



THE UNIVERSITY OF BRITISH COLUMBIA

# Security and Usability

EECE 412  
Session 23

# What's More Important:

The correctness of security  
functions/mechanisms,  
or  
the correct use of them?



# Outline

- Principles of secure interaction design
- Five lessons about usable security



# Usability and Security Tradeoffs

- A computer is secure from a particular user's perspective if the user can depend on it and its software to behave as the user expects.
- Acceptable security is a requirement for usability.
- Acceptable usability is a requirement for security.



# Secure Interaction Design

Ka-Ping Yee

with Norm Hardy, Mark Miller, Chip Morningstar, Kragen Sitaker, Marc Stiegler, Dean Tribble, and Miriam Walker

## Basic Concepts

### ACTOR-ABILITY MODEL

At any point in time, the user's model contains a set of **actors** in the system and a set of **potential actions** for each actor. For a system to be secure, the actual abilities of any actor must never come to exceed the bounds in the user model.

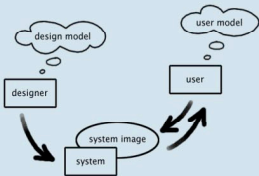
actors  $A = \{A_0, A_1, \dots, A_n\}$   
 perceived abilities  $P = \{P_0, P_1, \dots, P_n\}$   
 real abilities  $R = \{R_0, R_1, \dots, R_n\}$

$$P_0 \subseteq R_0$$

$$P_i \supseteq R_i \text{ for } i > 0$$

### SYSTEM IMAGE

The actors, actions, and objects in the user's mental model are derived from observing the **system image**, not from knowledge of its internal design.



### USERS AND USER AGENTS

The software system intended to serve and protect the interests of the user is the **user agent**. On a stand-alone PC, this is the operating system shell, through which the user interacts with an arena of entities such as files and programs. On a networked PC, a second level of user agent represents the user's interests in a larger arena of interacting computers.

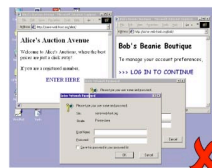
## Fundamental Principles

### Actor-Ability State

### Input and Output

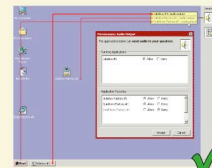
#### TRUSTED PATH

The interface must provide an unspooftable and faithful communication channel between the user and any entity trusted to manipulate authorities on the user's behalf.



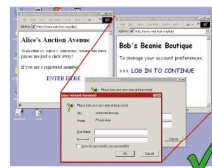
#### VISIBILITY

The interface should allow the user to easily review any active authority relationships that would affect security-relevant decisions.



#### REVOCABILITY

The interface should allow the user to easily revoke authorities that the user has granted, wherever revocation is possible.



#### IDENTIFIABILITY

The interface should enforce that distinct objects and distinct actions have unspooftably identifiable and distinguishable representations.

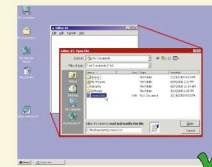


#### PATH OF LEAST RESISTANCE

The natural way to do any task should also be the secure way.

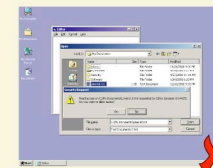
#### EXPLICIT AUTHORITY

A user's authorities must only be provided to other actors as a result of an explicit action that is understood by the user to imply granting.



#### EXPECTED ABILITY

The interface must not generate the impression that it is possible to do something that cannot actually be done.

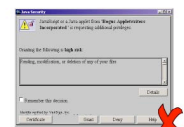


#### EXPRESSIVENESS

The interface should provide enough expressive power to (a) describe a safe security policy without undue difficulty and (b) allow users to express security policies in terms that fit their goals.

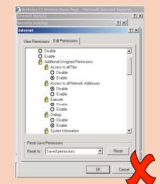
#### CLARITY

The effect of any security-relevant action must be clearly apparent to the user before the action is taken.



#### APPROPRIATE BOUNDARIES

The interface should expose distinctions between objects and between actions along boundaries that matter to the user.



