Paper Reading Group

User Interaction Design for Secure Systems

Ka-Ping Yee, 2002

Carsten Weinhold

<weinhold@os.inf.tu-dresden.de>

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

Fundamental

1) Principle of Least Resistance
2) Principle of Appropriate Boundaries

Actor-Ability State

3) Principle of Explicit Authorization
4) Principle of Visibility
5) Principle of Revocability
6) Principle of Expected Ability

Input/Output

7) Principle of Trusted Path
8) Principle of Identifiability
9) Principle of Expressiveness
10) Principle of Clarity

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"

→ Path of Least Resistance

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*

Objects, Actors, and Actions (2)

“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.”

Aggregation and Appropriate Boundaries

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

Fundamental

1) Principle of Least Resistance
2) Principle of Appropriate Boundaries

Actor-Ability State

3) Principle of Explicit Authorization
4) Principle of Visibility
5) Principle of Revocability
6) Principle of Expected Ability

Input/Output

7) Principle of Trusted Path
8) Principle of Identifiability
9) Principle of Expressiveness
10) Principle of Clarity

Actor-Ability State

The user's model of the system:

- Actors: $\{ A_0, A_1, A_2, ..., A_n \}$
- Potential abilities: P_i
- Real abilities: R_i
- Actor-Ability State: $\{ (A_0, P_0), (A_1, P_i), ..., (A_n, P_n) \}$
- No-surprise condition:

$$P_0 \leq R_0$$

$$P_i \geq R_i \qquad \textit{(for i > 0)}$$

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

Revocability

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

Fundamental

1) Principle of Least Resistance
2) Principle of Appropriate Boundaries

Actor-Ability State

3) Principle of Explicit Authorization
4) Principle of Visibility
5) Principle of Revocability
6) Principle of Expected Ability

Input/Output

7) Principle of Trusted Path
8) Principle of Identifiability
9) Principle of Expressiveness
10) Principle of Clarity

Trusted Path

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:

