Lecture 18, May 11, 2026

ECS 235B, Foundations of Computer and Information Security

Spring Quarter 2026

CC Requirements

- Requirements divided into classes based on common purposes
- Classes broken into families
- Families made up of components
  - Definitions of detailed requirements, dependent requirements, definition of hierarchy of requirements
- Functional requirements
- Assurance requirements
  - EALs built from these

CC Security Functional Requirements

11 classes, each with at least 1 family; family has:

- Management section with specific information about management issues for subdivisions, requirements of family
- Audit section identifies auditable events
- Hierarchical dependencies
  - Requirement A hierarchical to requirement B if A's functional requirements offer more security, or is more restrictive, than those of B
- Nonhierarchical dependencies also identified
  - May cross classes

CC Security Functional Requirements Classes

- FAU: Security Audit; 6 families
  - Audit automatic response, data generation, analysis, review, storage
- FCO: Communication; 2 families
  - Nonrepudiation of origin, receipt
- FCS: Cryptographic Support; 2 families
  - Cryptographic key management, operation
- FDP: User Data Protection, 13 families
  - Access control policies, information flow policies; object reuse, data authentication, rollback, stored data integrity

CC Security Functional Requirements Classes

- FIA: Identification and Authentication; 6 families
  - Authentication failures, definitions of user attributes, user/subject binding
- FMT: Security Management; 7 families
  - Security attribute and management of TSF and TSF data, roles, attribute expiration
- FPR: Privacy; 4 families
  - Anonymity, pseudonymity, unlinkability, unobservability
- FPT: Protection of Security Functions, 14 families
  - TSF physical protection, trusted recovery, confidentiality, integrity, availability of exported TSF data, replay detection, TSF self-tests

CC Security Functional Requirements Classes

- FRU: Resource Utilization; 3 families
  - Fault tolerance, resource allocation, priority of service
- FTA: TOE Access; 6 families
  - limitations on multiple concurrent sessions, session locking and termination, TOE access history, access banners, system entry constraints
- FTP: Trusted Path; 2 families
  - Inter-TSF channel function, trusted path

Example

- Class FAU has 6 families
  - For each family, management section identifies management functions of class FMT that should be considered
  - For each family, audit section identifies auditable events that must be addressed if component FAU_GEN is selected in the PP or ST
- Component FAU_SSA: security audit analysis; 4 components in it
  - FAU_SSA.1, potential violation analysis component not hierarchical to any other component
  - FAU_SSA.1 depends on requirement FAU_GEN.1 (from another FAU family); so if FAU_SSA.1 selected, also must select FAU_GEN.1

Example

- FAU_SSA.1 has 2 functional requirements
- FAU_SSA.2 has 2 functional requirements
  - It is profile-based anomaly detection
  - Hierarchical to FAU_SSA.1, meaning requirements of FAU_SSA.2 more stringent than those of FAU_SSA.1, subsuming requirements
  - FAU_SSA.2 depends on FIA_UID.1,, requirement for family in another class

CC Security Assurance Requirements Classes

- APR: Protection Profile Evaluation; 6 families
  - One for each section of PP
- ACE: Protection Profile Configuration Evaluation; 8 families
  - Used to evaluate a PP-Configuration
- ASE: Security Target Evaluation; 7 families
  - One for each section of ST
- ADV: Development; 6 families
  - Security architecture, functional specification, implementation representation, TSF internals, TOE design, security policy modeling

CC Security Assurance Requirements Classes

- AGD: Guidance Documentation; 2 families
  - Operational user guidance, preparative procedures
- ALC: Life Cycle; 7 families
  - configuration management capabilities and scope, delivery, development security, flaw remediation, tools and techniques, life cycle definition
- ATE: Tests; 4 families
  - Test coverage, depth, functional tests, independent testing
- AVA: Vulnerabilities Assessment; 1 family
- ACO: Composition; 5 families
  - Composition rationale, development evidence, reliance of dependent component, composed TOE testing, composition vulnerability analysis

Evaluation Assurance Levels (EALs)

- EAL1: *Functionally Tested*; level based on analysis of security functions using functional and interface specifications, and examination of provided guidance documentation
  - Requires unique TOE identification
  - Applicable to systems for which you need some confidence in correct operations, but security threats are not serious
- EAL2: *Structurally Tested*; EAL1 + analysis of basic description of TOE architecture
  - Supported by search for vulnerabilities, evidence of developer testing, and vulnerability analysis to show resistance to basic attacks
  - Applicable to systems requiring low to moderate level of assurance, but complete development record might not be available

Evaluation Assurance Levels (EALs)

