What is Information Security?

- Confidentiality
  - *Is this all?*
  - *Why not?*
- Availability
  - *To whom?*
- Authentication
  - *Still not there*
- Integrity

*It's about more than network security!*
Course Outline

1. Introduction: Role of security, Types of security, Definitions.
2. Access Control Matrix model
3. Protection Models
5. Formal policy models.
6. Information Flow
7. Authentication and Identity
8. TBD (probably basics of Cryptography)
13. Malicious Code: Viruses, Worms, etc.
15. Physical threats, operational security, Legal and Societal Issues

Midterm. Most likely date: 10/18.
Let me know of bad dates this week

Final Exam
December 18, 9pm – earliest you should count on leaving campus before you see the exam schedule

Course Administration
www.cs.purdue.edu/homes/clifton/cs526/

- Teaching Assistant: Ashish Kundu
- Course Announcements
  - Mailing list (directed to you@purdue.edu)
  - Discussion, grades, assignment submission through blackboard
- Evaluation/Grading
  - Midterm 25%, Final 36%
  - Exercises, projects 36%
    - 1-2 programming projects
    - 9-11 written assignments (similar to exercises in the book)
- Let me know if you will be taking the qual1
  - See web page for more
Course Text

• Recommended Text:
  – Matthew Bishop
    Computer Security: Art and Science
    Addison-Wesley, 2003
    ISBN 0-201-44099-7
  – If you don’t have the latest printing, see the above link for Errata pages

• Not required, but easier than finding/reading original papers

Waiting List / Registration

• Send me “background information” as follows:
  Career ID, Infosec Masters, Expected graduation, Research focus, Had CS555, Will take CS555, Taking CS626, likely TA next year

• Sample:
  clifton, no, 6/1991, Privacy and Data Mining, no, no, no, no

• Course is planned for spring as well
Introduction

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

Basic Components

• Confidentiality
  – Keeping data and resources hidden
• Integrity
  – Data integrity (integrity)
  – Origin integrity (authentication)
• Availability
  – Enabling access to data and resources
Classes of Threats

- Disclosure
  - Snooping
- Deception
  - Modification, spoofing, repudiation of origin, denial of receipt
- Disruption
  - Modification
- Usurpation
  - Modification, spoofing, delay, denial of service

Policies and Mechanisms

- Policy says what is, and is not, allowed
  - This defines “security” for the site/system/etc.
  - Policy definition: Informal? Formal?
- Mechanisms enforce policies
- Composition of policies
  - If policies conflict, discrepancies may create security vulnerabilities
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

Trust and Assumptions

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

- **secure**
- **precise**
- **broad**

- set of reachable states
- set of secure states
Assurance

- Specification
  - Requirements analysis
  - Statement of desired functionality
- Design
  - How system will meet specification
- Implementation
  - Programs/systems that carry out design

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?
- Laws and Customs
  - Are desired security measures illegal?
  - Will people do them?
Human Issues

• Organizational Problems
  – Power and responsibility
  – Financial benefits

• People problems
  – Outsiders and insiders
    • *Which do you think is the bigger problem?*
  – Social engineering

Tying the Definitions Together
Key Points

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

Student Choice Topics

- Trusted Computing Systems
  - How does software know underlying system can be trusted?
  - Case study of trusted system / verification
  - Validation process
- Forensics
  - Recovery/Prevention
  - Tracing/Prosecution
- Digital Rights Management
- Legal issues
- …
Models: Access Control

• What is access control?
  – Limiting who is allowed to do what

• What is an access control model?
  – Specifying who is allowed to do what

• What makes this hard?
  – Interactions between types of access
Basics

- **State**: Status of the system
  - Protection state: subset that deals with protection
- **Access Control Matrix**
  - Describes protection state
- **Formally**:
  - Objects $O$
  - Subjects $S$
  - Matrix $A \subseteq S \times O$
- **Tuple** $(S, O, A)$ defines protection states of system

Access Restriction Facility

- **Subject**: attributes (name, role, groups)
- **Verbs**: possible actions
  - Default rule for each verb
- **Objects associated with set of verbs**
  - Rule for each (object, verb) pair
  - Rule may be function of subject attributes
- **Can be converted to Access Control Matrix**
### Access Control Matrix: Boolean Evaluation Example

<table>
<thead>
<tr>
<th></th>
<th>Internal</th>
<th>Local</th>
<th>State University</th>
<th>Long Distance</th>
<th>International</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Public</strong></td>
<td>CR</td>
<td>T</td>
<td></td>
<td>R</td>
<td></td>
</tr>
<tr>
<td><strong>Student</strong></td>
<td>CR</td>
<td>T</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td><strong>Staff</strong></td>
<td>CR</td>
<td>T</td>
<td>CR</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td><strong>Account</strong></td>
<td>CR</td>
<td>T</td>
<td>CR</td>
<td>CR</td>
<td>CR</td>
</tr>
</tbody>
</table>