# Principle 1: Path of Least Resistance

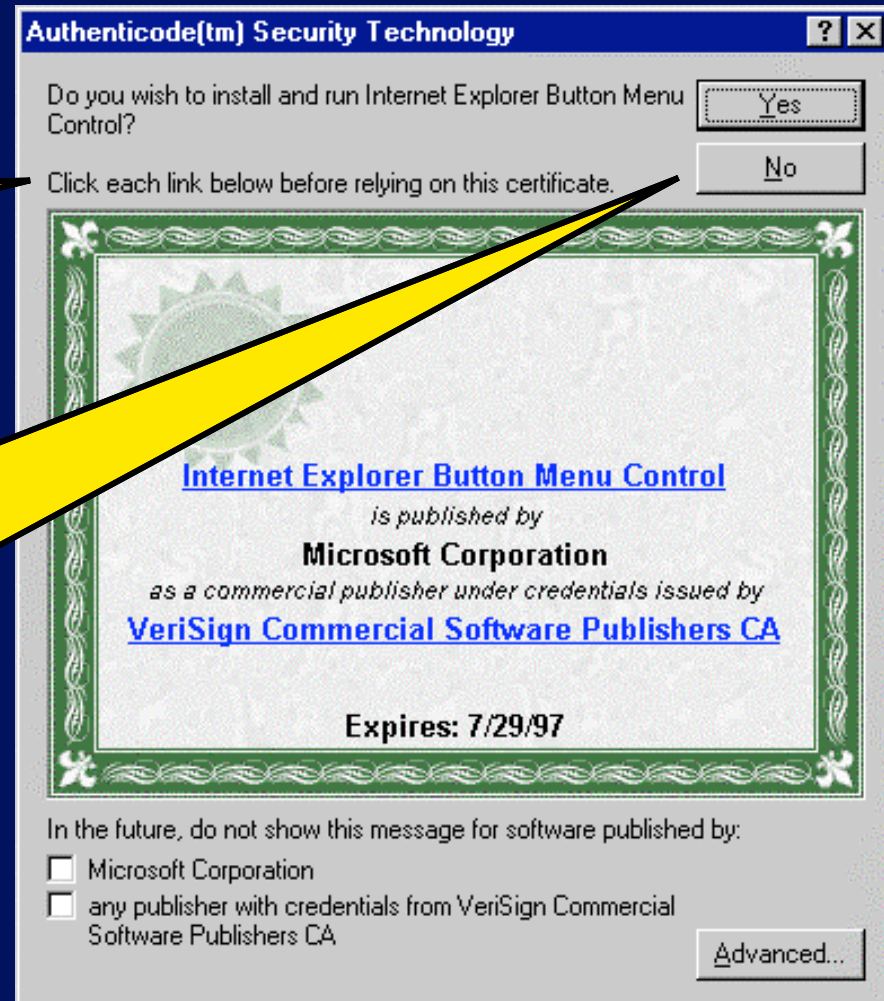
To the greatest extent possible,  
the **natural way** to do a task should be  
the **secure way**.



# Example 1: Least resistance



Click each link below before



## Principle 2: Appropriate Boundaries

The interface should expose, and the system should enforce, distinctions between objects and between actions that matter to the user.

I.e., any boundary that could have meaningful security implications to the user should be visible, and those that do not should not be visible.