- EAL3: *Methodically Tested and Checked*; like EAL2, but security function analysis requires architectural description of TOE design
  - Supported as EAL2, and high-level design for developer to base testing upon, use of development environment controls, configuration management
- EAL4: *Methodically Designed, Tested, and Reviewed*; EAL3 + low-level design, complete interface description, basic modular TOE design, subset of implementation to inputs for security function analysis
  - Supported as EAL3, and implementation representation, vulnerability analysis to show resistance to enhanced-basic attacks
  - Applicable to systems requiring moderate to high level of assurance
  - Likely to be highest EAL level for retrofitting existing product line

Evaluation Assurance Levels (EALs)

- EAL5: *Semiformally Designed and Tested*; like EAL4, but add modular TSF design and full implementation to input for security functional analysis; requires semiformal functional specification, modular high-level design; comprehensive configuration management
  - Applicable to systems requiring high level of assurance
  - Highest EAL level for rigorous commercial development practices supported by moderate amount of computer security engineering
- EAL6: *Semiformally Verified Design and Tested*; like EAL5, but add formal model of security policies, semiformal TOE design and functional specification; methodical vulnerability search to address penetration attackers with high potential
  - Applicable to systems in high-risk situations where protected assets valuable enough to justify cost, effort of development, certification

Evaluation Assurance Levels (EALs)

- EAL7: *Formally Designed and Tested*; formal presentation of functional specification, high-level design; implementation representation used as basis for testing
  - Complete confirmation of developer test results, independent of developer
  - Applicable to systems in extremely high-risk situations, requires substantial security engineering

Comparison of Levels of Trust

TCSEC	ITSEC	CC	Other
D	E0	*no equivalent*	
no equivalent	*no equivalent*	EAL1	Private lab testing
C1	E1	EAL2	OS for FIPS 140-2 L2
C2	E2	EAL3	OS for FIPS 140-2 L3
B1	E3	EAL4	OS for FIPS 140-2 L4
B2	E4	EAL5	
B3	E5	EAL6	
A1	E6	EAL7	

Evaluation Process (in the U.S.)

- Controlled by CC-Evaluation Methodology (CEM) and NIST
  - Evaluations by NIST-accredited commercial labs for fee
- Labs may offer support for vendors preparing for evaluation
  - But staff that do this cannot work as evaluators on the evaluation
- Vendor chooses accredited laboratory
  - Work with vendor to develop baseline schedule
  - Co-ordinate with validating body
- When done, evaluation findings go to validating agency

Evaluation Process (in the U.S.)

- Evaluation of product or system
  - Some schemes require completed ST that has passed all CEM units *before* they agree to evaluate a product or system
  - Other schemes require an ST that is mostly complete
- Evaluation schemes
  - US scheme only accepts evaluations against NIST-approved PP
  - Other schemes may accept evaluations claiming EAL level
- Lab sends evaluation findings to national validating agency
  - This agency determines whether to accept evaluation and award EAL rating

SOG-IS International Cooperation Agreement

- Senior Officials Group Information Systems Security (SOG-IS) agreement: mutual recognition agreement between participating government organizations, agencies in EU or European Free Trade Association (EFTA)
  - Originally signed in 1997
  - Updated in 1999 to incorporate CC

SOG-IS International Cooperation Agreement

- Modified in 2010 to include authorizing (certification producing) and/or consuming (certification recognizing) participation
  - As of 2017, 8 authorizing nations including the UK, France, Germany; 6 consuming participants including Austria, Finland, Poland
  - Recognition of levels above EAL4 limited to approved technical areas
- 2 levels of certificate producers
  - Recognition of CC certificates claiming EAL1–EAL4
  - Recognition of CC certificates at higher levels for defined technical areas with SOG-IS approved scheme for that level

SOG-IS International Cooperation Agreement

- Participants collaborate to:
  - Standardize Common Criteria PPs and certificate policies among CC schemes in Europe
  - Present common position within the CCRA
  - Develop PPs when EU Commission issues IT security-related directive
- Authorizing nations still perform EAL3, EAL4 evaluations
- Two technical areas covered by SOG-IS; PPs developed for products in these areas
  - Smartcards, similar devices like passports
  - Hardware devices with security boxes

Common Criteria Users Forum

- Common Criteria Users Forum (CCUF): international group composed of people from academia, consultants, users, governments, CC laboratories, vendors, etc.
- Promotes worldwide recognition of:
  - CC evaluations
  - Focused technical communities to develop cPPs
  - Policies, processes for maintaining evaluation on future versions of product
  - Policies, processes for evaluating systems composed of evaluated products
- Governance: CCUF Management Board

CC Discussion

- Much more complete than functional requirements of previous evaluation technologies
- PP or ST may not be as strong as TCSEC classes
  - Fewer experts have reviewed it
  - Not yet faced test of time
- Some CC requirements derived from requirements of previous methodologies
  - These may have more credibility than other requirements
- Ultimately correctness of ST is up to vendor, evaluation team