### 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 2

- Procedures \textit{inc\_ctr}, \textit{dec\_ctr}, \textit{manage}
- Variable \textit{counter}
- Rights +, −, \textit{call}

\begin{tabular}{|c|c|c|c|}
  \hline
  \textit{counter} & \textit{inc\_ctr} & \textit{dec\_ctr} & \textit{manage} \\
  \hline
  \textit{inc\_ctr} & + & & \\
  \textit{dec\_ctr} & − & & \\
  \textit{manage} & \textit{call} & \textit{call} & \textit{call} \\
  \hline
\end{tabular}

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), groups (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 < 5

Protection State Transitions

- State $X_i = (S_i, O_i, A_i)$
- Transitions $\tau_i$
  - Single transition $X_i \vdash_{\tau_i} X_{i+1}$
  - Series of transitions $X \vdash^* Y$
- Access control matrix may change
  - Change command $c$ associated with transition
    $X_i \vdash_{c, j+1 (p_{j+1},...,p_{j+1})} X_{j+1}$
- Commands often called *transformation procedures*
Special Privileges: Copy, Ownership

- Copy (or grant)
  - Possessor can extend privileges to another
- Own right
  - Possessor can change their own privileges
- Principle of Attenuation of Privilege
  - A subject may not give rights it does not possess

---

Primitive Commands

*(Harrison, Ruzzo, Ullman CACM’76)*

- Create Object \( o \)
  - Adds \( o \) to objects with no access
  - \( S'=S, O'=O \cup \{o\}, (\forall x \in S')[a'[x,o] = \emptyset], (\forall x \in S')(\forall y \in O)[a'[x,y] = a[x,y]] \)
- Create Subject \( s \)
  - Adds \( s \) to objects, subjects, sets relevant access control to \( \emptyset \)
- Enter \( r \) into \( a[s,o] \)
- Delete \( r \) from \( a[s,o] \)
- Destroy subject \( s \), destroy object \( o \)
Create Subject

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

Create Object

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

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

Delete Right

- Precondition: \( s \in S, o \in O \)
- Primitive command: delete \( r \) from \( a[s, o] \)
- Postconditions:
  - \( S' = S, O' = O \)
  - \( a'[s, o] = a[s, o] - \{ r \} \)
  - \( (\forall x,y \in S \times O - \{ s,o \}) \ [a'[x, y] = a[x, y]] \)
Destroy Subject

• Precondition: \( s \in S \)

• Primitive command: \textbf{destroy subject} \( s \)

• Postconditions:
  
  \(- S' = S - \{ s \}, O' = O - \{ s \} \)
  
  \(- (\forall y \in O')[a'[s, y] = \emptyset], (\forall x \in S')[a'[x, s] = \emptyset] \)
  
  \(- (\forall x \in S')(\forall y \in O') [a'[x, y] = a[x, y]] \)

Destroy Object

• Precondition: \( o \in o \)

• Primitive command: \textbf{destroy object} \( o \)

• Postconditions:
  
  \(- S' = S, O' = O - \{ o \} \)
  
  \(- (\forall x \in S')[a'[x, o] = \emptyset] \)
  
  \(- (\forall x \in S')(\forall y \in O') [a'[x, y] = a[x, y]] \)
Creating File

- Process \( p \) creates file \( f \) with \( r \) and \( w \) permission

\[
\text{command } create\cdot file(p, f) \\
\text{create object } f; \\
\text{enter } own \text{ into } A[p, f]; \\
\text{enter } r \text{ into } A[p, f]; \\
\text{enter } w \text{ into } A[p, f]; \\
\text{end}
\]

Mono-Operational Commands

- Make process \( p \) the owner of file \( g \)

\[
\text{command } make\cdot owner(p, g) \\
\text{enter } own \text{ into } A[p, g]; \\
\text{end}
\]

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

• Let \( p \) give \( q \) \( r \) rights over \( f \), if \( p \) owns \( f \)

\[
\text{command} \; \text{grant} \cdot \text{read} \cdot \text{file} \cdot 1(p, f, q) \\
\text{if own in } A[p, f] \\
\text{then} \\
\text{enter } r \text{ into } A[q, f]; \\
\text{end}
\]

• 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 \( q \)

\[
\text{command} \; \text{grant} \cdot \text{readwrite} \cdot \text{file} \cdot 2(p, f, q) \\
\text{if own in } A[p, f] \text{ and } c \text{ in } A[p, q] \\
\text{then} \\
\text{enter } r \text{ into } A[q, f]; \\
\text{enter } w \text{ into } A[q, f]; \\
\text{end}
\]
Copy Right

- Allows possessor to give rights to another
- Often attached to a right, 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 give rights you do not possess
  – Restricts addition of rights within a system
  – Usually *ignored* for owner
    • Why? Owner gives herself rights, gives them to others, deletes her rights.

