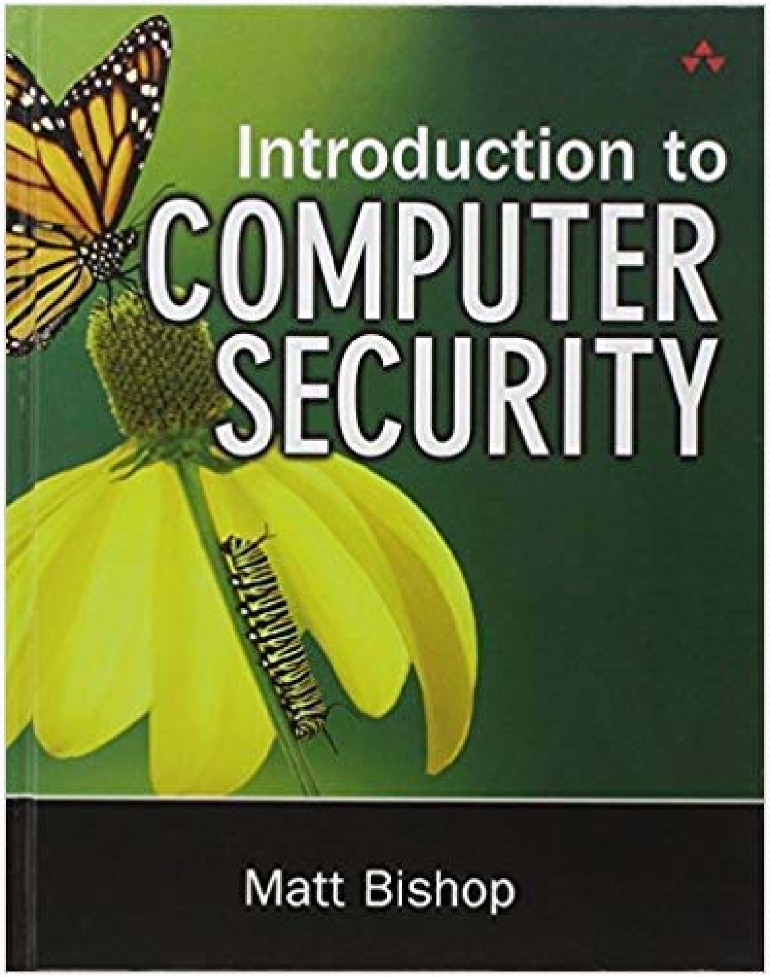
Chapter 1: An Overview of Computer Security

- Components of computer security
- Threats
- Policies and mechanisms
- The role of trust
- Assurance
- Operational Issues
- Human Issues

Textbook



Basic Components: the CIA

- Confidentiality
  - Definition: is the avoidance of the unauthorized disclosure of information
  - Example
  - How to achieve this: cryptography, access control

- Integrity
  - Trustworthiness of data or resources, or, improper or unauthorized modifications can be prevented or detected
  - Example
  - How to achieve this: prevention (via authentication and access control) and detection (analyze system events or the data itself)

- Availability
  - Definition: the property that information is accessible and modifiable in a timely fashion by those authorized to do so
  - Denial of Service attacks

Classes of Threats

- Threat: a potential violation of security
- Attacks: actions that actually cause a violation

- Disclosure: unauthorized access to information
- Deception: acceptance of false data
- Disruption: interruption or prevention of correct operation
- Usurpation: unauthorized control of some part of a system

Case Studies of Threats

- Snooping → passive wiretapping, packet sniffing
  - a form of disclosure
- Phishing: an impersonation of one entity (e.g., legitimate web site) by another (e.g., fake web site)
  - a form of deception
- Other examples on the Web