CC Discussion

- CC project board manages interpretations
  - Any national scheme can submit their interpretations
  - Final interpretations become required on all subsequent evaluations
  - “Criteria creep”: newer evaluations may have to meet more stringent requirements than older evaluations
- Evaluation process monitored by validating body

CC Discussion

- CC documentation, methodology are evolving
  - 8 official versions of CC/CEM so far
  - New technical committees form, continue to develop cPPs
- Common Criteria Management Board (CCMB): international body that maintains CC, ensures CCRA is operated as defined by its rules
  - Each signatory of CCRA has representative on CCMB
  - CCMB discusses change proposals from CCRA participants; determination I Agreed (worthy of being adopted internationally by CC), Concurred (proposal acceptable, does not violate mutual recognition, but not worthy of international adoption), Disagreed (proposal violates mutual recognition rules or too incomplete to be accepted)

System Security Engineering Capability Maturity Model (SSE-CMM)

- Methodology for developing secure systems
  - Based on Software Engineering Capability Maturity Model (SE-CMM)
  - It focuses on processes used to develop system
- Provides maturity levels
  - Contrast with previous ones, which provide trust levels
- Can provide assurance evidence, thereby increasing confidence in trustworthiness of product
- Organized into processes, maturity levels

SSE-CMM Model: Processes

- *Process capability*: range of expected results that can be achieved by following process
  - Indicates potential; a predictor of future project outcomes
- *Process performance*: measure of actual results achieved
- *Process maturity*: extent to which process is explicitly defined, managed, measured, controlled, effective

SSE-CMM Model: Processes

11 systems security engineering process areas:

- Administrator security controls
- Assess impact
- Assess security risk
- Assess threat
- Assess vulnerability
- Build assurance argument
- Coordinate security
- Monitor system security posture
- Provide security input
- Specify security needs
- Verify and validate security

Definition of each process area contains goal, set of supporting base practices (total of 61 base practices within all areas)

Example: Assess Threat Process Area

- Goal: threats to security of system be identified, characterized
- Base processes:
  - Identify Natural Threats
  - Identify Human-Made Threats
  - Identify Threat Units of Measure
  - Assess Threat Agent Capability
  - Assess Threat Likelihood
  - Monitor Threats and Their Characteristics

SSE-CMM Model: Project Practices

11 process areas for project, organizational practices (adapted from SE-CMM):

- Ensure quality
- Manage configuration
- Manage project risk
- Monitor and control technical effort
- Plan technical effort
- Define organization's system engineering process
- Improve organization's system engineering process
- Manage product line evolution
- Manage systems enfineering support environment
- Provide ongoing skills and Knowledge
- Coordinate with suppliers

