Design

Secure Application Development
Modules 12, 13
Konstantin Beznosov
What Do you Already Know?

- What principles of designing secure systems do you already know?
- What anti-principles do you know?
Outline

- Overview
- Principles of designing secure systems
- Principles of designing usable security
Principles of Designing Secure Systems
Design Principles Outline

- Overview
- Principles
  1. Least Privilege
  2. Fail-Safe Defaults
  3. Economy of Mechanism
  4. Complete Mediation
  5. Open Design
  6. Separation of Privilege
  7. Least Common Mechanism
  8. Psychological Acceptability
  9. Defense in Depth
  10. Question Assumptions

J. Saltzer and M. Schroeder
"The Protection of Information in Computer Systems" 1975
Introductory Example: Sendmail

- Editor
- Config. Text
- Sendmail compiler
- Config. binary
Overarching Goals

- **Simplicity**
  - Less to go wrong
  - Fewer possible inconsistencies
  - Easy to understand

- **Restriction**
  - Minimize access
    - “need to know” policy
  - Inhibit communication to minimize abuse of the channels
Example 1: Privileges in Operating Systems

- Until Windows NT, all privileges for everybody
- Separate admin (a.k.a., root) account on Windows and Unix
  - Ways to switch between accounts
Example 2: RBAC

Differentiation between assigned and activated roles

- Manager
  - Senior Administrator
  - Administrator
  - Employee
- Senior Engineer
  - Engineer
Principle 1: Least Privilege

Every program and every user of the system should operate using the least set of privileges necessary to complete the job.

- Rights added as needed, discarded after use
- Limits the possible damage
- Unintentional, unwanted, or improper uses of privilege are less likely to occur
- Guides design of protection domains
Example 3: Temporary Upgrade of the Privileges in MacOS

- `sudo` command on MacOS
- admin account authentication
Principle 2: Fail-Safe Defaults

Base access decisions on permission rather than exclusion.

- Default action is to deny access
- If action fails, system as secure as when action began

suggested by E. Glaser in 1965
Principle 3: Economy of Mechanism

Keep the design as simple and small as possible.

- KISS Principle
- Rationale?
- Essential for analysis
- Simpler means less can go wrong
  - And when errors occur, they are easier to understand and fix
Example 4: .rhosts mechanism abused by 1988 Internet Worm

Access to one account opened unchecked access to other accounts on different hosts
Principle 4: Complete Mediation

Every access to every object must be checked for authority.

- If permissions change after, may get unauthorized access
Example 5: Multiple reads after one check

- Process rights checked at file opening
- No checks are done at each read/write operation
- Time-of-check to time-of-use
Kerckhoff’s Principle

“The security of a cryptosystem must not depend on keeping secret the crypto-algorithm. The security depends only on keeping secret the key”

Auguste Kerckhoff von Nieuwenhof
Dutch linguist
1883
Principle 5: Open Design

Security should not depend on secrecy of design or implementation

P. Baran, 1965

- “Security through obscurity”
- Does not apply to information such as passwords or cryptographic keys
Example 6: Content Scrambling System

- DVD content
  - SecretEncrypt(Movie, $K_T$)
  - SecretEncrypt($K_T$, $K_D$)
  - Hash($K_D$)
  - SecretEncrypt($K_D$, $K_{p1}$)
  - ...
  - SecretEncrypt($K_D$, $K_{pn}$)

- 1999
  - Norwegian group derived SecretKey by using $K_{Pi}$
  - Plaintiff’s lawyers included CSS source code in the filed declaration
  - The declaration got out on the internet
Example 7: Getting *root* Access in BSD Unix

Two-conditions for getting root access

- **Knowledge** of *root* password
- Group *wheel* membership
Principle 6: Separation of Privilege

Require multiple conditions to grant permission

R. Needham, 1973

- Similar to separation of duty

- Another example:
  - Two-factor authentication
**Principle 7:**

**Least Common Mechanism**

Mechanisms should not be shared

- Information can flow along shared channels in uncontrollable way
- Covert channels
- Isolation
  - Virtual machines
  - Sandboxes
Example 8: Switching between user accounts

- Windows NT -- pain in a neck
- Windows 2000/XP -- “Run as …”
- Unix -- “su” or “sudo”
Principle 8: Psychological Acceptability

Security mechanisms should not add to difficulty of accessing resource

- Hide complexity introduced by security mechanisms
- Ease of installation, configuration, use
- Human factors critical here
### Example 9: Windows Server 2003

<table>
<thead>
<tr>
<th>Potential problem</th>
<th>Mechanism</th>
<th>Practice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer overflow</td>
<td>defensive programming</td>
<td>check preconditions</td>
</tr>
<tr>
<td>Even if it were vulnerable</td>
<td>IIS 6.0 is not up by default</td>
<td>no extra function-ty</td>
</tr>
<tr>
<td>Even if IIS were running</td>
<td>default URL length 16 KB</td>
<td>conservative limits</td>
</tr>
<tr>
<td>Even if the buffer were large</td>
<td>the process crashes</td>
<td>fail-safe</td>
</tr>
<tr>
<td>Even if the vulnerability were expl.</td>
<td>Low privileged account</td>
<td>least privileged</td>
</tr>
</tbody>
</table>
Principle 9: Defense in Depth

Layer your defenses
Example 10: Assumptions, Assumptions, ...