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
What is **Secure**?

- A secure system doesn't allow violations of policy
  - Is this a good definition?
  - Can we use it?
- Alternative view: based on rights
  - Start with access control matrix $A$
  - **Leak**: commands can add right $r$ to an element of $A$ not containing $r$
  - **Safe**: System is safe with respect to $r$ if $r$ cannot be leaked

Formally:

- Given
  - initial state $X_0 = (S_0, O_0, A_0)$
  - Set of primitive commands $c$
- Can we reach a state $X_n$ where $\exists s, o$ such that $A_n[s, o]$ includes a right $r$ not in $A_0[s, o]$?
  - If so, the system is not safe
  - But is “safe” secure?

*Are commands correctly implemented?*
Example: Unix File System

- **Access Control Matrix**
  - Root has access to all files
  - Owner has access to their own files
- **Safe with respect to file access right?**
  - No chmod/chown command
  - Only “root” can get root privileges
  - Only user can authenticate as themselves

*Is “Safe” definition useful?*

Solution: Trust

- Safety doesn’t distinguish leak from authorized transfer of rights
- Subjects authorized to receive transfer of rights deemed “trusted”
  - Eliminate trusted subjects from matrix
Decidability Result
(Harrison, Ruzzo, Ullman CACM’76)

• Given a system where each command consists of a single primitive command, there exists an algorithm that will determine if a protection system with initial state \( X_0 \) is safe with respect to right \( r \).

• Proof: determine minimum commands \( k \) to leak
  – Delete/destroy: Can’t leak (or be detected)
  – Create/enter: new subjects/objects “equal”, so treat all new subjects as one
  – If \( n \) rights, leak possible, must be able to leak in \( n(|S_0|+1)(|O_0|+1)+1 \) commands

• Enumerate all possible to decide

Decidability: Non-Primitive Commands

• It is undecidable if a given state of a given protection system is safe for a given generic right

• Proof: Reduction from halting problem
  – Symbols, states \( \Rightarrow \) rights
  – Tape cell \( \Rightarrow \) subject (can create new subjects)
  – Right own: \( s_i \) owns \( s_i+1 \) for \( 1 \leq i < k \)
  – Cell \( s_i A \Rightarrow s_i \) has \( A \) rights on itself
  – Cell \( s_k \Rightarrow s_k \) has end rights on itself
  – State \( p \), head at \( s_i \Rightarrow s_i \) has \( p \) rights on itself
Example:

Turing Machine

| A | B | C | D | … |

Matrix

<table>
<thead>
<tr>
<th></th>
<th>s₁</th>
<th>s₂</th>
<th>s₃</th>
<th>s₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>s₁</td>
<td>A</td>
<td>own</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s₂</td>
<td>B</td>
<td>own</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s₃</td>
<td>C, p</td>
<td>own</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s₄</td>
<td>D, end</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mapping

| 1 | 2 | 3 | 4 |

| A | B | X | D | … |

head

After $\delta(k, C) = (k₁, X, R)$ where $k$ is the current state and $k₁$ the next state

<table>
<thead>
<tr>
<th></th>
<th>s₁</th>
<th>s₂</th>
<th>s₃</th>
<th>s₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>s₁</td>
<td>A</td>
<td>own</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s₂</td>
<td>B</td>
<td>own</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s₃</td>
<td>X</td>
<td>own</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s₄</td>
<td>D k₁</td>
<td>end</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Command Mapping

\[ \delta(k, C) = (k_1, X, R) \] at intermediate becomes

command \( c_{k,C}(s_3, s_4) \)

if own in \( A[s_3, s_4] \) and \( k \) in \( A[s_3, s_3] \)
and \( C \) in \( A[s_3, s_3] \)

then

\begin{align*}
&\text{delete } k \text{ from } A[s_3, s_3]; \\
&\text{delete } C \text{ from } A[s_3, s_3]; \\
&\text{enter } X \text{ into } A[s_3, s_3]; \\
&\text{enter } k_1 \text{ into } A[s_4, s_4]; \\
&\text{end}
\end{align*}

Commands:

- Halting problem Turing Machine: Symbols A, B; states \( p, q \)
- \( C_{p,A}(s_p, s_{i-1}) \) (move left)
  - if own in \( a[s_{i-1}, s_i] \) and \( p \) in \( a[s_i, s_i] \) and \( A \) in \( a[s_p, s_i] \)
    - Delete \( p \) from \( a[s_i, s_i] \), A from \( a[s_i, s_i] \)
    - Enter B into \( a[s_i, s_i] \), q into \( a[s_{i-1}, s_{i-1}] \)
- Similar commands for move right, move right at end of tape
- Simulates Turing machine
  - Leaks halting state \( \Rightarrow \) halting state in the matrix \( \Rightarrow \) Halting state reached

This is undecidable!
After $\delta(k_1, D) = (k_2, Y, R)$ where $k_1$ is the current state and $k_2$ the next state

\[\begin{array}{c|ccccc}
1 & 2 & 3 & 4 & 5 & \text{head} \\
A & B & X & Y & b & \\
\end{array}\]

\[
\begin{array}{c|ccccc}
& s_1 & s_2 & s_3 & s_4 & s_5 \\
\hline
s_1 & A & \text{own} \\
s_2 & B & \text{own} \\
s_3 & X & \text{own} \\
s_4 & Y & \text{own} \\
s_5 & b & k_2 & \text{end} \\
\end{array}
\]

\[\text{Command Mapping:}\]

\[\delta(k_1, D) = (k_2, Y, R) \text{ at end becomes}\]

\[
\text{command crightmost}_{k_1, C}(s_4, s_5)
\]

\[
\text{if } \text{end in } A[s_4, s_4] \text{ and } k_1 \text{ in } A[s_4, s_4]
\]

\[
\text{and } D \text{ in } A[s_4, s_4]
\]

\[
\text{then}
\]

\[
\text{delete end from } A[s_4, s_4];
\]

\[
\text{create subject } s_5;
\]

\[
\text{enter own into } A[s_4, s_5];
\]

\[
\text{enter end into } A[s_5, s_5];
\]

\[
\text{delete } k_1 \text{ from } A[s_4, s_4];
\]

\[
\text{delete } D \text{ from } A[s_4, s_4];
\]

\[
\text{enter } Y \text{ into } A[s_4, s_4];
\]

\[
\text{enter } k_2 \text{ into } A[s_5, s_5];
\]

\[
\text{end}
\]
Rest of Proof

- Protection system exactly simulates a TM
  - Exactly 1 *end* right in ACM
  - 1 right in entries corresponds to state
  - Thus, at most 1 applicable command
- If TM enters state $q_f$, then right has leaked
- If safety question decidable, then represent TM as above and determine if $q_f$ leaks
  - Implies halting problem decidable
- Conclusion: safety question undecidable

Other Results

*(most from the same authors)*

- Set of unsafe systems recursively enumerable
- Without create primitive, safety in P-SPACE
  - Like halting problem reduction, but no unlimited tape
- Without delete/destroy, still undecidable
  - Decidable if at most one condition allowed per command
  - Still holds if delete allowed
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: insert every right into every entry; with $s$ subjects and $o$ objects initially, and $n$ rights, upper bound is $k \leq n(s+1)(o+1)$

What Else Might We Add?

- Default Rule
  – General default: Receive
  – Object default: Call Internal
  – Requires ability to override with negative and positive access
- Time-based access
  – Allow students to call on State University system after hours?
- History-based access
Access Control by History

- Example: Statistical Database
  - Allows queries for general statistics
  - But not individual values
- Valid queries: Statistics on 20+ individuals
  - Total salary of all Deans
  - Salary of Computer Science Professors
- See a problem coming?
  - Salary of CS Professors who aren’t Deans

Solution: Query Set Overlap Control
(Dobkin, Jones & Lipton ’79)

- Query valid if intersection of query coverage and each previous query < r
- Given K minimum query size, r overlap:
  - Need 1 + (K-1)/r queries to compromise
- Can represent as access control matrix
  - Subjects: entities issuing queries
  - Objects: Powerset of records
  - \( O_s(i) \) : objects referenced by s in queries 1..i
  - \( A[s,o] = \text{read} \iff \bigwedge_{q \in 1..i} \bigcap q \circ O < r \)
Next

• Optional reading: Dobkin, Jones, and Lipton (TODS 4(1), see course web site)
• Basic theorems on protection states
  – Decidability of safety of a state with respect to a right
• More Protection Models

Protection Study: Your Homework

• What does it take to make sure your homework is secure?
  – Let’s assume a Unix system (mentor.ics)
  – Issues?
• Participation Expected!
Where does this leave us?

• Safety decidable for some models
  – Are they practical?

• Safety only works if maximum rights known in advance
  – Policy must specify all rights someone could get, not just what they have
  – Where might this make sense?

• Next: Example of a decidable model
  – Take-Grant Protection Model