SSE-CMM: Capability Maturity Levels

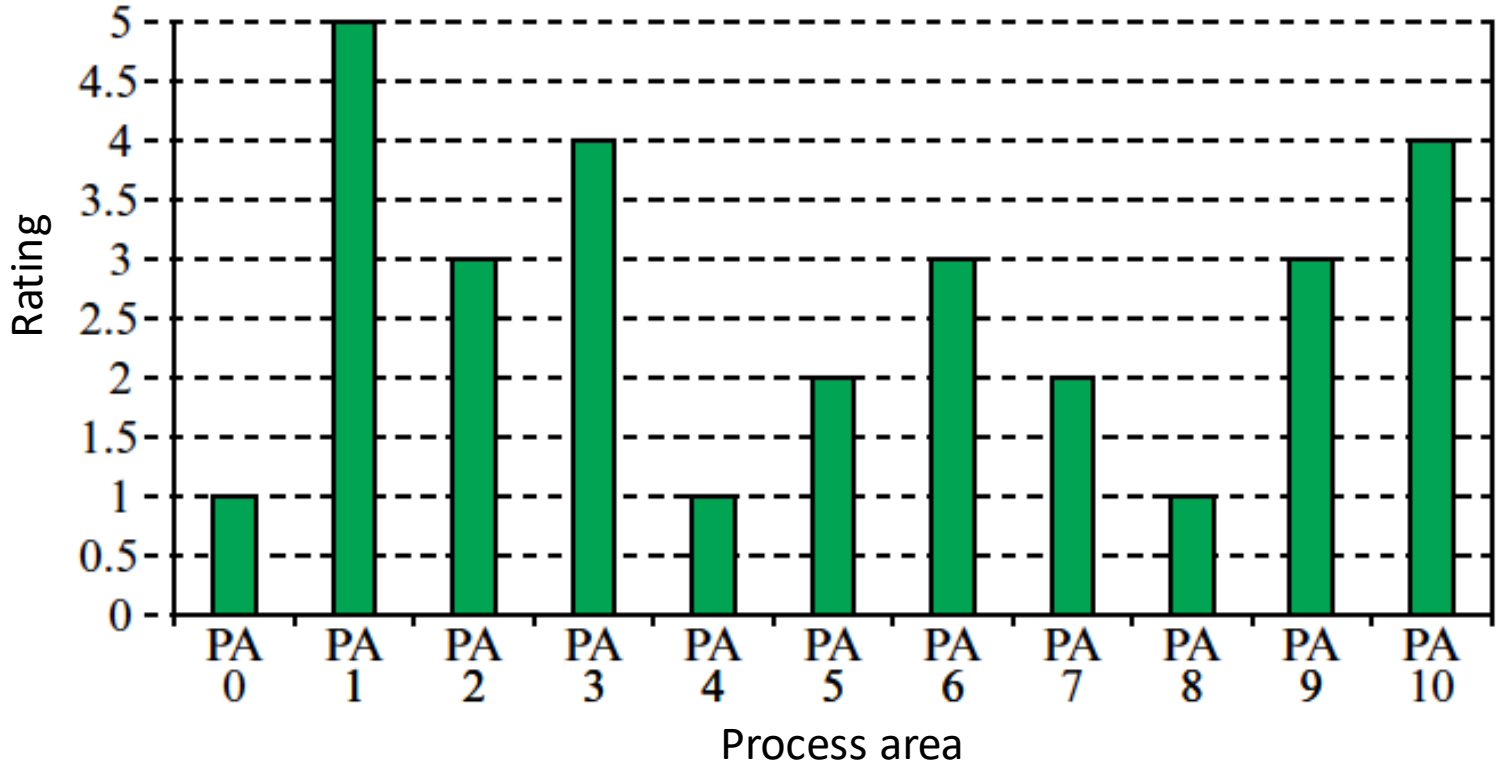
- *Performed informally*: base processes are performed
- *Planned and tracked*: project-level definition, planning, performance verification issues addressed
- *Well-defined*: defining, refining standard practice, coordinating it across organization
- *Quantitatively controlled*: establishing measurable quality goals, objectively managing their performance
- *Continuously improving*: improve organizational capability and process effectiveness

Using SSE-CMM

- Straightforward analysis of existing processes
  - See which base processes have been met
  - See which maturity levels achieved
- Pick project area
- Identify area goals, base processes defined for that process area
  - If all present, assess processes against capability maturity levels
- Result is identification of current level of maturity for each base process in process area
- Repeat for each process area; level of maturity is lowest level represented by set of levels of base process

Rating Profile

- Tabular representation of process areas vs. maturity levels



Key Points

- First public, widely used evaluation technique was TCSEC
  - Led to much research, development of other approaches addressing concerns about TCSEC
- Other methodologies:
  - ITSEC in Europe, CTCPEC in Canada, FC in US
- These led to Common Criteria international evaluation methodology
- Other current evaluation techniques
  - FIPS 140-2 (cryptographic modules, managed by US NIST, Canadian CSE)
  - SSE-CMM (process oriented)

Review of Entropy

- Random variables
- Joint probability
- Conditional probability
- Entropy (or uncertainty in bits)
- Joint entropy
- Conditional entropy
- Applying it to secrecy of ciphers

Random Variable

- Variable that represents outcome of an event
  - X represents value from roll of a fair die; probability for rolling n: $p(X=n) = 1/6$
  - If die is loaded so 2 appears twice as often as other numbers, $p(X=2) = 2/7$ and, for $n \neq 2$, $p(X=n) = 1/7$
- Note: $p(X)$ means specific value for X doesn't matter
  - Example: all values of X are equiprobable

Joint Probability

- Joint probability of X and Y, $p(X, Y)$, is probability that X and Y simultaneously assume particular values
  - If X, Y independent, $p(X, Y) = p(X)p(Y)$
- Roll die, toss coin
  - $p(X=3, Y=\text{heads}) = p(X=3)p(Y=\text{heads}) = 1/6 \times 1/2 = 1/12$

Two Dependent Events

- X = roll of red die, Y = sum of red, blue die rolls

  $p(Y=2) = 1/36$ $p(Y=3) = 2/36$ $p(Y=4) = 3/36$ $p(Y=5) = 4/36$

  $p(Y=6) = 5/36$ $p(Y=7) = 6/36$ $p(Y=8) = 5/36$ $p(Y=9) = 4/36$

  $p(Y=10) = 3/36$ $p(Y=11) = 2/36$ $p(Y=12) = 1/36$

- Formula:

  $p(X=1, Y=11) = p(X=1)p(Y=11) = (1/6)(2/36) = 1/108$

- But if the red die (X) rolls 1, the most their sum (Y) can be is 7
- The problem is X and Y are dependent