Policies and Mechanisms

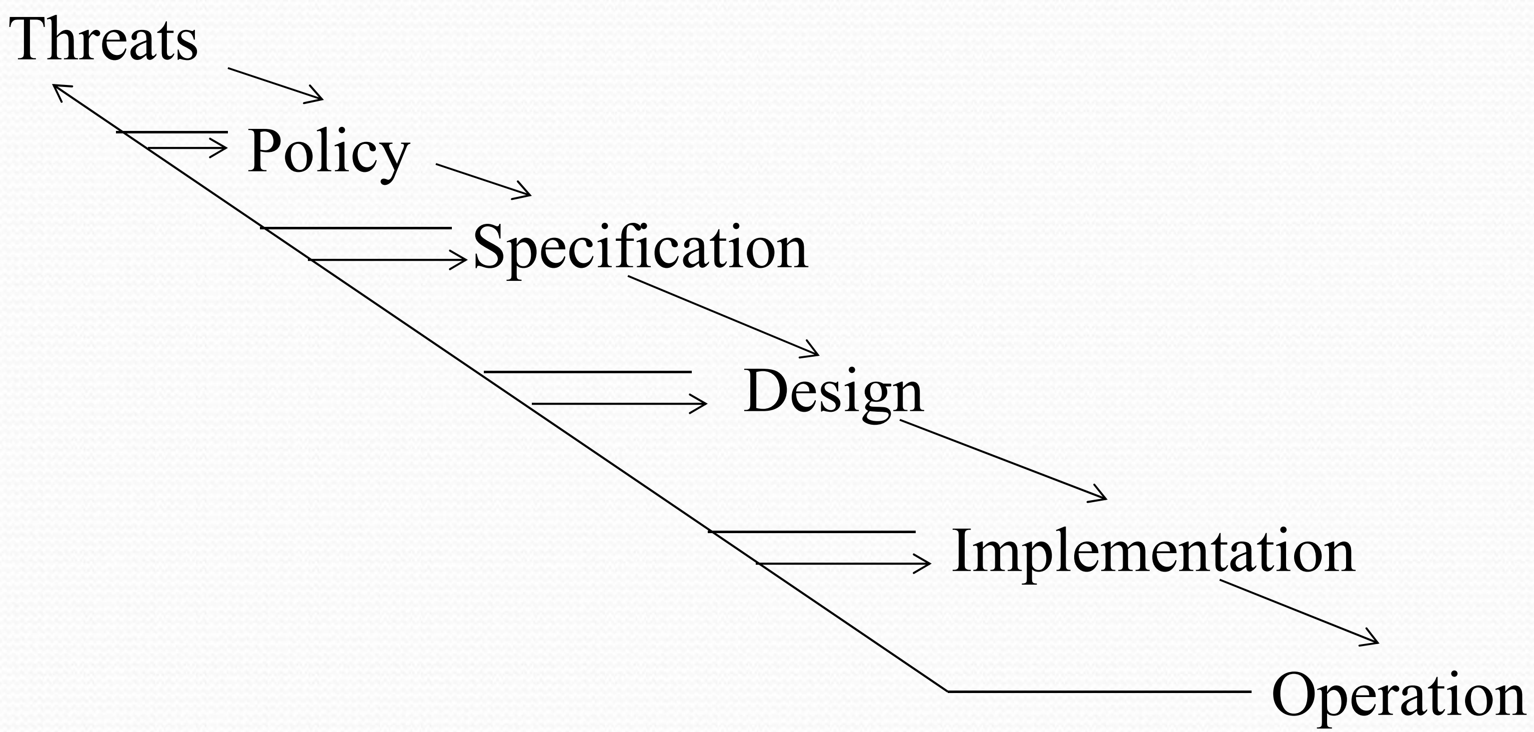
- Policy says what is, and is not, allowed
  - This defines "security" for the site/system/*etc*.
- Mechanism: a method, tool, or procedure for enforcing a security policy
- Example: access to your project source files on the lab computers

Goals of Security

- Prevention
  - Prevent attackers from violating security policy
- Detection
  - Detect attackers' violation of security policy
- Recovery
  - Stop attack, assess and repair damage
  - Continue to function correctly even if attack succeeds