- *ident*
- *finger* protocol
Example 11: Assuming Honest Client

- Web server application requires browser to validate input

- Attack plan:
  - Remove the client from the communications loop and talk directly to the server
  - Leverage incorrect trust model, trusting the client
Principle 10: Question Assumptions

Frequently re-examine all the assumptions about the threat agents, assets, and especially the environment of the system.
Summary

- Overarching Goals

- Principles
  1. Least Privilege
  2. Fail-Safe Defaults
  3. Economy of Mechanism
  4. Complete Mediation
  5. Open Design
  6. Separation of Privilege
  7. Least Common Mechanism
  8. Psychological Acceptability
  9. Defense in depth
  10. Question assumptions
Principles of Designing Usable Security
"Humans are incapable of securely storing high-quality cryptographic keys, and they have unacceptable speed and accuracy when performing cryptographic operations. (They are also large, expensive to maintain, difficult to manage, and they pollute the environment. It is astonishing that these devices continue to be manufactured and deployed. But they are sufficiently pervasive that we must design our protocols around their limitations.)"

Charlie Kaufman, Radia Perlman, Mike Speciner in "Network Security: Private Communication in a Public World"
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

Basic Concepts

**ACTORABILITY 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** $A = \{A_0, A_1, ..., A_n\}$
- **Perceived abilities** $P = \{P_0, P_1, ..., P_n\}$
- **Real abilities** $R = \{R_0, R_1, ..., R_n\}$

$$P_0 \subseteq R_0$$
$$P_i \subseteq R_i \text{ for } i > 0$$

**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.

**INPUT AND OUTPUT**

**USER 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**

**PATH OF LEAST RESISTANCE**

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

**APPROPRIATE BOUNDARIES**

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

**VISIBILITY**

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

**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.

**REVOCABILITY**

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

**EXPECTED ABILITY**

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

**TRUSTED PATH**

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

**IDENTIFIABILITY**

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

**EXPRESSIONSIVENESS**

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.
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 relying on this certificate.

Internet Explorer Button Menu Control
is published by
Microsoft Corporation
as a commercial publisher under credentials issued by
VeriSign Commercial Software Publishers CA
Expires: 7/29/97

In the future, do not show this message for software published by:
- Microsoft Corporation
- any publisher with credentials from VeriSign Commercial Software Publishers CA
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?