Conditional Probability

- Conditional probability of X given Y, $p(X \mid Y)$, is probability that X takes on a particular value given Y has a particular value
- Continuing example …
  - $p(Y=7 \mid X=1) = 1/6$
  - $p(Y=7 \mid X=3) = 1/6$

Relationship

- $p(X, Y) = p(X \mid Y)\, p(Y) = p(X)\, p(Y \mid X)$
- Example:

  $p(X{=}3, Y{=}8) = p(X{=}3 \mid Y{=}8)\, p(Y{=}8) = (1/5)(5/36) = 1/36$
- Note: if X, Y independent:

  $p(X \mid Y) = p(X)$

Entropy

- Uncertainty of a value, as measured in bits
- Example: X value of fair coin toss; X could be heads or tails, so 1 bit of uncertainty
  - Therefore entropy of X is $H(X) = 1$
- Formal definition: random variable X, values x_1, …, x_n; so

  Σ_i p($X = x_i$) = 1; then entropy is:

$$H(X) = -\Sigma_i\, p(X{=}x_i)\, \lg\, p(X{=}x_i)$$

Heads or Tails?

- $H(X) = -p(X=\text{heads}) \lg p(X=\text{heads}) - \text{p}(X=\text{tails}) \lg p(X=\text{tails})$

  $= -(1/2) \lg (1/2) - (1/2) \lg (1/2)$

  $= -(1/2)(-1) - (1/2)(-1) = 1$
- Confirms previous intuitive result

n-Sided Fair Die

$H(X) = -\Sigma_i\, p(X = x_i) \lg p(X = x_i)$

As $p(X = x_i) = 1/n$, this becomes

$H(X) = -\Sigma_i\, (1/n) \lg (1/ n) = -n(1/n) (-\lg n)$

so

$H(X) = \lg n$

which is the number of bits in n, as expected

Ann, Pam, and Paul

Ann, Pam twice as likely to win as Paul

W represents the winner. What is its entropy?

- w_1 = Ann, w_2 = Pam, w_3 = Paul
- $p(W=w_1) = p(W=w_2) = 2/5$, $p(W=w_3) = 1/5$

- So $H(W) = -\sum_i p(W=w_i) \lg p(W=w_i)$

  $= -(2/5) \lg (2/5) - (2/5) \lg (2/5) - (1/5) \lg (1/5)$

  $= -(4/5) + \lg 5 \approx 1.52$

- If all equally likely to win, $H(W) = \lg 3 \approx 1.58$

Joint Entropy

- X takes values from $\{ x_1, \ldots, x_n \}$, and $\Sigma_i \, p(X=x_i) = 1$
- Y takes values from $\{ y_1, \ldots, y_m \}$, and $\Sigma_i \, p(Y=y_i) = 1$
- Joint entropy of X, Y is:

$$H(X, Y) = -\Sigma_j \, \Sigma_i \, p(X=x_i, Y=y_j) \lg p(X=x_i, Y=y_j)$$

Example

X: roll of fair die, *Y*: flip of coin

As *X*, *Y* are independent:

$$p(X=1, Y=\text{heads}) = p(X=1)\ p(Y=\text{heads}) = 1/12$$

and

$$H(X, Y) = -\sum_j \sum_i p(X=x_i, Y=y_j) \lg p(X=x_i, Y=y_j)$$

$$= -2\ [\ 6\ [\ (1/12) \lg (1/12)\ ]\ ] = \lg 12$$

Conditional Entropy (Equivocation)

- X takes values from $\{ x_1, \ldots, x_n \}$ and $\Sigma_i\, p(X=x_i) = 1$
- Y takes values from $\{ y_1, \ldots, y_m \}$ and $\Sigma_i\, p(Y=y_i) = 1$
- Conditional entropy of X given $Y=y_j$ is:

$$H(X \mid Y=y_j) = -\Sigma_i\, p(X=x_i \mid Y=y_j)\ \lg\ p(X=x_i \mid Y=y_j)$$

- Conditional entropy of X given Y is:

$$H(X \mid Y) = -\Sigma_j\, p(Y=y_j)\ \Sigma_i\, p(X=x_i \mid Y=y_j)\ \lg\ p(X=x_i \mid Y=y_j)$$

Example

- X roll of red die, Y sum of red, blue roll
- Note $p(X=1 \mid Y=2) = 1$, $p(X=i \mid Y=2) = 0$ for $i \neq 1$
  - If the sum of the rolls is 2, both dice were 1
- Thus

$$H(X \mid Y=2) = -\sum_i p(X=x_i \mid Y=2) \lg p(X=x_i \mid Y=2) = 0$$

Example (*con't*)