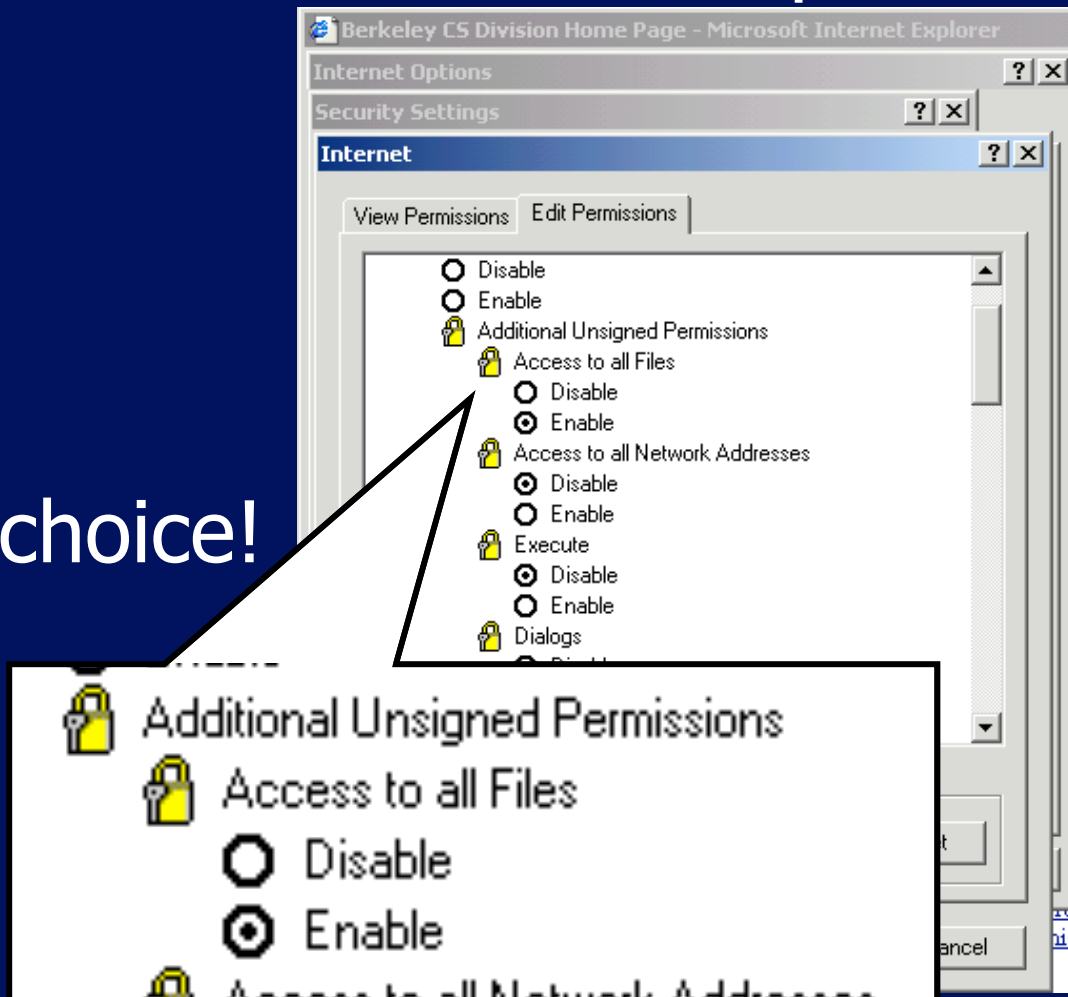




# Example 2: Bad boundaries

- A real dialog window in Internet Explorer:

- User is forced to make an all-or-nothing choice!

