What?

- It’s about computer security
- Textbook:
  - *Computer Security: Art and Science* by Matt Bishop
  - Does everybody have the book?
- Web page:
  - [http://www.cs.uoregon.edu/classes/cis410sec/](http://www.cs.uoregon.edu/classes/cis410sec/)
  - [http://www.cs.uoregon.edu/classes/cis510sec/](http://www.cs.uoregon.edu/classes/cis510sec/)

Who?

- Instructor:
  - Jun Li: lijun@cs.uoregon.edu
  - Office: Deschutes 334
- GTF:
  - Jin Zhang: jinzhang@cs.uoregon.edu
  - Office: Deschutes 230

When?

- Class hours:
  - Tuesday, Thursday 10 - 11:20 a.m.
  - 102 Deady Hall
- Office hours:
  - Tuesdays 11:30 - 12:30 a.m.
  - 334 Deschutes
- TA hours:
  - MW 2:30 - 3:30 p.m.
  - 230 Deschutes

Prerequisites

- CIS 314
- CIS 415/515
  - Recommended for undergrads
  - Required for grads
- CIS 432/532
  - Recommended for undergrads
  - Required for grads

410 Grading Policy

- Undergrads:
  - Class participation: 10%
  - In-class quiz: 10%
  - Homework: 20%
  - Midterm: 25%
  - Final: 35%
510 Grading Policy

• Graduate students
  – Class participation: 10%
  – In-class quiz: 10%
  – Midterm: 25%
  – Project: 55%

Quiz and Homework

• Quiz:
  – To check your learning quality
  – No announcement in advance
  – Often in 5 minutes
• Homework (undergrads)
  – Announced on a Thursday
  – Due the next Tuesday
    • No late submission will be accepted
    – Handed out within one week

Project (grads only)

• 1-3 people per team
• Identify a security problem to solve by yourself
  – But talk to me before decided
  – Typically you should get your hands dirty
• Be ambitious:
  – A good work can be publishable
• Be down-to-earth
  – Make sure you make solid progress day by day

Schedule

• Midterm:
  – May 13th, Tuesday
  – Coverage: all materials covered up to May 8th
• Final (undergrads only):
  – June 10th, Tuesday, 8-10:00 a.m.
• Project (grads only)
  – A 5-10 minutes presentation (4/17)
  – A 2-page progress report (5/6)
  – A final report (6/6)

On using emails

• Subject line must be in the format of
  – CIS 410: <issue>, or
  – CIS 510: <issue>
• Otherwise may be ignored
  – Sorry I have many emails to handle every day

An Overview of Computer Security
Basic Components

• Confidentiality
• Integrity
• Availability

Confidentiality

• The concealment of information or resources
  – “need to know” principle
• Mechanisms to support confidentiality
  – Cryptography
  – System-dependent mechanisms
    • E.g. File permission handling

Integrity

• Trustworthiness of data or resource
• Includes:
  – Data integrity
  – Origin integrity
    • Source authentication to ensure credibility
• Mechanisms to support integrity:
  – Prevention
    • Block outsiders and insiders
  – Detection
    • Analyze system events and data

Availability

• The ability to use the info or resource as desired
• Mechanisms to support availability:
  – ?

Threat and Attack

• Threat: a potential violation of security
  – Need not actually occur
    • E.g. data loss
• Attack: an action that can cause a threat to become real
  – E.g. wire eavesdropping
• Interchangeable in the following text

Four Classes of Threat

<table>
<thead>
<tr>
<th></th>
<th>Sampling</th>
<th>Alteration</th>
<th>Repudiation</th>
<th>Denial of origin</th>
<th>Denial of receipt</th>
<th>Delay</th>
<th>Denial of service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disclosure</td>
<td>v</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deception</td>
<td>v</td>
<td>v</td>
<td></td>
<td></td>
<td></td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>Disruption</td>
<td>v</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequestration</td>
<td>v</td>
<td>v</td>
<td></td>
<td></td>
<td>v</td>
<td></td>
<td>v</td>
</tr>
</tbody>
</table>
### Security Policy and Mechanism

- **Policy**: a statement of what is allowed and what is not
  - In order to provide confidentiality, integrity, and availability in the context of a specific system
- **Mechanism**: a method, tool, or procedure for enforcing a security policy
  - So that those threats cannot endanger a system’s security

### Goals of Security

- **Prevention**
- **Detection**
- **Recovery**

- Based on the security policy defined
- Using the security mechanism

### Assumptions

- How to determine whether or not a security policy is **correctly** defined?
  - Depending on the specific assumptions made to a specific system in a specific environment
- Two assumptions often made of a security policy
  - The policy leads to two types of states: **secure** and **nonsecure**
  - The security mechanisms prevent the system from entering into a nonsecure state

### Mechanisms: secure, precise, or broad

- **Denote**
  - $P$: the set of all possible states
  - $Q$: the set of secure states as specified by security policy
  - $R$: the set of states that a system can enter with the security mechanisms provided
- **A security mechanism** is
  - **Secure**: if $R \cap Q$
  - **Precise**: if $R=Q$
  - **Broad**: if $[] \models r$ [if $R$ but $\not\models Q$]

### Assurance

- How much can one believe that a system is secure?
- Look at three aspects:
  - Specification
    - Formally or informally states the desired functions, and security policies
  - Design
    - Creates a system that satisfies the design
  - Assurance: what one must trust in order to believe the security of a system

### Operational Issues

- Useful security policy and mechanism must consider cost-benefit tradeoff
- **Risk analysis** is often the key
  - In order to know whether and to what level to protect a system
  - A function of environment, time, service provided, etc.
- **Laws and Customs**
Human Issues

- Organizational problems
  - Who is responsible, where are necessary resources, what to protect, etc.
- People problems
  - Heart of any security problem is people
  - Human intervention can often bypass security mechanisms
  - Outsiders vs. insiders
  - Social engineering
  - Configuration errors
  - etc.