- Note $p(X=i, Y=7) = 1/6$
  - If the sum of the rolls is 7, the red die can be any of 1, …, 6 and the blue die must be 7–roll of red die
- $H(X|Y=7) = -\sum_i p(X=x_i|Y=7) \lg p(X=x_i|Y=7)$

$$= -6\ (1/6) \lg (1/6) = \lg 6$$

Example: Perfect Secrecy

- Cryptography: knowing the ciphertext does not decrease the uncertainty of the plaintext
- $M = \{ m_1, \ldots, m_n \}$ set of messages
- $C = \{ c_1, \ldots, c_n \}$ set of messages
- Cipher $c_i = E(m_i)$ achieves *perfect secrecy* if $H(M \mid C) = H(M)$

Review of Information Flow Notation

- Bell-LaPadula Model embodies information flow policy
  - Given compartments *A*, *B*, info can flow from *A* to *B* iff *B dom A*
- So does Biba Model
  - Given compartments *A*, *B*, info can flow from *A* to *B* iff *A dom B*
- Variables *x*, *y* assigned compartments $\underline{x}$, $\underline{y}$ as well as values
  - Confidentiality (Bel-LaPadula): if $\underline{x}$ = *A*, $\underline{y}$ = *B*, and *B dom A*, then *y* := *x* allowed but not *x* := *y*
  - Integrity (Biba): if $\underline{x}$ = *A*, $\underline{y}$ = *B*, and *A dom B*, then *x* := *y* allowed but not *y* := *x*
- For now, focus on confidentiality (Bell-LaPadula)
  - We'll get to integrity later

Entropy and Information Flow

- Idea: information flows from *x* to *y* as a result of a sequence of commands *c* if you can deduce information about *x* before *c* from the value in *y* after *c*
- Formally:
  - *s* time before execution of *c*, *t* time after
  - $H(x_s \mid y_t) < H(x_s \mid y_s)$
  - If no *y* at time *s*, then $H(x_s \mid y_t) < H(x_s)$

Example 1

- Command is $x := y + z$; where:
  - x does not exist initially (that is, has no value)
  - $0 \le y \le 7$, equal probability
  - $z = 1$ with probability 1/2, $z = 2$ or 3 with probability 1/4 each
- s state before command executed; t, after; so
  - $H(y_s) = H(y_t) = -8(1/8) \lg (1/8) = 3$
- You can show that $H(y_s \mid x_t) = (3/32) \lg 3 + 9/8 \approx 1.274 < 3 = H(y_s)$
  - Thus, information flows from y to x

Example 2

- Command is

  if $x = 1$ **then** $y := 0$ **else** $y := 1$;

  where x, y equally likely to be either 0 or 1
- $H(x_s) = 1$ as x can be either 0 or 1 with equal probability
- $H(x_s \mid y_t) = 0$ as if $y_t = 1$ then $x_s = 0$ and vice versa
  - Thus, $H(x_s \mid y_t) = 0 < 1 = H(x_s)$
- So information flowed from x to y

Implicit Flow of Information

- Information flows from *x* to *y* without an *explicit* assignment of the form *y* := *f*(*x*)
  - *f*(*x*) an arithmetic expression with variable *x*
- Example from previous slide:

if *x* = 1 **then** *y* := 0 **else** *y* := 1;

- So must look for implicit flows of information to analyze program

Notation

- $\underline{x}$ means class of x
  - In Bell-LaPadula based system, same as “label of security compartment to which x belongs”
- $\underline{x} \leq \underline{y}$ means “information can flow from an element in class of x to an element in class of y
  - Or, “information with a label placing it in class $\underline{x}$ can flow into class $\underline{y}$”

Information Flow Policies

Information flow policies are usually:

- reflexive
  - So information can flow freely among members of a single class
- transitive
  - So if information can flow from class 1 to class 2, and from class 2 to class 3, then information can flow from class 1 to class 3

Non-Transitive Policies

- Betty is a confident of Anne
- Cathy is a confident of Betty
  - With transitivity, information flows from Anne to Betty to Cathy
- Anne confides to Betty she is having an affair with Cathy's spouse
  - Transitivity undesirable in this case, probably

Non-Lattice Transitive Policies

- 2 faculty members co-PIs on a grant
  - Equal authority; neither can overrule the other
- Grad students report to faculty members
- Undergrads report to grad students
- Information flow relation is:
  - Reflexive and transitive
- But some elements (people) have no “least upper bound” element
  - What is it for the faculty members?