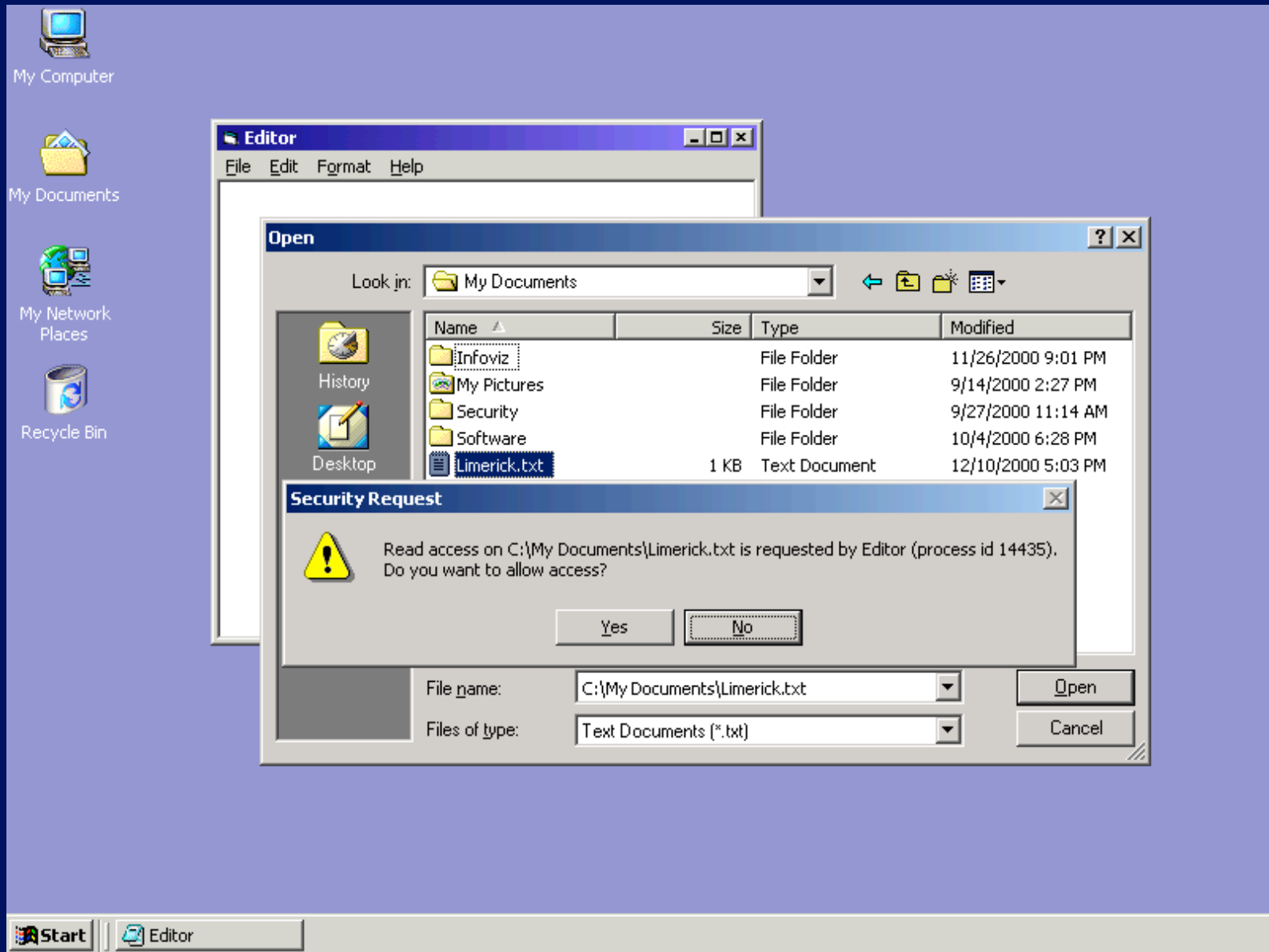


# Principle 3: Explicit Authorization

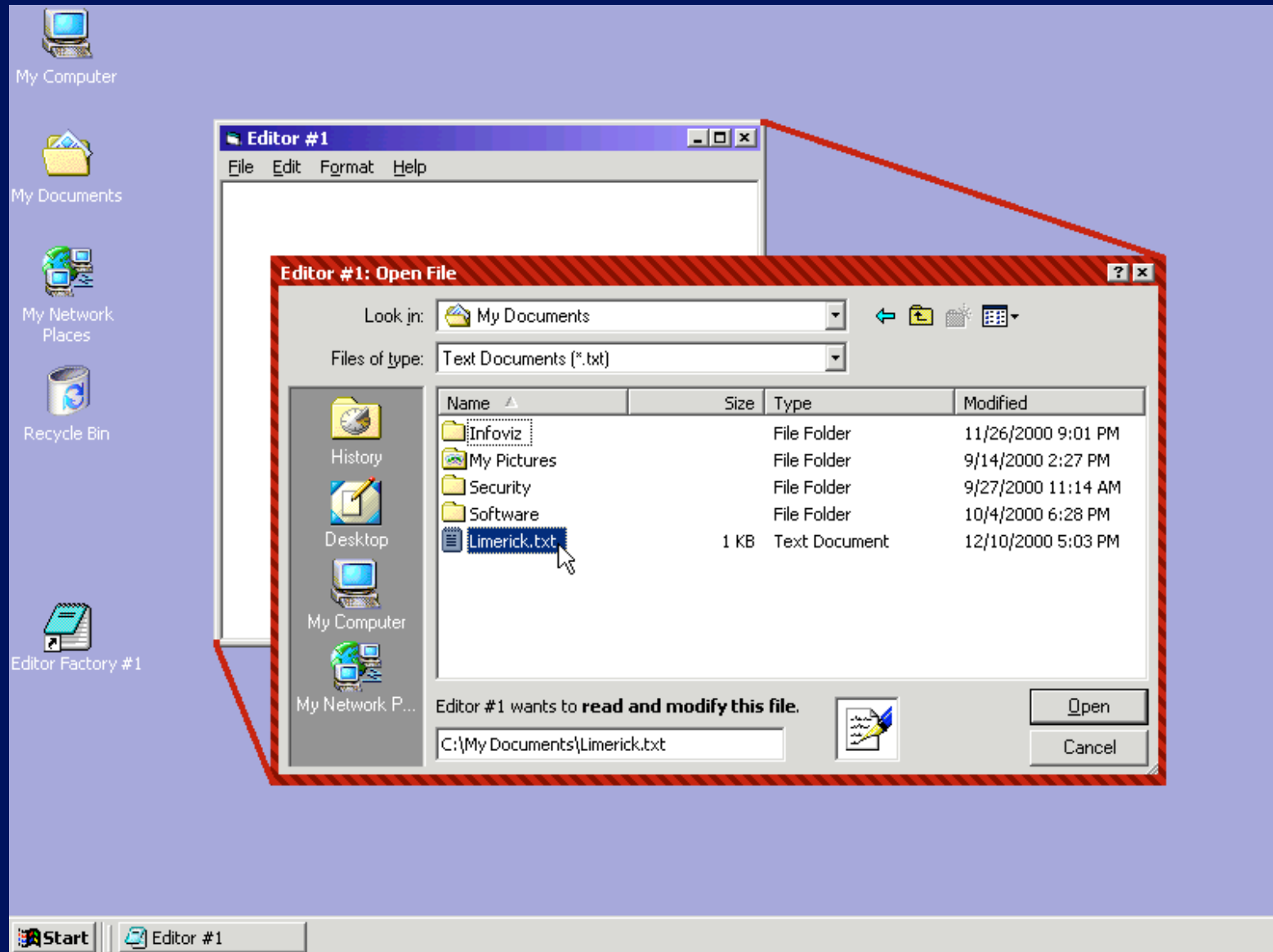
A user's authorities must only be provided to other actors as a result of an explicit action that is understood to imply granting.

