Lecture 1, March 30

ECS 235B, Foundations of Computer and Information Security

Spring Quarter 2026

Let’s Get the Hard Stuff Out of the Way

- Adding the class
  - The department policy controls who gets PTAs
  - Available at https://cs.ucdavis.edu/graduate/policies
    - Click on“PTA Process and Expectations”

I can only issue PTAs in accordance with those directions

(that is, only when *the department* asks me)

General Information

- Who we are
  - Instructor: Matt Bishop, mabishop@ucdavis.edu
    Office hours: Tu 12:10pm–1:00pm, ThF 1:30pm–2:20pm
  - TA: Chien Chou, billin@ucdavis.edu; office hours: *to be arranged*
- Goals:
  - Learn about the reference monitor and high assurance systems;
  - Learn about the access control matrix model and its variants, and how it is used to analyze the security of classes of systems;
  - Learn about the mathematics underlying security policies and their composition;
  - Learn about the confinement problem and information flow; and
  - Explore other topics of interest.

General Information

- Prerequisites
  - ECS 235A (Computer and Information Security), or knowledge of security commensurate with it
  - Recommended: ECS 150 (Operating Systems); ECS 120 (Theory of Computation)
- Texts:
  - M. Bishop, *Computer Security: Art and Science*, 2nd Edition, Addison-Wesley Professional, Boston, MA (2018). ISBN: 978-0-321-71233-2.
  - *Recommended*: R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, 3rd Edition, John Wiley & Sons, Inc., New York, NY (2020). ISBN: 978-1-119-64281-7.

General Information

- Class Web Site
  - It's on Canvas; log in there, and go to ECS 235B
  - If you don't have access and need it, please email me and I will add you
    - **This *does not* mean you will be admitted to the course!**
  - Backup web site: http://nob.cs.ucdavis.edu/classes/ecs235b-2026-02
- Grading
  - Homework is 50% total, and I expect 4 or 5 equally weighted assignments
  - Project is 50%, weighted as described in the handout

General Information

- General UC Davis resources
  - Cover health and wellness, virtual classroom fatigue, etc.
  - You can also come to me and I will either help or suggest where you can get help
  - Web site: https://ebeler.faculty.ucdavis.edu/resources/faq-student-resources/
- Academic integrity
  - ***Paramount*!!!** And plagiarism is a violation of academic integrity
  - Follow the UC Davis Code of Academic Conduct
    - Available at https://ossja.ucdavis.edu/code-academic-conduct
  - I report cheating to the Office of Student Support and Judicial affairs; they handle it appropriately

Homework

- Due at 11:55pm on the due date, unless stated otherwise
- Submit it on Canvas
  - We will give you comments and grades on Canvas too
  - None of this will be on the secondary web site; only on Canvas
- Don't be wishy-washy in your answer
- Write it in good English; be clear and concise
  - An example of a good answer is on the web site
- Submit PDF or text files *only*; do *not* submit other formats!
  - Other formats often don't render well across all systems

Homework

- Late policy: I will accept homework up to 5 week days late
  - You will lose 20% per weekday late
  - Example: your homework is due Monday and you turn it in Wednesday; that is 2 days late, so your actual score will be 60% of what it would have been had you turned it in on time
  - Extensions granted on request, for good reason
- Extra credit
  - It is *not* added into your homework score
  - It is used *only* if your grade is on a borderline (for example, right between an A- and an A); it helps determine which one you get

Term Project

- Goal: get you to explore an area of security in depth
- It can be:
  - Detailed survey or research paper
  - Programming project on validating or working with some formalism
  - Implementing a policy model and formally verify your implementation is correct
- Key rule: ***pick something that interests you***
- Team size: no more than 5
  - If for some reason a team wants more than 5 people, talk to me about it
  - The larger the team, the more I will expect from it

Term Project

Due dates:

- *Project selection*: April 17, 2026; 10% of project score
- *Progress report*: May 11, 2026; 20% of project score
- *Final project*: June 8, 2026, by 5:30pm; 70% of project score

Only one team member should submit these

- *Be sure to name all team members in all submissions!*
- Other term members should submit a short note saying who turned it in (one sentence is fine)
  - Please do not put it in the comments field!