Confidentiality Policy Model

- Lattice model fails in previous 2 cases
- Generalize: policy $I = (SC_I, \leq_I, join_I)$:
  - SC_I set of security classes
  - $\leq_I$ ordering relation on elements of SC_I
  - $join_I$ function to combine two elements of SC_I
- Example: Bell-LaPadula Model
  - SC_I set of security compartments
  - $\leq_I$ ordering relation *dom*
  - $join_I$ function *lub*

Confinement Flow Model

- $(I, O, confine, \rightarrow)$
  - $I = (SC_I, \leq_I, join_I)$
  - O set of entities
  - $\rightarrow$: $O \times O$ with $(a, b) \in \rightarrow$ (written $a \rightarrow b$) iff information can flow from a to b
  - for $a \in O$, $confine(a) = (a_L, a_U) \in SC_I \times SC_I$ with $a_L \leq_I a_U$
    - Interpretation: for $a \in O$, if $x \leq_I a_U$, information can flow from x to a, and if $a_L \leq_I x$, information can flow from a to x
    - So a_L lowest classification of information allowed to flow out of a, and a_U highest classification of information allowed to flow into a

Assumptions, *etc.*

- Assumes: object can change security classes
  - So, variable can take on security class of its data
- Object x has security class $\underline{x}$ currently
- Note transitivity *not* required
- If information can flow from a to b, then b dominates a under ordering of policy I:

  $(\forall\ a, b \in O)[\ a \rightarrow b \Rightarrow a_L \leq_I b_U\ ]$

Example 1

- SC_I = { U, C, S, TS }, with U $\leq_I$ C, C $\leq_I$ S, and S $\leq_I$ TS
- $a, b, c \in O$
  - confine(a) = [C, C]
  - confine(b) = [S, S]
  - confine(c) = [TS, TS]
- Secure information flows: $a \rightarrow b$, $a \rightarrow c$, $b \rightarrow c$
  - As $a_L \leq_I b_U$, $a_L \leq_I c_U$, $b_L \leq_I c_U$
  - Transitivity holds

Example 2

- SC_I, $\leq_I$ as in Example 1
- $x, y, z \in O$
  - confine(x) = [C, C]
  - confine(y) = [S, S]
  - confine(z) = [C, TS]
- Secure information flows: $x \rightarrow y$, $x \rightarrow z$, $y \rightarrow z$, $z \rightarrow x$, $z \rightarrow y$
  - As $x_L \leq_I y_U$, $x_L \leq_I z_U$, $y_L \leq_I z_U$, $z_L \leq_I x_U$, $z_L \leq_I y_U$
  - Transitivity does not hold
    - $y \rightarrow z$ and $z \rightarrow x$, but $y \rightarrow x$ is false, because $y_L \leq_I x_U$ is false

Transitive Non-Lattice Policies

- Q = (S_Q, $\leq_Q$) is a *quasi-ordered set* when $\leq_Q$ is transitive and reflexive over S_Q
- How to handle information flow?
  - Define a partially ordered set containing quasi-ordered set
  - Add least upper bound, greatest lower bound to partially ordered set
  - It's a lattice, so apply lattice rules!

In Detail …

- $\forall x \in S_Q$: let $f(x) = \{ y \mid y \in S_Q \wedge y \leq_Q x \}$
  - Define $S_{QP} = \{ f(x) \mid x \in S_Q \}$
  - Define $\leq_{QP} = \{ (x, y) \mid x, y \in S_{QP} \wedge x \subseteq y \}$
    - S_{QP} partially ordered set under $\leq_{QP}$
    - f preserves order, so $y \leq_Q x$ iff $f(x) \leq_{QP} f(y)$
- Add upper, lower bounds
  - $S_{QP}' = S_{QP} \cup \{ S_Q, \varnothing \}$
  - Upper bound $ub(x, y) = \{ z \mid z \in S_{QP} \wedge x \subseteq z \wedge y \subseteq z \}$
  - Least upper bound $lub(x, y) = \cap ub(x, y)$
    - Lower bound, greatest lower bound defined analogously

And the Policy Is …

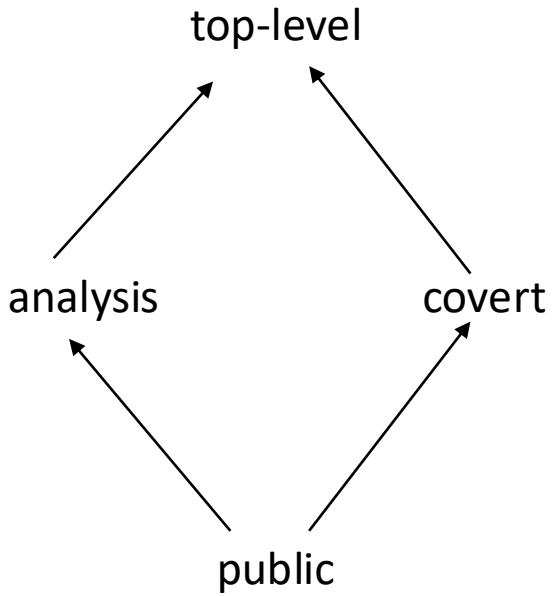
- Now (S_{QP}', $\leq_{QP}$) is lattice
- Information flow policy on quasi-ordered set emulates that of this lattice!