Acknowledgement: Matt Bishop

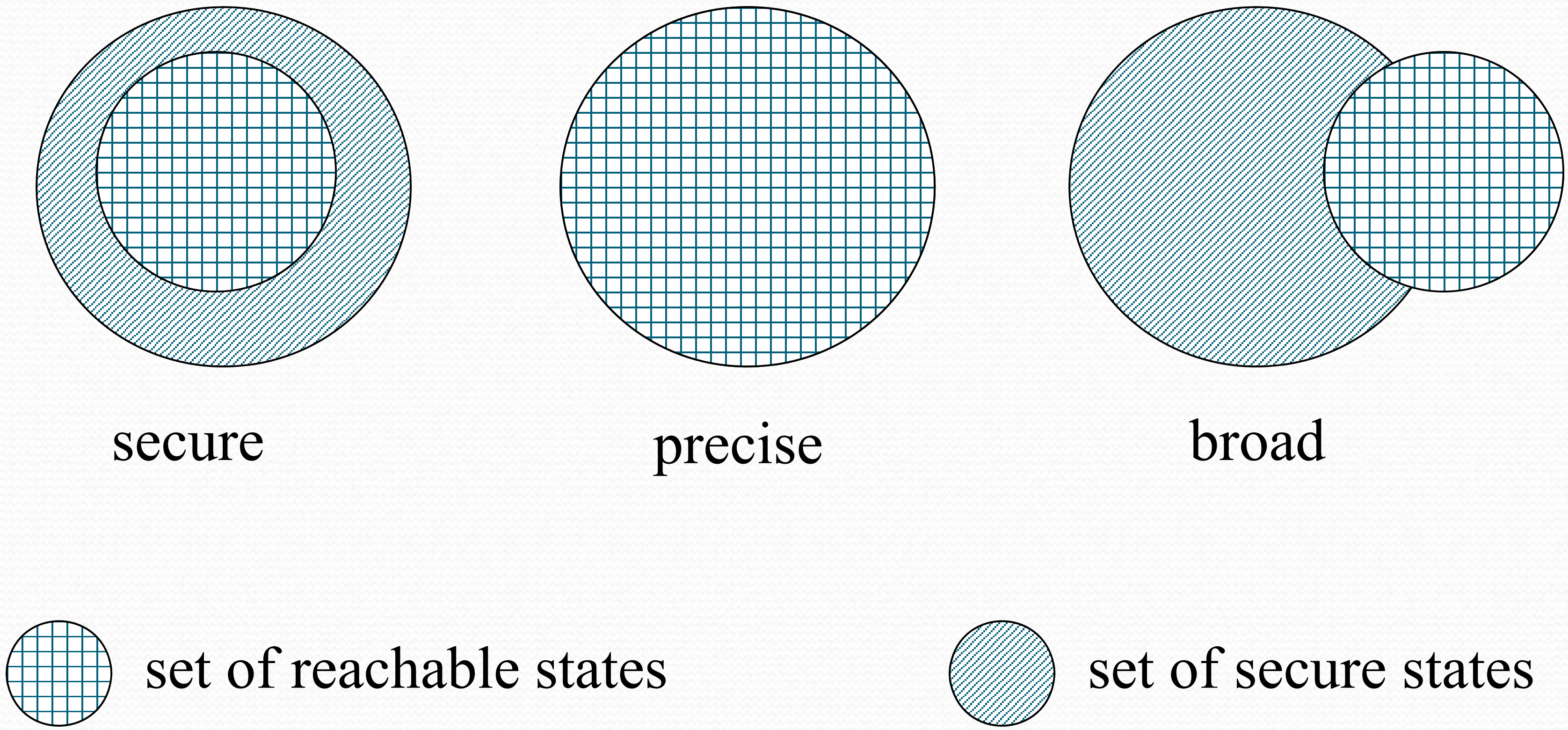
Road to a Secure System



Trust and Assumptions

- Underlie *all* aspects of security
- Policies
  - Unambiguously partition system states
  - Correctly capture security requirements
- Mechanisms
  - Assumed to enforce policy, i.e., must be appropriate
    - ✓ For example, using cryptography to ensure that a web site is available won't work
  - Support mechanisms (e.g., compilers, libraries, the hardware, and networks) work correctly

Types of Mechanisms



Assurance

- The degree to which the policies meet the requirements of the organizations using the system
- The degree to which the mechanisms correctly implement the policies
- More details in Chapter 17

Operational Issues

- Cost-Benefit Analysis
  - Is it cheaper to prevent or recover?

