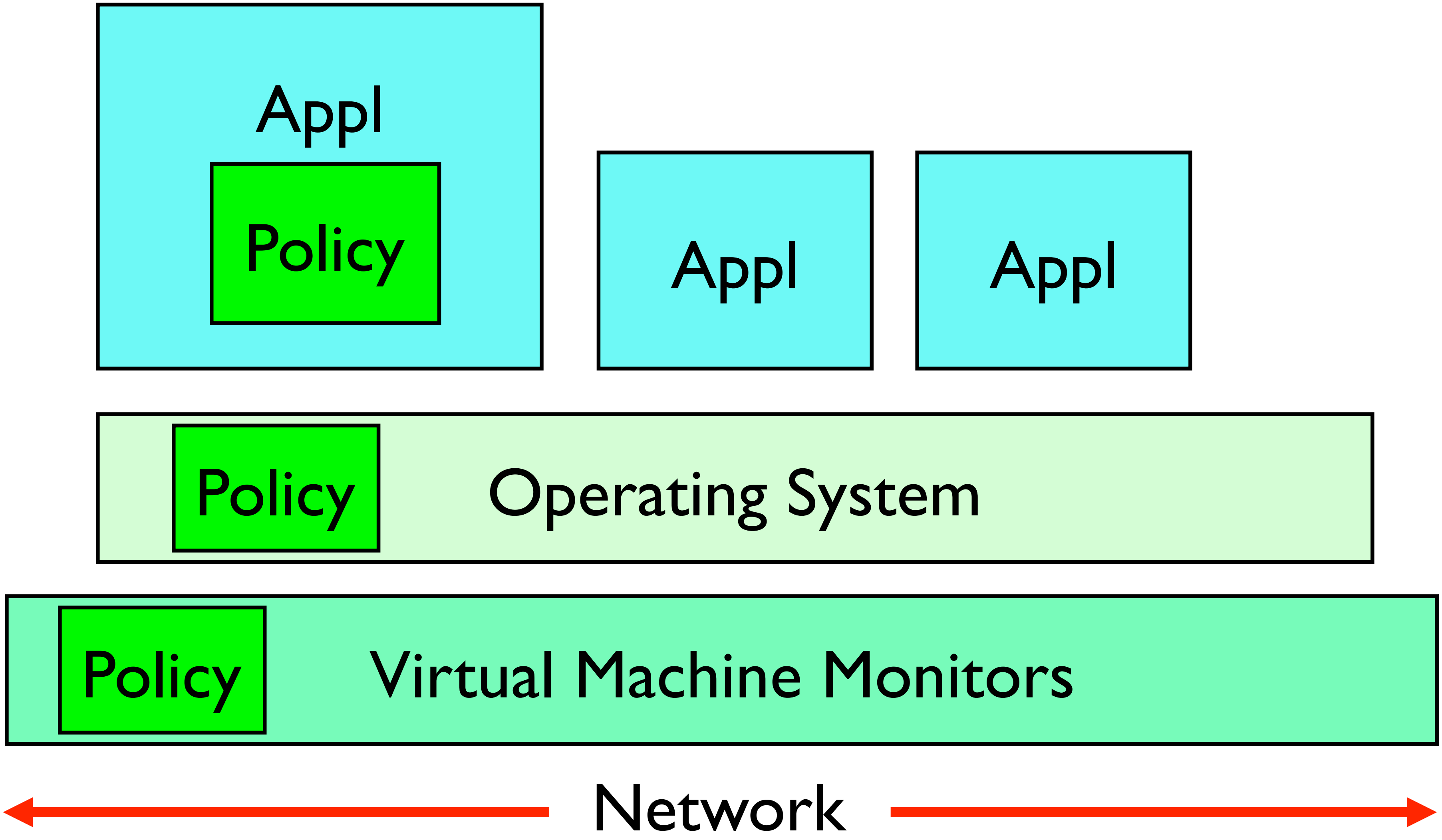


- Application policy enforcement: databases, JVM, X Windows, daemons, browsers, email clients, servers

Security Enforcement

- Several applications include access control
  - *Databases, window servers, web servers, browsers, …*
- Some programming systems include access control to system resources
  - *Java, Safe-Tcl, Ruby, Python, Perl – Jif, Flow Caml (information flow);*
- Some systems recognize that programs may contribute to access control
  - *User-level policy server for SELinux*
  - *Information Flow Control*
- *Requirement: Ensure that all layers are using their authority in a manner consistent with system security goals*

Multi-Layered Enforcement



Questions for This Class

- How do we keep bad code off our systems?
- 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?