Class

- Asking questions: speak up!
  - If you have a question, so does at least half the class

Remember, the only stupid question is the one you want answered and don't ask

Basic Components of Security

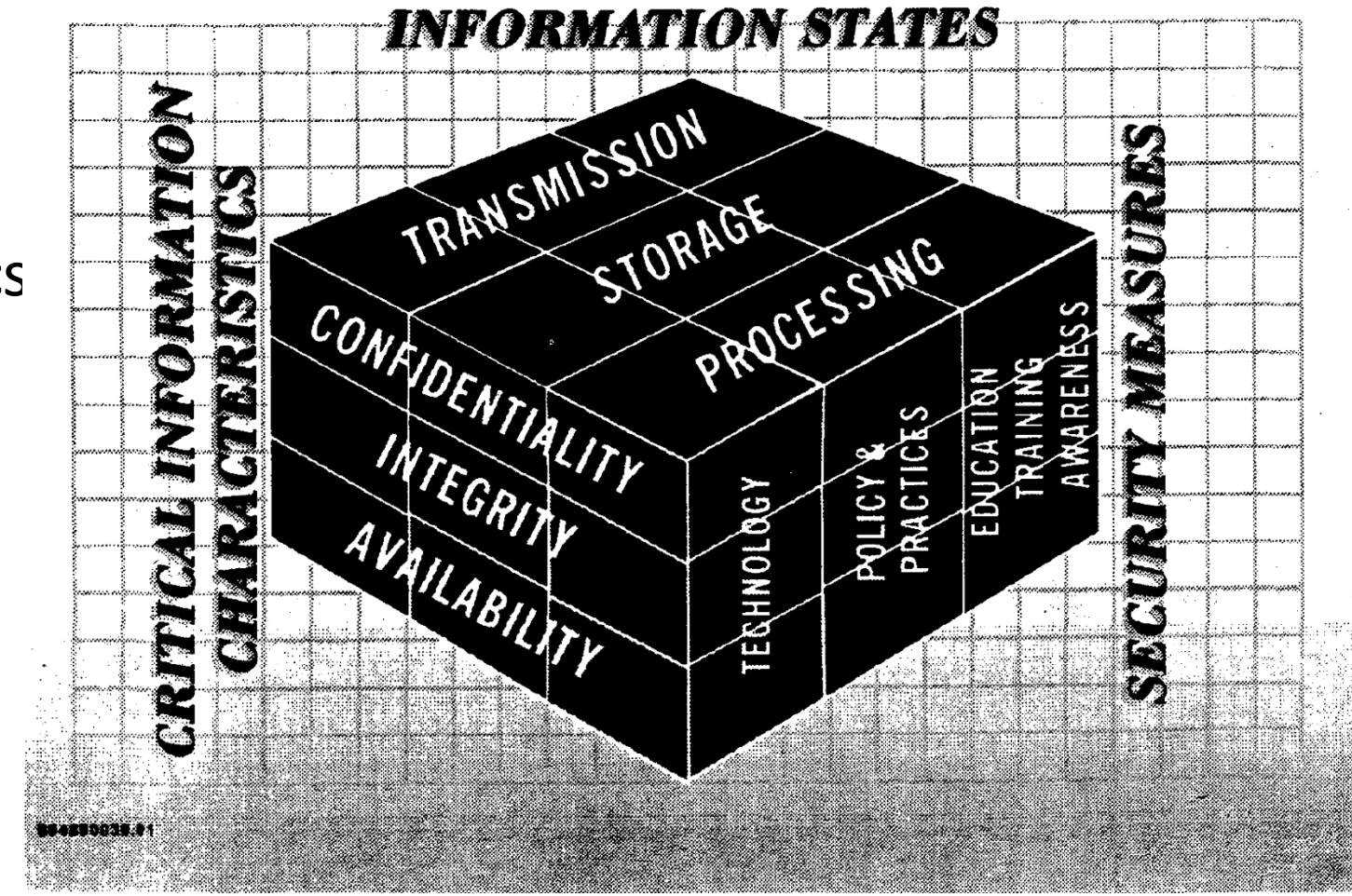
- Confidentiality
  - Keeping data and resources hidden
- Integrity
  - Data integrity (integrity)
  - Origin integrity (authentication)
- Availability
  - Allowing access to data and resources

McCumber Cube

Critical Information Characteristics
- CIA Triad

Information States
- Stages in handling information

Security Measures
- Controls for information access and handling



Picture from J. McCumber, "Information Systems Security: A Comprehensive Model", *Proceedings of the 14th National Computer Security Conference* pp. 328–337 (Oct. 1991); on p. 334.