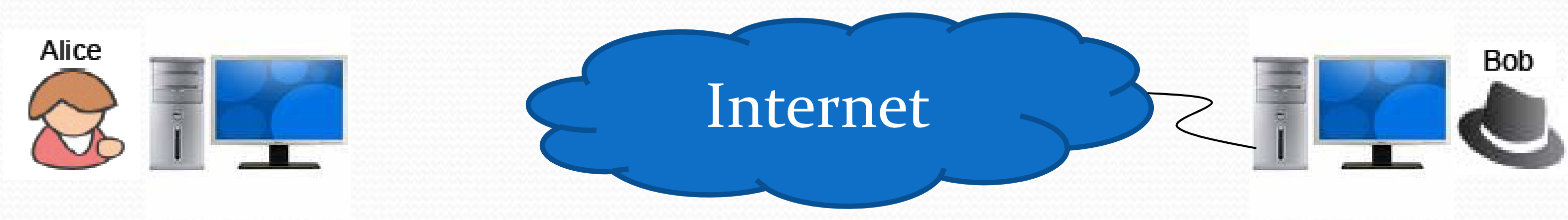
Operational Issues

- Cost-Benefit Analysis
  - Is it cheaper to prevent or recover?
- Risk Analysis:
  - Should we protect something?
  - How much should we protect this thing?

Case study: salary database

Operational Issues

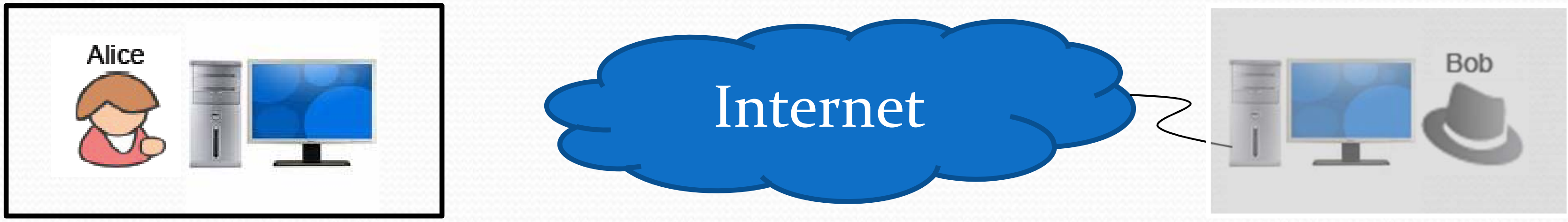
- Cost-Benefit Analysis
  - Is it cheaper to prevent or recover?
- Risk Analysis:
  - Should we protect something?
  - How much should we protect this thing?
  - A function of the environment and dynamically changing.



Acknowledgement: Matt Bishop

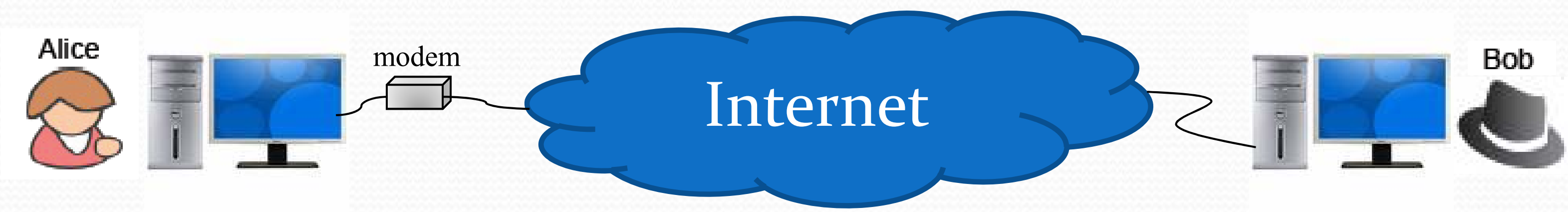
Operational Issues

- Cost-Benefit Analysis
  - Is it cheaper to prevent or recover?
- Risk Analysis:
  - Should we protect something?
  - How much should we protect this thing?
  - A function of the environment and dynamically changing.



