CS 4410
Operating Systems

Security

Summer 2011
Cornell University
Today

• How does the OS provide security?
• Secure System
• Security Violations
• Security Measures
• Threats
• User Authentication
• Protection
Secure System

• A system is secure if its resources are used and accessed as intended under all circumstances.

• But, total security cannot be achieved.

• We must have mechanisms to make security breaches a rare occurrence, rather than the norm.
Security Violations

• Breach of **confidentiality**
  • Unauthorized reading of data.

• Breach of **integrity**
  • Unauthorized modification of data.

• Breach of **availability**
  • Unauthorized destruction of data.

• **Theft of service**
  • Unauthorized use of resources.

• **Denial of service**
  • Prevent legitimate use of the system.
Security Measures

- The **protection** of the system is split into four levels:
  - Physical
  - Human
  - Operating System
  - Network
- This chain is as weak as its weakest link.
- As intruders exploit security vulnerabilities, security countermeasures are created.
  - Cat-and-mouse game
Program Threats

• Trojan Horse
  • A code segment that belongs to an 'innocent' program.
  • The user gets the program (cracked game or application) and installs it.
  • While the program is being executed, the malicious code:
    - obtains user's privileges and
    - modifies/deletes user’s file, sends important info to cracker, etc
Program Threats

- **Stack and Buffer Overflow**
  - It exploits a bug in a program:
    - The programmer neglects to code bounds checking on an input field.

```c
int A(argc, argv) {
    char buffer[BUFFER_SIZE];
    if (argc < 2)
        return -1;
    else{
        strcpy(buffer, argv[1]);
        return 0;
    }
}
```
Program Threats

• Viruses
  • A fragment of code embedded in a legitimate program or file.
  • It corrupts/modifies files.
  • They are incorporated in emails (spams) and infect the contacts of the user.
System and Network Threats

● **Worms:**
  - Self-replicated malware program.
  - It is spread through interconnected computers.
  - It does not need to attach itself to an existing program.
  - Harmful for the network.

● **Denial of Service**
User Authentication

- The ability of the system to **identify** each user.
- Major security problem for the OS.
- It is based on one or more of three things:
  - The user's **possession** of something (a key/card).
  - The user's **knowledge** of something (identifier/card).
  - An **attribute** of the user (fingerprint, signature).
- Examples:
  - Passwords
  - One-time Passwords
  - Biometrics
Protection

- **Multiprogramming and timesharing** OSes should define access rights (read, write, execute) that each process should have.

- Programs, users and even systems should be given just enough privileges to perform their tasks.
  - **Principle of least privilege**

- The OS should define which **access rights** each **process** has for each **object** (hardware, software).
Protection

- The OS defines **Protection Domains**.
- Each domain defines a set of objects and the types of operations that may be invoked on each object.
  - A domain is a **collection of access rights**.
  - `<object-name, rights-set>`
  - Ex. `<fine_route.c, {read, execute}>`
- Each process operates within a protection domain.
Protection

- **Access Matrix**

<table>
<thead>
<tr>
<th>domain \ object</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>printer</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>read</td>
<td></td>
<td>read</td>
<td></td>
</tr>
<tr>
<td>D2</td>
<td></td>
<td></td>
<td></td>
<td>print</td>
</tr>
<tr>
<td>D3</td>
<td></td>
<td>read</td>
<td>execute</td>
<td></td>
</tr>
<tr>
<td>D4</td>
<td>read write</td>
<td></td>
<td>read write</td>
<td></td>
</tr>
</tbody>
</table>

- **Implementation**
  - Global Table
  - Access Lists for Objects
  - Capability Lists for Domains
Today

• How does the OS provide security?
• Secure System
• Security Violations
• Security Measures
• Threats
• User Authentication
• Protection