Information States

- Storage
  - Where data is kept
  - Examples: disks, USB memory sticks
- Transmission
  - How data moves from one place to another
  - Examples: network connections, pipes
- Processing
  - Computations using the information
  - Examples: computing statistics, drawing pictures

Security Measures

- Technology
  - Something implemented and used to ensure critical information characteristics maintained through information states
  - Example: encryption, access controls
- Policy and Practice
  - Something which says what information can be accessed, by whom, and how; a procedure to enhance security
  - Example: students may not access one another's homework files
- Education, Training, and Awareness
  - Make people understand security at level appropriate for them
  - Example: cybersecurity training UC Davis folks must take

Assumptions and Trust

- Underlie *all* aspects of security
- Policies
  - Unambiguously partition system states
  - Correctly capture security requirements
- Mechanisms
  - Assumed to enforce policy
  - Support mechanisms work correctly

Some Terms: Entities

- Subject: active entity
  - Causes information to flow or system state to change
  - Examples: processes, some devices
  - At a higher layer of abstraction: users, other computers
- Object: passive entity
  - Contains or receives information
  - Examples: files, some devices
  - At a higher layer of abstraction: file server, network
- Note: in older papers, objects are entities and so include subjects
  - This has changed in modern papers

Reference Monitor

- *Reference monitor* is access control concept of an abstract machine that mediates all accesses to objects by subjects
- *Reference validation mechanism* (RVM) is an implementation of the reference monitor concept.
  - Tamperproof
  - Complete (always invoked and can never be bypassed)
  - Simple (small enough to be subject to analysis and testing, the completeness of which can be assured)
    - Last engenders trust by providing evidence of correctness
- Note: RVM is almost always called a reference monitor too

Examples

- *Security kernel* combines hardware and software to implement reference monitor
- *Trusted computing base* (*TCB*) consists of all protection mechanisms within a system responsible for enforcing security policy
  - Includes hardware and software
  - Generalizes notion of security kernel

Policy and Reference Monitor

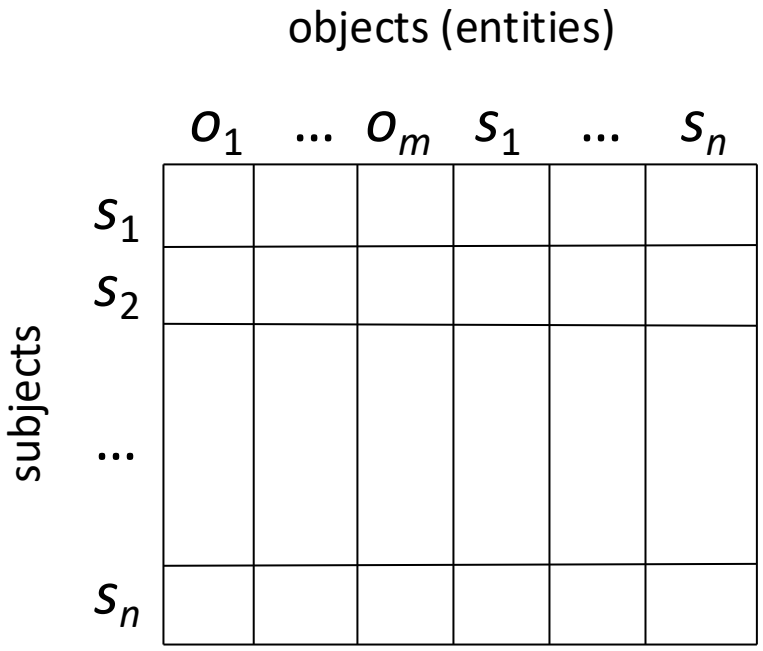
- Reference monitor implements a given policy
  - It has a tamperproof authorization database
  - Also maintains an audit trail (record of security-related events) for review