Operational Issues

- Cost-Benefit Analysis
  - Is it cheaper to prevent or recover?
- Risk Analysis:
  - Should we protect something?
  - How much should we protect this thing?
  - A function of the environment and dynamically changing.



Operational Issues

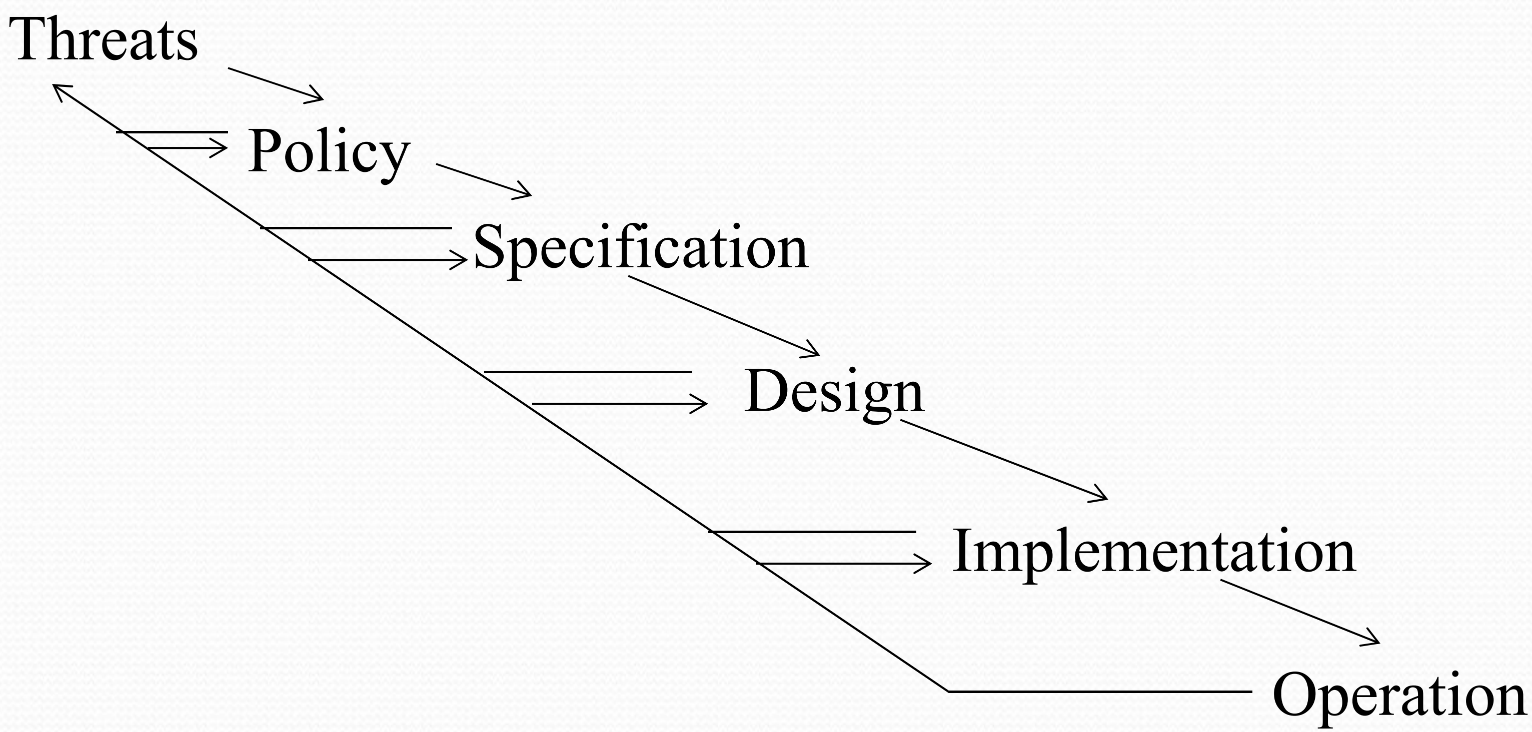
- Cost-Benefit Analysis
  - Is it cheaper to prevent or recover?
- Risk Analysis:
  - Should we protect something?
  - How much should we protect this thing?
  - A function of the environment and dynamically changing.
- Laws and Customs
  - Are desired security measures illegal? Crypto software
  - Will people do them? DNA samples for authentication, SSN as a password unacceptable

