Lecture 4
October 1, 2025

ECS 235A, Computer and Information Security

Administrative Stuff

- TA Office Hours
  - M 5:30pm–7:30pm, F 10:00am–12:00pm
  - In TA office, 3106 Kemper Hall
  - I have updated the **General Information** page accordingly
- Added what output for question 8 of the homework requires; here's an example (your results may differ!)

```
Opened file testfile for writing...
First write successful ...
Turned off write permission on testfile; permission code 400 ...
Second write failed ...
This system honors the Principle of Complete Mediation
```

Digital IDs

- The UK plans to issue free digital ID credentials for all UK citizens and legal residents
- From the UK government website: "digital ID will be mandatory as a means of proving your Right to Work."
  - Right to work means you are legally authorized to be employed in the UK
  - Web site: https://www.gov.uk/government/news/new-digital-id-scheme-to-be-rolled-out-across-uk
- What are the benefits of this?
- What are the drawbacks to this?

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
```

- Mono-conditional command
  - Single condition in this command

Multiple Conditions

- 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
```

Multiple Conditions: No `or`

- Let *p* give *q* *r* 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];
end
command grant•read•file•4(p, f, q)
    if c in A[p, q]
    then
        enter r into A[q, f];
end
```

Multiple Conditions: No `or`

- Let *p* give *q* *r* rights over *f*, if *p* owns *f* **or** *p* has *c* rights over *q*
- Now run:

```
grant•read•file•3(p, f, q)
grant•read•file•4(p, f, q)
```

- If either is true, then *r* is entered into *A*[*q*, *f*], as required

Copy Flag and Right

- Allows possessor to give rights to another
- Often attached to a right (called a *flag*), so only applies to that right
  - *r* is read right that cannot be copied
  - *rc* is read right that can be copied
- Is copy flag copied when giving *r* rights?
  - Depends on model, instantiation of model

Own Right

- Usually allows 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

Attenuation of Privilege

- Principle says you can't increase your rights, or give rights you do not possess
  - Restricts addition of rights within a system
  - Usually *ignored* for owner
    - Why? Owner gives themselvs rights, gives them to others, deletes their rights.

What Is "Secure"?

- Adding a generic right *r* where there was not one is "leaking"
  - In what follows, a right leaks if it was not present *initially*
  - Alternately: not present *in the previous state* (not discussed here)
- If a system *S*, beginning in initial state s_0, cannot leak right *r*, it is *safe with respect to the right r*
  - Otherwise it is called *unsafe with respect to the right r*

Safety Question

- Does there exist an algorithm for determining whether a protection system *S* with initial state s_0 is safe with respect to a generic right *r*?
  - Here, “safe” = “secure” for an abstract model

Mono-Operational Commands

- Answer: *yes*
- Sketch of proof:

  Consider minimal sequence of commands c_1, …, c_k to leak the right.

  - Can omit **delete**, **destroy**
  - Can merge all **create**s into one

  Worst case: insert every right into every entry; with *s* subjects and *o* objects initially, and *n* rights, upper bound is $k \leq n(s+1)(o+1)$

General Case

- Answer: *no*
- Sketch of proof:

  Reduce halting problem to safety problem
  - Map head motion of Turing machine into entering, deleting rights in the access control matrix
  - Turing machine symbols mapped into rights
  - Head position, end of tape indicated by special rights
  - Head motion represented by commands; two sets for R motion
    - One for mid-tape, one for end of tape
  - So protection system simulates a Turing machine *exactly*
  - TM halts when it enters state q_f; this means right has leaked

Confidentiality Models

- Overview
  - What is a confidentiality model
- Bell-LaPadula Model
  - General idea
  - Informal description of rules
- Tranquility
- Declassification

Confidentiality Policy

- Goal: prevent the unauthorized disclosure of information
  - Deals with information flow
  - Integrity incidental
- Multi-level security models are best-known examples
  - Bell-LaPadula Model basis for many, or most, of these

Bell-LaPadula Model, Step 1

- Security levels arranged in linear ordering
  - Top Secret: highest
  - Secret
  - Confidential
  - Unclassified: lowest
- Levels consist are called *security clearance* $L(s)$ for subjects and *security classification* $L(o)$ for objects

Example

security level	*subject*	*object*
Top Secret	Tamara	Personnel Files
Secret	Samuel	E-Mail Files
Confidential	Claire	Activity Logs
Unclassified	Ulaley	Telephone Lists

- Tamara can read all files
- Claire cannot read Personnel or E-Mail Files
- Ulaley can only read Telephone Lists

Reading Information

- Information flows *up*, not *down*
  - "Reads up" disallowed, "reads down" allowed
- Simple Security Condition (Step 1)
  - Subject *s* can read object *o* iff $L(o) \leq L(s)$ and *s* has permission to read *o*
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called "no reads up" rule

Writing Information

- Information flows up, not down
  - "Writes up" allowed, "writes down" disallowed
- *-Property (Step 1)
  - Subject *s* can write object *o* iff $L(s) \leq L(o)$ and *s* has permission to write *o*
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called "no writes down" rule

Basic Security Theorem, Step 1

- If a system is initially in a secure state, and every transition of the system satisfies the simple security condition, step 1, and the *-property, step 1, then every state of the system is secure
  - Proof: induct on the number of transitions

Lattices

- Lattices used to analyze several models
  - Bell-LaPadula confidentiality model
  - Biba integrity model
- A lattice consists of a set and a relation
- Relation must partially order set
  - Relation orders some, but not all, elements of set

Sets and Relations

- S set, R: $S \times S$ relation
  - If $a, b \in S$, and $(a, b) \in R$, write aRb
- Example
  - $I = \{ 1, 2, 3 \}$; R is $\leq$
  - $R = \{ (1, 1), (1, 2), (1, 3), (2, 2), (2, 3), (3, 3) \}$
  - So we write $1 \leq 2$ and $3 \leq 3$ but not $3 \leq 2$

Relation Properties

- Reflexive
  - For all $a \in S$, aRa
  - On I, $\leq$ is reflexive as $1 \leq 1$, $2 \leq 2$, $3 \leq 3$
- Antisymmetric
  - For all $a, b \in S$, $aRb \wedge bRa \Rightarrow a = b$
  - On I, $\leq$ is antisymmetric as $1 \leq x$ and $x \leq 1$ means $x = 1$
- Transitive
  - For all $a, b, c \in S$, $aRb \wedge bRc \Rightarrow aRc$
  - On I, $\leq$ is transitive as $1 \leq 2$ and $2 \leq 3$ means $1 \leq 3$

Example

- $\mathbb{C}$ set of complex numbers
- $a \in \mathbb{C} \Rightarrow a = a_R + a_I i$, where a_R, a_I integers
- $a \leq_C b$ if, and only if, $a_R \leq b_R$ and $a_I \leq b_I$
- $a \leq_C b$ is reflexive, antisymmetric, transitive
  - As ≤ is over integers, and a_R , a_I are integers

Partial Ordering

- Relation *R* orders some members of set *S*
  - If all ordered, it's a total ordering
- Example
  - ≤ on integers is total ordering
  - $\leq_{\mathbb{C}}$ is partial ordering on $\mathbb{C}$
    - Neither $3+5i \leq_{\mathbb{C}} 4+2i$ nor $4+2i \leq_{\mathbb{C}} 3+5i$ holds

Upper Bounds

- For $a, b \in S$, if u in S with aRu, bRu exists, then u is an *upper bound*
  - A *least upper bound* if there is no $t \in S$ such that aRt, bRt, and tRu
- Example
  - For $1 + 5i, 2 + 4i \in \mathbb{C}$
    - Some upper bounds are $2 + 5i$, $3 + 8i$, and $9 + 100i$
    - Least upper bound is $2 + 5i$

Lower Bounds

- For $a, b \in S$, if l in S with lRa, lRb exists, then l is a *lower bound*
  - A *greatest lower bound* if there is no $t \in S$ such that tRa, tRb, and lRt
- Example
  - For $1 + 5i, 2 + 4i \in \mathbb{C}$
    - Some lower bounds are 0, $-1 + 2i$, $1 + 1i$, and $1+4i$
    - Greatest lower bound is $1 + 4i$

Lattices

- Set S, relation R
  - R is reflexive, antisymmetric, transitive on elements of S
  - For every s, $t \in S$, there exists a greatest lower bound under R
  - For every s, $t \in S$, there exists a least upper bound under R

Example

- $S = \{ 0, 1, 2 \}$; $R = \leq$ is a lattice
  - R is clearly reflexive, antisymmetric, transitive on elements of S
  - Least upper bound of any two elements of S is the greater of the elements
  - Greatest lower bound of any two elements of S is the lesser of the elements

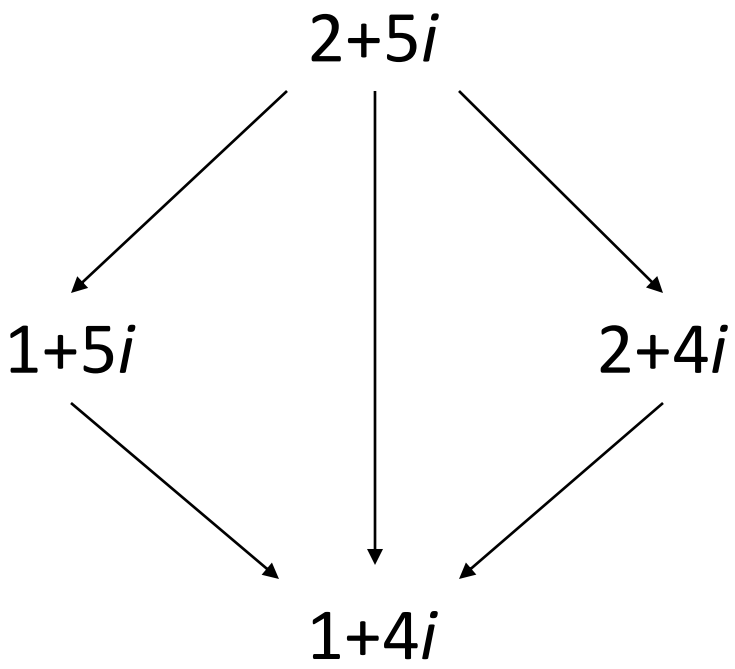
Picture

Arrows represent ≤; this forms a total ordering

Example

- $\mathbb{C}, \leq_{\mathbb{C}}$ form a lattice
  - $\leq_{\mathbb{C}}$ is reflexive, antisymmetric, and transitive
    - Shown earlier
  - Least upper bound for *a* and *b*:
    - $c_R = \max(a_R, b_R)$, $c_I = \max(a_I, b_I)$; then $c = c_R + c_I i$
  - Greatest lower bound for *a* and *b*:
    - $c_R = \min(a_R, b_R)$, $c_I = \min(a_I, b_I)$; then $c = c_R + c_I i$

Picture



Arrows represent $\leq_{\mathbb{C}}$