- Conflicts with Least Resistance?
- Authorizes the increase of privileges
- Combining **designation** with **authorization**

# Example 3: When do we ask?



# Example 3: When do we ask?



# Principle 4: Visibility

The interface should allow the user to easily **review any active authorizations** that would **affect security-relevant decisions**.

# Example 4: What do we show?

Not this:

```
7:09am up 117 days, 6:02, 1 user, load average: 0.17, 0.23, 0.23
110 processes: 109 sleeping, 1 running, 0 zombie, 0 stopped
CPU states: 7.6% user, 4.5% system, 0.0% nice, 87.8% idle
Mem: 512888K av, 496952K used, 15936K free, 60K shrd, 29728K buff
Swap: 1052216K av, 146360K used, 905856K free 181484K cached
```

```
PID USER PRI NI SIZE RSS SHARE STAT %CPU %MEM TIME COMMAND
24733 root 18 0 2556 2556 488 S 6.0 0.4 1:42 chargen
25184 ping 16 0 996 996 748 R 3.9 0.1 0:01 top
24276 root 9 0 1888 1864 1484 S 0.7 0.3 0:04 sshd
23519 apache 10 0 21792 13M 8080 S 0.1 2.6 0:23 httpd
23520 apache 10 0 21456 12M 8076 S 0.1 2.5 0:20 httpd
1 root 8 0 188 148 148 S 0.0 0.0 0:25 init
2 root 9 0 0 0 0 SW 0.0 0.0 0:00 keventd
3 root 9 0 0 0 0 SW 0.0 0.0 0:00 kapm-idled
4 root 19 19 0 0 0 SWN 0.0 0.0 0:33 ksoftirqd_CPU0
5 root 9 0 0 0 0 SW 0.0 0.0 94:12 kswapd
6 root 9 0 0 0 0 SW 0.0 0.0 0:02 kreclaimd
7 root 9 0 0 0 0 SW 0.0 0.0 0:08 bdflush
8 root 9 0 0 0 0 SW 0.0 0.0 0:15 kupdated
9 root -1 -20 0 0 0 SW< 0.0 0.0 0:00 mdrecoveryd
654 root 9 0 348 288 288 S 0.0 0.0 2:41 syslogd
659 root 9 0 852 120 120 S 0.0 0.0 0:06 klogd
744 root 9 0 1988 1988 1728 S 0.0 0.3 0:07 ntpd
757 daemon 9 0 172 116 116 S 0.0 0.0 0:00 atd
786 root 9 0 360 232 200 S 0.0 0.0 0:03 sshd
807 root 8 0 476 336 292 S 0.0 0.0 0:56 xinetd
866 root 8 0 396 332 312 S 0.0 0.0 0:34 crond
915 root 9 0 2076 476 476 S 0.0 0.0 0:25 miniserv.pl
919 root 9 0 108 48 48 S 0.0 0.0 0:00 mingetty
920 root 9 0 108 48 48 S 0.0 0.0 0:00 mingetty
```