- More on this later; we need some background first

Access Control Matrix (ACM)

- Basis for representing security policies
- Standard version developed by Harrison, Ruzzo, Ullman to study decidability of security
  - Needed to represent a very general security policy
  - Also a very simple protection system
  - More on this after discussing what we need in an ACM

Description



- Subjects $S = \{ s_1, \ldots, s_n \}$
- Objects $O = \{ o_1, \ldots, o_m \}$
- Rights $R = \{ r_1, \ldots, r_k \}$
- Entries $A[s_i, o_j] \subseteq R$
- $A[s_i, o_j] = \{ r_x, \ldots, r_y \}$ means subject s_i has rights $r_x, \ldots, r_y$ over object o_j

Example 1

- Processes *p*, *q*
- Files *f*, *g*
- Rights *r*, *w*, *x*, *a*, *o*

	f	*g*	*p*	*q*
p	*rwo*	*r*	*rwxo*	*w*
q	*a*	*ro*	*r*	*rwxo*

Example 2

- Host names *telegraph*, *nob*, *toadflax*
- Rights *own*, *ftp*, *nfs*, *mail*

	telegraph	*nob*	*toadflax*
telegraph	*own*	*ftp*	*ftp*
nob		*ftp, mail, nfs, own*	*ftp, nfs, mail*
toadflax		*ftp, mail*	*ftp, mail, nfs, own*

Example 3

- Procedures *inc_ctr*, *dec_ctr*, *manage*
- Variable *counter*
- Rights +, –, *call*

	counter	*inc_ctr*	*dec_ctr*	*manage*
inc_ctr	+			
dec_ctr	–			
manager		*call*	*call*	*call*

UNIX/Linux Access Controls

- Files
  - *A* is ~bishop/a.out (permissions set to 0755, or rwxr-xr-x)
  - *B* is /etc/passwd (permissions set to 0644, or rw-r--r--)
  - *H* is /home/bishop (permissions set to 0711, or rwx--x--x)
  - *S* is /bin/su (permissions set to 4711, or s--rwx--x--x)

	A	*B*	*S*	*H*
bishop	*rwxo*	*r*	*x*	*rwxo*
holly	*rx*	*r*	*x*	*x*
root	*rwx*	*rwo*	*rwxo*	*rwx*

UNIX/Linux Access Controls

- Access control matrices are dynamic:
- After bishop executes `chmod 700 /home/bishop`:
  - Same as chmod u=rwx,g-rwx,o-rwx /home/bishop

	A	*B*	*S*	*H*
bishop	*rwxo*	*r*	*x*	*rwxo*
heidi		*r x*		
root	*rwx*	*rwo*	*rwxo*	*rwx*

Boolean Expression Evaluation

- ACM controls access to database fields
  - Subjects have attributes
  - Verbs define type of access
  - Rules associated with objects, verb pair
- Subject attempts to access object
  - Rule for object, verb evaluated, grants or denies access

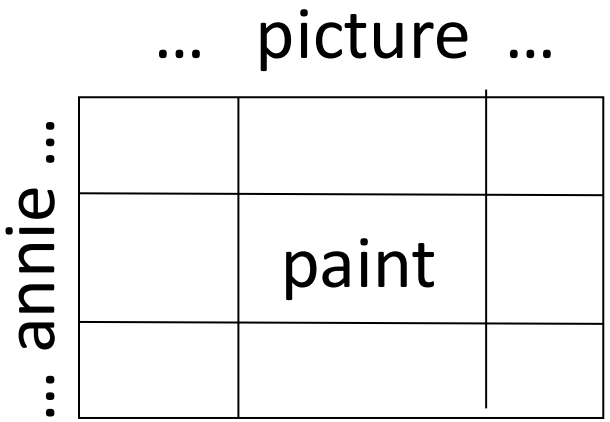
Example

- Subject annie
  - Attributes *role* (artist), *group* (creative)
- Verb paint
  - Default 0 (deny unless explicitly granted)
- Object picture
  - Rule:

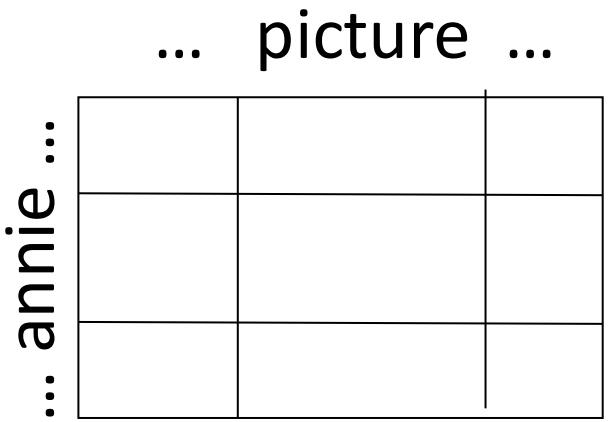
    paint: ‘artist’ in subject.role and
    ‘creative’ in subject.groups and
    time.hour ≥ 0 and time.hour ≤ 4

ACM at 3AM and 10AM

At 3AM, time condition met
ACM is:



At 10AM, time condition not met
ACM is:



History

- Problem: what a process has accessed may affect what it can access now
- Example: procedure in a web applet can access other procedures depending on what procedures it has already accessed
  - *S* set of *static rights* associated with procedure
  - *C* set of *current rights* associated with each executing process
  - When process calls procedure, rights are $S \cap C$

Example Program

```
// This routine has no filesystem access rights
// beyond those in a limited, temporary area
procedure helper_proc()
        return sys_kernel_file

// But this has the right to delete files
program main()
        sys_load_file(helper_proc)
        tmp_file = helper_proc()
        sys_delete_file(tmp_file)
```

- *sys_kernel_file* contains system kernel
- *tmp_file* is in limited area that *helper_proc*() can access it

Before *helper_proc* Called

- Static rights of program

	sys_kernel_file	*tmp_file*
main	delete	delete
helper_proc		delete

- When program starts, current rights:

	sys_kernel_file	*tmp_file*
main	delete	delete
helper_proc		delete
process	delete	delete

After *helper_proc* Called

- Process rights are intersection of static, previous “current” rights:

	sys_kernel_file	*tmp_file*
main	delete	delete
helper_proc		delete
process		delete

State Transitions

- Change the protection state of system
- $\vdash$ represents transition
  - $X_i \vdash_\tau X_{i+1}$: command τ moves system from state X_i to X_{i+1}
  - $X_i \vdash^* Y$: a sequence of commands moves system from state X_i to Y
- Commands often called *transformation procedures*

Primitive Operations

- **create subject** *s*; **create object** *o*
  - Creates new row, column in ACM; creates new column in ACM
- **destroy subject** *s*; **destroy object** *o*
  - Deletes row, column from ACM; deletes column from ACM
- **enter** *r* **into** *A*[*s*, *o*]
  - Adds *r* rights for subject *s* over object *o*
- **delete** *r* **from** *A*[*s*, *o*]
  - Removes *r* rights from subject *s* over object *o*

Create Subject

- Precondition: $s \notin S$
- Primitive command: **create subject** *s*
- Postconditions:
  - $S' = S \cup \{ s \}$, $O' = O \cup \{ s \}$
  - $(\forall y \in O')\ [A'[s, y] = \varnothing]$, $(\forall x \in S')\ [A'[x, s] = \varnothing]$
  - $(\forall x \in S)(\forall y \in O)\ [A'[x, y] = A[x, y]]$

Create Object

- Precondition: $o \notin O$
- Primitive command: **create object** o
- Postconditions:
  - $S' = S$, $O' = O \cup \{ o \}$
  - $(\forall x \in S')\ [A'[x, o] = \varnothing]$
  - $(\forall x \in S)(\forall y \in O)\ [A'[x, y] = A[x, y]]$

Add Right

- Precondition: $s \in S, o \in O$
- Primitive command: **enter** r **into** $A[s, o]$
- Postconditions:
  - $S' = S, O' = O$
  - $A'[s, o] = A[s, o] \cup \{ r \}$
  - $(\forall x \in S')(\forall y \in O' - \{ o \})\ [A'[x, y] = A[x, y]]$
  - $(\forall x \in S' - \{ s \})(\forall y \in O')\ [A'[x, y] = A[x, y]]$

