Motivation

Security problems:

- Often viewed as software errors:
  - Buffer overruns
  - Race conditions
  - Weak crypto systems

- Extended view in this paper:
  - Correct use of software is equally important
  - User interfaces and usability are critical for security
Motivation (2)

“A computer is secure if you can depend on it and its software to behave as you expect.”

(Definition by Garfinkel and Spafford)
# 10 Design Principles

<table>
<thead>
<tr>
<th></th>
<th>Principle</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Principle of Least Resistance</td>
<td>Fundamental</td>
</tr>
<tr>
<td>2</td>
<td>Principle of Appropriate Boundaries</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Principle of Explicit Authorization</td>
<td>Actor-Ability State</td>
</tr>
<tr>
<td>4</td>
<td>Principle of Visibility</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Principle of Revocability</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Principle of Expected Ability</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Principle of Trusted Path</td>
<td>Input/Output</td>
</tr>
<tr>
<td>8</td>
<td>Principle of Identifiability</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Principle of Expressiveness</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Principle of Clarity</td>
<td></td>
</tr>
</tbody>
</table>
User and User Agent

**User:**
- Person sitting in front of the computer

**User Agent:**
- Local Computer: Shell
- Internet: Web Browser

➡️ Nesting possible
**Path of Least Resistance**

**Principle of Least Resistance:**
- “Users do not care about security, they want to do their work efficiently”

**Hints:**

1) Secure default settings (“do nothing”)
2) Indicate how to use the interface (“Perceived affordances”)
3) Secure way must not be inconvenient (provide payoff if inconvenience cannot be avoided)
Objects, Actors, and Actions

**Objects:**
- Files, data records, ...  
  Physical Stance

**Actors:**
- Applications  
  Design Stance
- Other users  
  Intentional Stance

**Actions:**
- Operation performed on an object (delete file, copy text, ...)  
- Performed by Actors
“A system is secure from a given user's perspective if the set of actions that each actor can do are bounded by what the user believes it can do.”
Principle of Appropriate Boundaries:

- Aggregate Actions/Actors in units that the user actually cares about
- Make boundaries relevant to security visible (e.g., applications)

Example: Granting authorities:

- Application spawns multiple helper processes
- Does the user have to grant authorities to each individual process?
## 10 Design Principles

<table>
<thead>
<tr>
<th></th>
<th>Principle</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Principle of Least Resistance</td>
<td>Fundamental</td>
</tr>
<tr>
<td>2</td>
<td>Principle of Appropriate Boundaries</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Principle of Explicit Authorization</td>
<td>Actor-Ability State</td>
</tr>
<tr>
<td>4</td>
<td>Principle of Visibility</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Principle of Revocability</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Principle of Expected Ability</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Principle of Trusted Path</td>
<td>Input/Output</td>
</tr>
<tr>
<td>8</td>
<td>Principle of Identifiability</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Principle of Expressiveness</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Principle of Clarity</td>
<td></td>
</tr>
</tbody>
</table>
Actor-Ability State

The user's model of the system:

◆ Actors: \( \{ A_0, A_1, A_2, \ldots, A_n \} \)
◆ Potential abilities: \( P_i \)
◆ Real abilities: \( R_i \)
◆ Actor-Ability State: \( \{ (A_0, P_0), (A_1, P_i), \ldots, (A_n, P_n) \} \)
◆ No-surprise condition:
  \[
  P_0 \leq R_0 \\
  P_i \geq R_i \quad (\text{for } i > 0)
  \]
Explicit Authorization

**Principle of Explicit Authorization:**
- Derived from “principle of least privilege”
- User can extend $A_i$'s real abilities $R_i$

**Example:** Opening files
- Application needs authorization to open a file
- Grant authorization through system interface:
  - Choose the file in the File-open dialog
  - Drag'n'drop
Visibility

**Principle of Visibility:**

- Actor-ability state represents the user's knowledge about the security of the system
- However, this view may be incomplete

- Make past granting actions visible to the user
- Inspect:
  - Holder of authority
  - Object
Principle of Revocability:

- Keep actor-ability state manageable
- Accommodate for error situations:
  - The user accidentally granted authorities
  - The user has been fooled about the true nature of an application
  - A security bug is identified
Expected Ability

**Principle of Expected Ability:**
- The user has an expectation of his future abilities that can have security implications

**Example:** Ability to revoke authorities

**Example:** Ability to discard data
- The user keeps records of private data that he wishes to delete at a later time
# 10 Design Principles

<table>
<thead>
<tr>
<th></th>
<th>Principle</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Principle of Least Resistance</td>
<td>Fundamental</td>
</tr>
<tr>
<td>2</td>
<td>Principle of Appropriate Boundaries</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Principle of Explicit Authorization</td>
<td>Actor-Ability State</td>
</tr>
<tr>
<td>4</td>
<td>Principle of Visibility</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Principle of Revocability</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Principle of Expected Ability</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Principle of Trusted Path</td>
<td>Input/Output</td>
</tr>
<tr>
<td>8</td>
<td>Principle of Identifiability</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Principle of Expressiveness</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Principle of Clarity</td>
<td></td>
</tr>
</tbody>
</table>
Principle of Trusted Path:
- Unspoofable and incorruptible channel to interact with the system

Example:
- Authorities may only be edited through a trustworthy user interface

Example:
- Windows Login Dialog: Ctrl-Alt-Del
Identifiability

**Principle of Identifiability:**
- Actions and objects must be identifiable

**Continuity:**
"The same thing should appear the same"

**Discriminability:**
"Different things should appear different"

The user must **perceive** things differently!
Identifiability (2)
Expressiveness

Principle of Expressiveness:

- The user specifies security policies according to his model of the system.

- To be useful, the system must allow the following:
  - The user can safely specify a security policy.
  - The user can express the security policy he wants.

Example: Standard Unix file-system ACLs.
Principle of Clarity:

Security policies must be expressible clearly:

“When the user is given control to manipulate authorities, we must ensure that the results reflect the user's intent.”
Summary

To be able to use a system safely, the user must have confidence in the following statements:

- Things don't become unsafe all by themselves. *(Explicit Authorization)*
- I can know whether things are safe. *(Visibility)*
- I can make things safer. *(Revocability)*
- I don't choose to make things unsafe. *(Path of Least Resistance)*
- I know what I can do with the system. *(Expected Ability)*
- I can distinguish the things that matter to me. *(Appropriate Boundaries)*
- I can tell the system what I want. *(Expressiveness)*
- I know what I'm telling the system to do. *(Clarity)*
- The system protects me from being fooled. *(Identifiability, Trusted Path)*
Points of Discussion

- Principle of Visibility: How can we avoid violating the Principle of Least Resistance?
- How do the principles depend on each other?
- Your questions?