# Example 4: What do we show?

The screenshot shows a Windows XP desktop environment. On the left, the Start menu is open, displaying icons for 'My Computer', 'My Documents', 'My Network Places', 'Recycle Bin', and 'Editor Factory #1'. In the center, a 'Jukebox Factory #1' icon is visible. A red dotted line traces a path from the Start menu, through the desktop icons, and to the 'Jukebox #1' taskbar icon. A red box highlights a tooltip in the system tray that lists three audio output devices: 'Jukebox #1: audio output', 'Jukebox Factory #1: audio output', and 'Quicktime Factory #1: audio output'. A mouse cursor is pointing at the first device. In the foreground, a 'Permissions: Audio Output' dialog box is open, displaying the following information:

**Permissions: Audio Output**

The applications below can **send audio to your speakers.**

**Running Applications**

Jukebox #1	<input checked="" type="radio"/> Allow	<input type="radio"/> Deny
------------	--	----------------------------

**Application Factories**

Jukebox Factory #1	<input checked="" type="radio"/> Allow	<input type="radio"/> Deny
Quicktime Factory #1	<input checked="" type="radio"/> Allow	<input type="radio"/> Deny
RealPlayer Factory #1	<input type="radio"/> Allow	<input checked="" type="radio"/> Deny

Buttons: Accept, Cancel

# Principle 5: Identifiability

The interface should enforce that distinct objects and distinct actions have unspoofably identifiable and distinguishable representations.

two aspects

- Continuity: the same thing should appear the same
  - Discriminability: different things should appear different
- *perceived vs. be* different/same



# Example 5: Violating identifiability

The screenshot illustrates a security vulnerability where a user's login credentials are reused across different websites. Two Internet Explorer windows are open side-by-side:

- Alice's Auction Avenue** (Address: <http://some-web-host.org/alice/>):  
Welcome to Alice's Auctions, where the best prices are just a click away!  
If you are a registered member,  
**ENTER HERE**
- Bob's Beanie Boutique** (Address: <http://some-web-host.org/bob/>):  
To manage your account preferences,  
**>>> LOG IN TO CONTINUE**

Two overlapping "Enter Network Password" dialog boxes are shown in the foreground. The front-most dialog box displays the following information:

- Site: some-web-host.org
- Realm: Private Area
- User Name: [input field]
- Password: [input field]
- Save this password in your password list

The dialog box also includes "OK" and "Cancel" buttons. The background dialog box is partially obscured and shows a similar prompt.

# Example 5: Fixing identifiability

The screenshot shows a Windows desktop environment with two Internet Explorer windows open. The left window, titled "Alice's Auction Avenue", has the address bar set to "http://some-web-host.org/alice/" and displays the text "Alice's Auction Avenue", "Welcome to Alice's Auctions, where the best prices are just a click away!", "If you are a registered member,", and a blue link "ENTER HERE". The right window, titled "Bob's Beanie Boutique", has the address bar set to "http://some-web-host.org/bob/" and displays "Bob's Beanie Boutique", "To manage your account preferences,", and a blue link ">>> LOG IN TO CONTINUE".

Two "Enter Network Password" dialog boxes are overlaid on the desktop. The one in the foreground is highlighted with a red dashed border and contains the following fields and options:

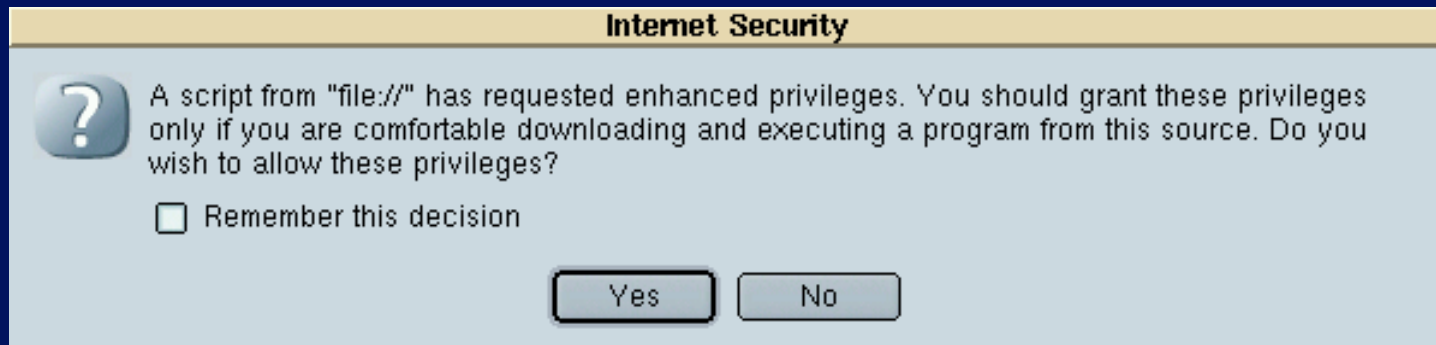
- Site: some-web-host.org
- Realm: Private Area
- User Name: [text input field]
- Password: [password input field]
- Save this password in your password list
- Buttons: OK, Cancel