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PRI</th>
<th>NI</th>
<th>SIZE</th>
<th>RSS</th>
<th>SHARE</th>
<th>STAT</th>
<th>%CPU</th>
<th>%MEM</th>
<th>TIME</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>24733</td>
<td>root</td>
<td>18</td>
<td>0</td>
<td>2556</td>
<td>2556</td>
<td>488</td>
<td>S</td>
<td>6.0</td>
<td>0.4</td>
<td>1:42</td>
<td>chargen</td>
</tr>
<tr>
<td>25184</td>
<td>ping</td>
<td>16</td>
<td>0</td>
<td>996</td>
<td>996</td>
<td>748</td>
<td>R</td>
<td>3.9</td>
<td>0.1</td>
<td>0.01</td>
<td>top</td>
</tr>
<tr>
<td>24276</td>
<td>root</td>
<td>9</td>
<td>0</td>
<td>1888</td>
<td>1864</td>
<td>1484</td>
<td>S</td>
<td>0.7</td>
<td>0.3</td>
<td>0.04</td>
<td>sshd</td>
</tr>
<tr>
<td>23519</td>
<td>apache</td>
<td>10</td>
<td>0</td>
<td>21752</td>
<td>13M</td>
<td>708</td>
<td>S</td>
<td>0.1</td>
<td>2.6</td>
<td>0.23</td>
<td>httpd</td>
</tr>
<tr>
<td>23520</td>
<td>apache</td>
<td>10</td>
<td>0</td>
<td>21456</td>
<td>12M</td>
<td>706</td>
<td>S</td>
<td>0.1</td>
<td>2.5</td>
<td>0.20</td>
<td>httpd</td>
</tr>
<tr>
<td>1</td>
<td>root</td>
<td>8</td>
<td>0</td>
<td>188</td>
<td>148</td>
<td>148</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0.25</td>
<td>init</td>
</tr>
<tr>
<td>2</td>
<td>root</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SW</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
<td>keventd</td>
</tr>
<tr>
<td>3</td>
<td>root</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SW</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
<td>kapm-idled</td>
</tr>
<tr>
<td>4</td>
<td>root</td>
<td>19</td>
<td>-20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SWN</td>
<td>0.0</td>
<td>0.0</td>
<td>0.33</td>
<td>ksoftirqd_CPU0</td>
</tr>
<tr>
<td>5</td>
<td>root</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SW</td>
<td>0.0</td>
<td>0.0</td>
<td>94.12</td>
<td>kswapd</td>
</tr>
<tr>
<td>6</td>
<td>root</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SW</td>
<td>0.0</td>
<td>0.0</td>
<td>0.02</td>
<td>kcreclaim</td>
</tr>
<tr>
<td>7</td>
<td>root</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SW</td>
<td>0.0</td>
<td>0.0</td>
<td>0.08</td>
<td>bdfflush</td>
</tr>
<tr>
<td>8</td>
<td>root</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SW</td>
<td>0.0</td>
<td>0.0</td>
<td>0.15</td>
<td>kupdated</td>
</tr>
<tr>
<td>9</td>
<td>root</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SW&lt;</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
<td>mdrecoveryd</td>
</tr>
<tr>
<td>654</td>
<td>root</td>
<td>9</td>
<td>0</td>
<td>348</td>
<td>288</td>
<td>288</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>2.41</td>
<td>syslogd</td>
</tr>
<tr>
<td>659</td>
<td>root</td>
<td>9</td>
<td>0</td>
<td>852</td>
<td>120</td>
<td>120</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0.06</td>
<td>klogd</td>
</tr>
<tr>
<td>744</td>
<td>root</td>
<td>9</td>
<td>0</td>
<td>1988</td>
<td>1788</td>
<td>1788</td>
<td>S</td>
<td>0.0</td>
<td>0.3</td>
<td>0.07</td>
<td>ntpd</td>
</tr>
<tr>
<td>757</td>
<td>daemon</td>
<td>9</td>
<td>0</td>
<td>172</td>
<td>116</td>
<td>116</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
<td>atd</td>
</tr>
<tr>
<td>786</td>
<td>root</td>
<td>9</td>
<td>0</td>
<td>360</td>
<td>232</td>
<td>232</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0.03</td>
<td>sshd</td>
</tr>
<tr>
<td>807</td>
<td>root</td>
<td>8</td>
<td>0</td>
<td>476</td>
<td>336</td>
<td>322</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0.56</td>
<td>xinetd</td>
</tr>
<tr>
<td>866</td>
<td>root</td>
<td>8</td>
<td>0</td>
<td>396</td>
<td>332</td>
<td>312</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0.34</td>
<td>crond</td>
</tr>
<tr>
<td>915</td>
<td>root</td>
<td>9</td>
<td>0</td>
<td>2076</td>
<td>476</td>
<td>476</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0.25</td>
<td>miniserv.pl</td>
</tr>
<tr>
<td>919</td>
<td>root</td>
<td>9</td>
<td>0</td>
<td>108</td>
<td>48</td>
<td>48</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
<td>mingetty</td>
</tr>
<tr>
<td>920</td>
<td>root</td>
<td>9</td>
<td>0</td>
<td>108</td>
<td>48</td>
<td>48</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
<td>mingetty</td>
</tr>
</tbody>
</table>

Not this:
Example 4: What do we show?
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

Alice's Auction Avenue
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
To manage your account preferences, >>> LOG IN TO CONTINUE
Example 5: Fixing identifiability
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 source?
What privileges?  What purpose?
How long?  How to revoke?
Remember this decision?  *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

<table>
<thead>
<tr>
<th>Permissions</th>
<th>Owner</th>
<th>Group</th>
<th>Size</th>
<th>Date</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>rwxr-xr-x</td>
<td>konstant konstant</td>
<td>3639577</td>
<td>8 Oct 17:32</td>
<td>MarineAquarium206_OSX.dmg</td>
<td></td>
</tr>
<tr>
<td>drwxrwxrwx</td>
<td>konstant konstant</td>
<td>102</td>
<td>17 Oct 08:11</td>
<td>My Great DVD.dvdproj</td>
<td></td>
</tr>
<tr>
<td>drwxr-xr-x</td>
<td>konstant konstant</td>
<td>267</td>
<td>25 Nov 11:33</td>
<td>downloads</td>
<td></td>
</tr>
<tr>
<td>rwxr-xr-x</td>
<td>konstant konstant</td>
<td>9204</td>
<td>29 Aug 14:29</td>
<td>konstantin_beznosov_thumbnail.jpg</td>
<td></td>
</tr>
<tr>
<td>rwxr-xr-x</td>
<td>konstant konstant</td>
<td>5318</td>
<td>9 Oct 23:16</td>
<td>sidney_fels.jpg</td>
<td></td>
</tr>
<tr>
<td>rwxr-xr-x</td>
<td>konstant konstant</td>
<td>139</td>
<td>22 Nov 13:09</td>
<td>wcse-notes.rtf</td>
<td></td>
</tr>
</tbody>
</table>
Usable Security 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