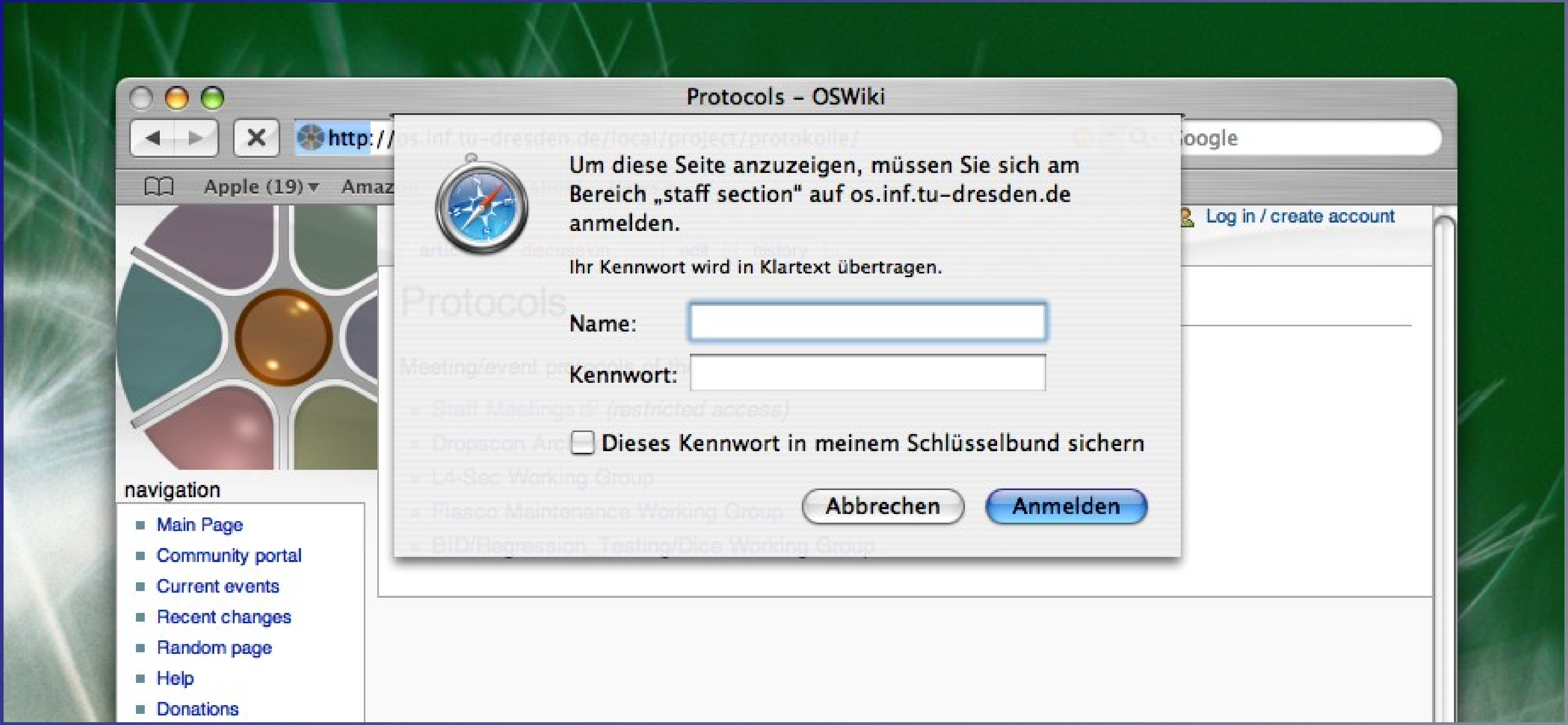
- Actions and objects must identifiable

Continuity:
“The same thing should appear the same”

Discriminability:
“Different things should appear different”

The user must ***perceive*** things different!

Identifiability (2)



Protocols – OSWiki

http://os.inf.tu-dresden.de/local/project/protokolle/

Um diese Seite anzuzeigen, müssen Sie sich am Bereich „staff section" auf os.inf.tu-dresden.de anmelden.

Ihr Kennwort wird in Klartext übertragen.

Name:

Kennwort:

□ Dieses Kennwort in meinem Schlüsselbund sichern

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

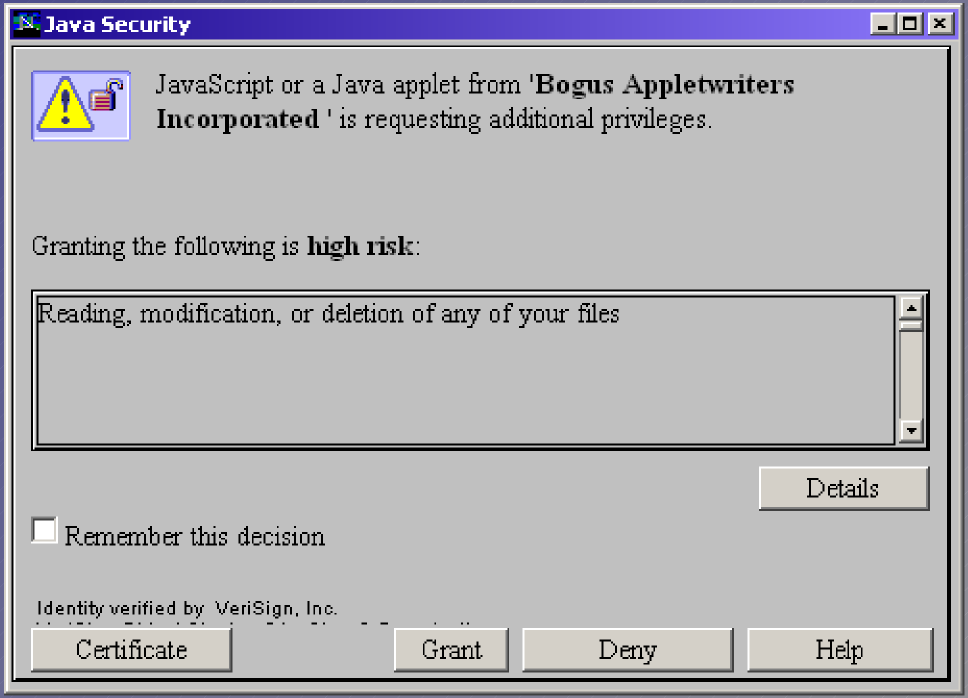
Example: Standard Unix file-system ACLs

Clarity

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.”



JavaScript or a Java applet from '**Bogus Appletwriters Incorporated** ' is requesting additional privileges.

Granting the following is **high risk**:

Reading, modification, or deletion of any of your files

□ Remember this decision

Identity verified by VeriSign, Inc.

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 priniciples depend on each other?
- Your questions?