Security Life Cycle

- Threats
- Policy
- Specification
- Design
- Implementation
- Operation and Maintenance

Introduction

- Primary abstraction mechanism in computer security
- Can express any expressible security policy
- Simple and precise
- Ideal for theoretical analysis of security problems

Access Control Matrix

- Specifies the rights every subject \( s \) has over every object \( o \) in a system
  - Every matrix element is \( a[s, o] \).
- Described what’s allowed and what’s not
  - Current state \( x \) can be compared against the matrix to determine whether or not \( x \in Q \).
- A system’s state can be represented as \((S, O, A)\)

Secure Protection State

- A system is always in some state
  - What we are concerned here is protection state
- Secure or nonsecure
- \( P \): the set of possible protection states
- \( Q \subseteq P \): those secure protection states
- A security policy is used to characterize \( Q \)
- A secure security mechanism is used to enforce a system to be in a state \( q \in Q \)
An example

<table>
<thead>
<tr>
<th>file 1</th>
<th>file 2</th>
<th>process 1</th>
<th>process 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>r, w, own</td>
<td>r, w</td>
<td>r, w, exec, own</td>
<td>w</td>
</tr>
</tbody>
</table>

process 2: append r, own r r, w, exec, own

Protection State Transition

- The state of a system changes as operations are executed
- Initial State: X_0 = (S_0, O_0, A_0)
- With a set of operations []_1, []_2, . . ., successive states are X_1, X_2, . . ., where X_i[]_i ≡ X_{i+1}
- Note that A_i also changes to A_{i+1}

Command

- To represent a sequence of state transitions after a series of operations, we use X[] *Y
- This series of operations can be called command
- Transition by command can be written as X_i[] c_{i+1}(p_{i+1,1}, . . . , p_{i+1,m}) X_{i+1}

Primitive Command

- (S, O, A) |[] (S', O', A') that alters access control matrix
- A set of primitive commands defined by Harrison etc.
  - Create subject s
  - Create object o
  - Enter r into a[s, o]
  - Delete r from a[s, o]
  - Destroy subject s
  - Destroy object o

Create Subject s

- Precondition: s [] O
- Postcondition:
  - S' = S[ s ], O' = O[]{ s }
  - [] s[] O'[ a' [x,y] = O[]{ s }]
  - [] x[] S' [ a' [x,y] = O[]{ s }]
  - [] x[] S ([]) y[] O'[ a' [x,y] = O[]{ s }]

Create Object o

- Precondition: o [] O
- Postcondition:
  - S' = S, O' = O[]{ o }
  - [] s[] S'[ a' [x,y] = O[]{ o }]
  - [] x[] S ([]) y[] O[ a' [x,y] = O[]{ o }].
Enter $r$ into $a[s,o]$

- Precondition: $s \notin S$, $o \notin O$
- Postconditions:
  - $S' = S \cup \{s\}$, $O' = O \cup \{r\}$
  - $\forall x \ni S \cup \{s\} \cup O \ni \{x,y\} \ni a'[x,y] = a[x,y]$

Delete $r$ from $a[s,o]$

- Precondition: $s \notin S$, $o \notin O$
- Postconditions:
  - $S' = S \setminus \{s\}$, $O' = O \setminus \{r\}$
  - $\forall x \ni S \setminus \{s\} \cup O \ni \{x,y\} \ni a'[x,y] = a[x,y]$

Destroy Subject $s$

- Precondition: $s \notin S$
- Postcondition:
  - $S' = S \setminus \{s\}$, $O' = O$
  - $\forall x \ni S \setminus \{s\} \cup O \ni \{x,y\} \ni a'[x,y] = a[x,y]$

Destroy Object $o$

- Precondition: $o \notin O$
- Postcondition:
  - $S' = S$, $O' = O \setminus \{o\}$
  - $\forall x \ni S \cup O \setminus \{o\} \ni \{x,y\} \ni a'[x,y] = a[x,y]$

Complex Command

- A sequence of multiple primitive commands
- Examples:
  - Command `create_file (p,f)`
    - `create object f`
    - `enter own into a[p,f]`
  - Command `make_owner (a,f)`
    - `enter own into a[a,f]`

Conditional Commands

- Some specific conditions must be met before a state can be changed
  - Command `grant_read_file (p,f,q)`
    - `if own in a[p,f]`
    - `then`
    - `enter r into a[q,f]`
    - `end`
  - Command `grant_read_file_2 (p,f,q)`
    - `if r in a[p,f] and c in a[p,f]`
    - `then`
    - `enter r into a[q,f]`
    - `end`
Although loneliness has always been a friend of mine
I'm leaving my life in your hands
People say I'm crazy and that I am blind
Risking it all in a glance
And how you got me blind is still a mystery
I can't get you out of my head
Don't care what is written in your history
As long as you're here with me

Chorus:
I don't care who you are
Where you're from
What you did
As long as you love me
Who you are
Where you're from
Don't care what you did
As long as you love me

Every little thing that you have said and done
Feels like it's deep within me
Doesn't really matter if you're on the run
It seems like we're meant to be

Bridge:
I've tried to hide it so that no one knows
But I guess it shows
When you look into my eyes
What you did and where you are coming from
I don't care, as long as you love me, baby
Who you are
Where you're from
Don't care what you did
As long as you love me
(repeat to fade)