Human Issues

- Organizational Problems
  - Financial benefits

- People problems
  - Outsiders and insiders (e.g., disgruntled employees)
    - It is speculated that insiders account for 80-90% of all security problems
  - Social engineering

Tying Together



Key Points

- Policy defines security, and mechanisms enforce security
  - Confidentiality
  - Integrity
  - Availability
- Trust and knowing assumptions
- Importance of assurance
- The human factor

Acknowledgement: Matt Bishop

Chapter 2: Access Control Matrix

- Overview
- Access Control Matrix Model
- Protection State Transitions
  - Commands
  - Conditional Commands

Overview

- Protection state of system
  - Describes current settings, values of system relevant to protection
- Access control matrix
  - Describes protection state precisely
  - Matrix describing rights of subjects
  - State transitions change elements of matrix

Description

objects (entities)

subjects

	o_1	…	o_m	s_1	…	s_n
s_1						
s_2						
…						
s_n						

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

The triple (S,O,A) represents the protection state

Acknowledgement: Matt Bishop

Example

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

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

r – read
w – write
x – execute
a – append
o – own

State Transitions

- Change the protection state $X=(S,O,A)$ of system
- |– represents transition
  - $X_i \vdash_\tau X_{i+1}$: command τ moves system from state X_i to X_{i+1}
  - $X_i \vdash^* X_{i+1}$: a sequence of commands moves system from state X_i to X_{i+1}
- Commands often called *transformation procedures*

- $X_i = (S_i, O_i, A_i)$

State Transition (An Example)

A

	o_1	o_2
s_1	w	x
s_2		

X_1

Command

Give s_2 r right over o_2 →

A

	o_1	o_2
s_1	w	x
s_2		r

X_2

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*

- ACM: Access Control Matrix

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

- Definition: single primitive operation in a command
- Make process *p* the owner of file *g*