Nontransitive Flow Policies

- Government agency information flow policy (on next slide)
- Entities public relations officers PRO, analysts A, spymasters S
  - *confine*(PRO) = [public, analysis]
  - *confine*(A) = [analysis, top-level]
  - *confine*(S) = [covert, top-level]

Information Flow

- By confinement flow model:
  - PRO ≤ A, A ≤ PRO
  - PRO ≤ S
  - A ≤ S, S ≤ A
- Data *cannot* flow to public relations officers; not transitive
  - S ≤ A, A ≤ PRO
  - S ≤ PRO is *false*



Transforming Into Lattice

- Rough idea: apply a special mapping to generate a subset of the power set of the set of classes
  - Done so this set is partially ordered
  - Means it can be transformed into a lattice
- Can show this mapping preserves ordering relation
  - So it preserves non-orderings and non-transitivity of elements corresponding to those of original set

Dual Mapping

- $R = (SC_R, \leq_R, join_R)$ reflexive info flow policy
- $P = (S_P, \leq_P)$ ordered set
  - Define *dual mapping* functions $l_R, h_R: SC_R \rightarrow S_P$
    - $l_R(x) = \{ x \}$
    - $h_R(x) = \{ y \mid y \in SC_R \wedge y \leq_R x \}$
  - S_P contains subsets of SC_R; $\leq_P$ subset relation
  - Dual mapping function *order preserving* iff

$$(\forall a, b \in SC_R)[\ a \leq_R b \Leftrightarrow l_R(a) \leq_P h_R(b)\ ]$$

Theorem

Dual mapping from reflexive information flow policy R to ordered set P order-preserving

Proof sketch: all notation as before

($\Rightarrow$) Let $a \leq_R b$. Then $a \in l_R(a)$, $a \in h_R(b)$, so $l_R(a) \subseteq h_R(b)$, or $l_R(a) \leq_P h_R(b)$

($\Leftarrow$) Let $l_R(a) \leq_P h_R(b)$. Then $l_R(a) \subseteq h_R(b)$. But $l_R(a) = \{ a \}$, so $a \in h_R(b)$, giving $a \leq_R b$

Information Flow Requirements

- Interpretation: let *confine*(x) = [$\underline{x}_L$, $\underline{x}_U$], consider class $\underline{y}$
  - Information can flow from x to element of $\underline{y}$ iff $\underline{x}_L \leq_R \underline{y}$, or $l_R(\underline{x}_L) \subseteq h_R(\underline{y})$
  - Information can flow from element of $\underline{y}$ to x iff y $\leq_R \underline{x}_U$, or $l_R(\underline{y}) \subseteq h_R(\underline{x}_U)$

Revisit Government Example

- Information flow policy is *R*
- Flow relationships among classes are:

public $\leq_R$ public

public $\leq_R$ analysis

public $\leq_R$ covert

public $\leq_R$ top-level

analysis $\leq_R$ top-level

analysis $\leq_R$ analysis

covert $\leq_R$ covert

covert $\leq_R$ top-level

top-level $\leq_R$ top-level

Dual Mapping of *R*

- Elements l_R, h_R:
  l_R(public) = { public }
  h_R(public = { public }
  l_R(analysis) = { analysis }
  h_R(analysis) = { public, analysis }
  l_R(covert) = { covert }
  h_R(covert) = { public, covert }
  l_R(top-level) = { top-level }
  h_R(top-level) = { public, analysis, covert, top-level }

confine

- Let *p* be entity of type PRO, *a* of type A, *s* of type S
- In terms of *P* (not *R*), we get:
  - *confine*(*p*) = [{ public }, { public, analysis }]
  - *confine*(*a*) = [{ analysis }, { public, analysis, covert, top-level }]
  - *confine*(*s*) = [{ covert }, { public, analysis, covert, top-level }]

And the Flow Relations Are …

- $p \rightarrow a$ as $l_R(p) \subseteq h_R(a)$
  - $l_R(p)$ = { public }
  - $h_R(a)$ = { public, analysis, covert, top-level }
- Similarly: $a \rightarrow p$, $p \rightarrow s$, $a \rightarrow s$, $s \rightarrow a$
- But $s \rightarrow p$ is false as $l_R(s) \not\subset h_R(p)$
  - $l_R(s)$ = { covert }
  - $h_R(p)$ = { public, analysis }

Analysis

- $(S_P, \leq_P)$ is a lattice, so it can be analyzed like a lattice policy
- Dual mapping preserves ordering, hence non-ordering and non-transitivity, of original policy
  - So results of analysis of $(S_P, \leq_P)$ can be mapped back into $(SC_R, \leq_R, join_R)$