Red lines connect the "ENTER HERE" link in the Alice window to the highlighted password dialog, and the "LOG IN TO CONTINUE" link in the Bob window to the same dialog, illustrating a security vulnerability where a single password prompt can be used to access multiple sites.

# Principle 6: Clarity

The effect of any security-relevant action must be apparent before the action is taken.

# Example 6: Violating Clarity



What program?

What privileges?

How long?

Remember this decision?

What source?

What purpose?

How to revoke?

*What* decision?

Might as well click "Yes": it's the default.

# Principle 7: Expressiveness

In order for the security policy enforced by the system to be useful, we must be able to express a **safe policy**, and we must be able to express the **policy we want**.

# Example 7: Unix File Permissions

```
rw-r--r-- 1 konstant konstant 89418 18 Oct 13:57 Berry 2002 painpaper.pdf
rwxr--r-- 1 konstant konstant 3639577 8 Oct 17:32 MarineAquarium206_OSX.dmg
drwxrwxrwx 3 konstant konstant 102 17 Oct 08:11 My Great DVD.dvdproj
rw-r--r-- 1 konstant konstant 50536 18 Oct 13:57 Shaw 2001.pdf
drwxr-xr-x 267 konstant konstant 9078 25 Nov 11:33 downloads
rw-r--r-- 1 konstant konstant 9204 29 Aug 14:29 konstantin_beznosov_thumbnail.jpg
rw-r--r-- 1 konstant konstant 158195 18 Oct 13:57 shaw 2002 SE rsrch.pdf
rw-r--r-- 1 konstant konstant 255671 18 Oct 13:57 shaw 2003 -icse03.pdf
rw-r--r-- 1 konstant konstant 5318 9 Oct 23:16 sidney_fels.jpg
rw-r--r-- 1 konstant konstant 139 22 Nov 13:09 wcsf-notes.rtf
```



# Design Principles Summary

In order to use a system safely, a **user needs to have confidence** in all of the following statements:

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

# Lessons learned about usable security

1. You cannot retrofit usable security
  - Adding explanatory dialogs to a confusing system makes it more confusing
2. Tools are not solutions
  - They are just Lego™ blocks
3. Mind the upper layers
  - Application-level security design allows intentional, implicit, application-specific security
4. Keep your users satisfied
  - Put your users' needs first
  - Evaluate your solution on the target audience
5. Think locally, act locally
  - Don't assume support from global infrastructure (e.g., PKI)
  - If a generic security tool can be used independently of application, the user(s) must deal with it directly



# Where To Go From Here

## Continue University Education

- UBC Undergrad. Research Conference, every March
- EECE 496: do a security project
- Undergraduate Student Research Assistantship (USRA) from NSERC
  - Get paid during summer while doing security research!
  - Application deadline some time in March. Talk to Dr. Beznosov
- Other security-related courses
  - EECE 512: grad course will help to start security research at grad level
  - MATH 342 "Algebra, Coding Theory, and Cryptography"
  - COMM 456 "Control and Security of Information Systems" at [mis.commerce.ubc.ca](http://mis.commerce.ubc.ca)

## Self Education

- Read good books on security. See EECE 412 resources page
- Keep up to date
  - IEEE Security & Privacy Magazine
    - Online -- free for UBC students
    - Paper -- subscription-based
  - Conferences
    - Local
      - West Coast Security Forum, every November in Vancouver, [www.wcsf.com](http://www.wcsf.com)
      - CanSecWest, May 4-6, 2005, Vancouver, [www.cansecwest.com](http://www.cansecwest.com)
  - Security professional groups:
    - CIPS Vancouver Security SIG
      - [www.infosecbc.org](http://www.infosecbc.org)
      - Monthly every first Wednesday 2PM -- 4 PM