Delete Right

- Precondition: $s \in S$, $o \in O$
- Primitive command: **delete** r **from** $A[s, o]$
- Postconditions:
  - $S' = S$, $O' = O$
  - $A'[s, o] = A[s, o] - \{ r \}$
  - $(\forall x \in S')(\forall y \in O' - \{ o \})\ [A'[x, y] = A[x, y]]$
  - $(\forall x \in S' - \{ s \})(\forall y \in O')\ [A'[x, y] = A[x, y]]$

Destroy Subject

- Precondition: $s \in S$
- Primitive command: **destroy subject** s
- Postconditions:
  - $S' = S - \{ s \}$, $O' = O - \{ s \}$
  - $(\forall y \in O')\ [A'[s, y] = \varnothing]$, $(\forall x \in S')\ [A'[x, s] = \varnothing]$
  - $(\forall x \in S')(\forall y \in O')\ [A'[x, y] = A[x, y]]$

Destroy Object

- Precondition: $o \in O$
- Primitive command: **destroy object** *o*
- Postconditions:
  - $S' = S, O' = O - \{ o \}$
  - $(\forall x \in S')\ [A'[x, o] = \varnothing]$
  - $(\forall x \in S')(\forall y \in O')\ [A'[x, y] = A[x, y]]$

Creating File

- Process *p* creates file *f* with *r* and *w* permission

```
command create•file(p, f)
    create object f;
    enter own into A[p, f];
    enter r into A[p, f];
    enter w into A[p, f];
end
```

Mono-Operational Commands

- Make process *p* the owner of file *g*

```
command make•owner(p, g)
    enter own into A[p, g];
end
```

- Mono-operational command
  - Single primitive operation in this command

Conditional Commands

- Let *p* give *q r* rights over *f*, if *p* owns *f*

```
command grant•read•file•1(p, f, q)
        if own in A[p, f]
        then
                enter r into A[q, f];
end
```

- Monoconditional command
  - Single condition in this command

Conditions with `and`

- Let *p* give *q* *r* and *w* rights over *f*, if *p* owns *f* and *p* has *c* rights over *q*

```
command grant•read•file•2(p, f, q)
        if own in A[p, f] and c in A[p, q]
        then
                enter r into A[q, f];
                enter w into A[q, f];
end
```

- Biconditional command
  - Two conditions in this command

There Is No `or`

- Let *p* give *q* *r* and *w* rights over *f*, if *p* owns *f* or *p* has *c* rights over *q*

```
command grant•read•file•3(p, f, q)
     if own in A[p, f]
     then
          enter r into A[q, f];
          enter w into A[q, f];
end
command grant•read•file•4(p, f, q)
     if c in A[p, q]
     then
          enter r into A[q, f];
          enter w into A[q, f];
end
grant•read•file•3(p, f, q);
grant•read•file•4(p, f, q)
```

General Form

```
command name of command(parameters)
        if condition [ and condition [ and condition … ] ]
        then
                list of commands, primitive operations
end
```

- Only one **`if`**; no nested **`if`**s either
- It must come *before* any primitive operations or subcommands
- No commands may follow the **`if`**
- There is no **`else`**

Copy Flag and Right

- Allows possessor to give rights to another
- Can be a right itself, in which case any other right of the subject can be copied
- Often attached to a right (called a *flag*), so only applies to that right
  - *r* is read right that cannot be copied
  - r^c is read right that can be copied
- Is copy flag copied when giving *r* rights?
  - Depends on model, instantiation of model

Own Right

- Distinguished right allowing possessor to change entries in ACM column
  - So owner of object can add, delete rights for others
  - May depend on what system allows
    - Can't give rights to specific (set of) users
    - Can't pass copy flag to specific (set of) users

Principle of Attenuation of Privilege

- Subjects can't increase their rights, or give rights they do not possess
  - Restricts addition of rights within a system
  - Usually *ignored* for owner
    - Why? Owner gives itself rights, gives them to others, deletes the rights it gave itself