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

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
```

Key Points

- Access control matrix simplest abstraction mechanism for representing protection state
- Transitions alter protection state
- 6 primitive operations alter matrix
  - Transitions can be expressed as commands composed of these operations and, possibly, conditions

Chapter 3: Foundational Results

What Is “Secure”?

- Adding a generic right *r* where there was not one is “leaking”
- If a system *S*, beginning in initial state s_o, cannot leak right *r*, it is *safe with respect to the right r*.
  - Counter-example: Bob should not see Alice’s photo in Facebook, but if he somehow is able to due to an implementation flaw (directory traversal) in the Facebook web server.

s_0

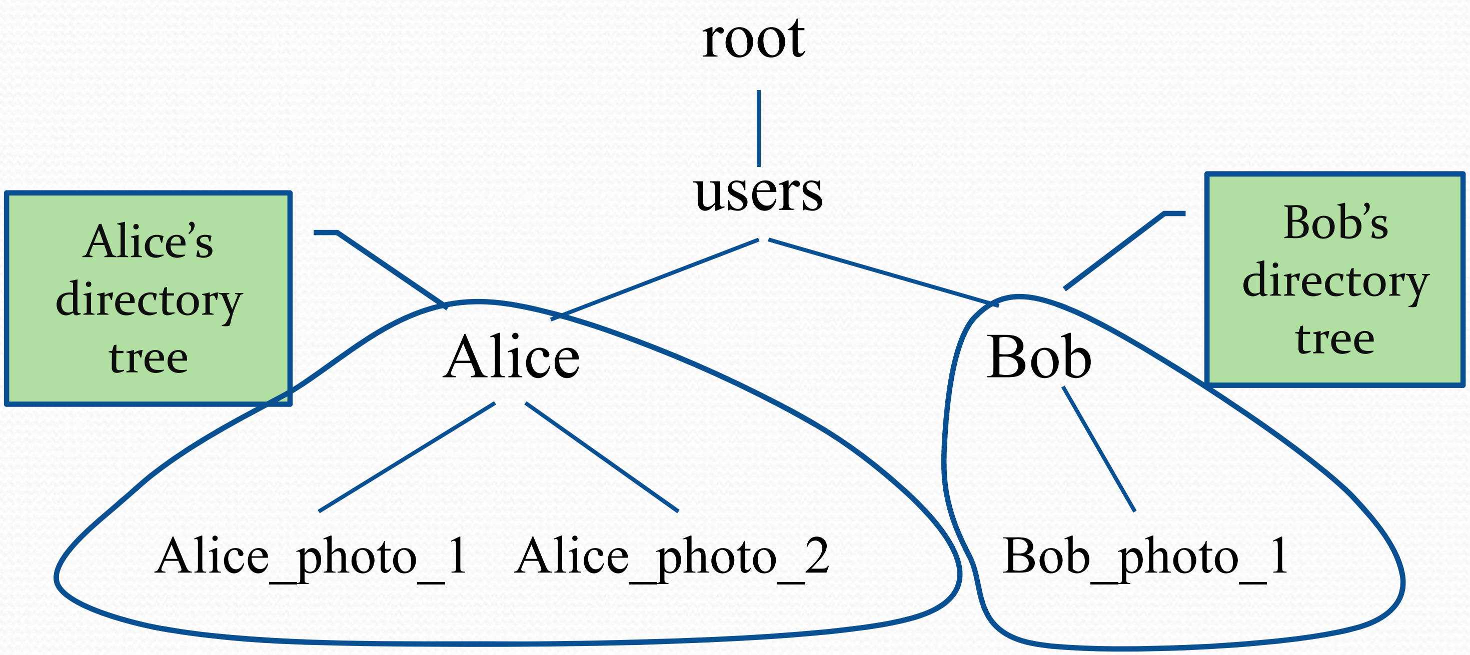
	o_1	…	o_m
p_1	…	…	…
…	…	…	…
p_n	…	…	…

$\Rightarrow \quad \dots \quad \Rightarrow$

	o_1	…	o_m
p_1	…	…	…
…	…	..r..	…
p_n	…	…	…

Acknowledgement: Matt Bishop

Example Insecure System



- Alice access url: http://facebook.com/Alice_photo_1
- Bob access url: http://facebook.com/../Alice/Alice_photo_1, violating security policy!

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
  - Assuming no authorized transfer of rights

Mono-Operational Commands

- Answer: *yes*
- Sketch of proof:

  Consider minimal sequence of commands $c_1, \ldots, c_k$ to leak the right.

  - Can omit **delete**, **destroy**
  - Can merge all **creates** into one

  Worst case: one new subject must be created; and the sequence of commands need to insert every right into every entry; with s subjects and o objects initially, and n rights, upper bound is $k \leq n(s+1)(o+1)+1$

- We can show that the length of this sequence is bounded. Therefore, we can enumerate all possible states and determine whether the system is safe.

General Case

- Answer: *no*
- Proof idea:

  Reduce halting problem to safety problem
- If safety question decidable, then represent Turing Machine using protection states and determine if q_f leaks
  - Implies halting problem decidable
- Conclusion: safety question undecidable

Key Points

- Safety problem undecidable
- Limiting scope of systems can make problem decidable