Design Principles and Patterns for Computer Systems that are Simultaneously Secure and Usable

Simson L. Garfinkel, April 26, 2005
A pattern is a recurring solution to a standard problem.

Patterns and “pattern languages” introduced by Architect Christopher Alexander in the 1970s.
Object Oriented Design adopted patterns in the 1990s. Johnson *et al.*, [OOPSLA 91]; Coad [CACM, 92]; “Gang of four” [95]

Why? Because patterns help us:

- reuse successful practices
- reason about what’s done and why
- document abstractions other than algorithms and data structures. [Schmidt *et al.*, 1996]

Patterns encapsulate knowledge and understanding, making it easier to teach and deploy solutions.
My thesis:

Usability and security can be made synergistic by redesigning systems with **specific principles** and through the adoption of **well-defined patterns**.
It has long been recognized that end-user security and usability are at odds in modern computer systems.

Username: simsong
Password: ••••••
The need to align end-user security and usability is recognized as a priority for both computing and the nation.

- CRA 2003 “Grand Challenge”
- PITAC 2005 “priority”
- Special publications [IEEE S&P 2004] [O’Reilly 2005]
- CHI 2005; SOUPS 2005

The traditional antagonism between usability and security can no longer be tolerated.
The root of the conflict: security and usability are different skills that *must both be applied from the beginning*.

**HCI-SEC:** The emerging field that seeks to align Human Computer Interfaces with Security.
Today computer security has many “principles,” “best practices” and “techniques.”

- Biometric authentication

- The password field

- Wrapping plaintext protocols with SSL
With patterns, we can decompose the problems and refactor the solutions.

Patterns are an easy way to communicate solutions to students, implementors, and organizations.
This talk presents two sets of related patterns for aligning usability and security.

1. Introduction to patterns.

2. Prior work in HCI-SEC.

3. Patterns for sanitization.

4. Patterns for secure messaging.
HCI-SEC seems hard because little work has been done!
I am going to focus on four HCI-SEC articles:

- Saltzer & Schroeder
  The Protection of Information in Computer Systems
  1975

- Whitten & Tygar
  Why Johnny Can't Encrypt
  1999

- Adams & Sasse
  Users are Not The Enemy
  1999

- Morris & Thompson
  Password Security: A Case History
  1979
Saltzer & Schroeder: 1975

- Introduced the term “Psychological Acceptability”
- “… so that users routinely and automatically apply the protection mechanisms correctly.”
- Mental images should match protection mechanisms.

[SS 75] argues that security should naturally emerge from normal operations.
Karat: Iterative Usability Testing [1989]

- Applies user-centered design techniques to an IBM security application deployed to 23,000 users.
- Articulates a *usability goal* — “95% of users will complete the sign-in task error free within the first three attempts.”
- Conducts field study; lab study; low-fidelity prototypes; live code tests;

**Karat and others (Sasse) argue that HCI-SEC is really just a usability problem.**
Whitten & Tygar identified five properties of “security software” [99]

- The secondary goal property
- The hidden failure property
- The barn door property
- The weakest link property
- The abstraction property

Primarily based on a study of PGP (secure messaging).
Yee [2002] argues that there is a fundamental mismatch between software capabilities and the user’s mental models.

Yee’s 10 principles for aligning security and usability primarily address virus and spyware problems.
My work builds these ideas, with specific techniques in two key areas:

Sanitization:
Patterns that address the problem of confidential information left behind on computer media and in applications.

Secure Messaging:
Patterns that increase the security of email today and point the way to future improvements.
The Sanitization Problem: Confidential information is left behind after it is no longer needed.

Data discovered on second-hand hard drives is an obvious case.

- Woman in Nevada bought a used PC with pharmacy records [Markoff 97]
- Paul McCartney’s bank records sold by his bank [Leyden 04]
- Pennsylvania sold PCs with “thousands of files” on state employees [Villano 02]
We found:

- Thousands of credit card numbers (many disks)
- Financial records
- Medical information
- Trade secrets
- Highly personal information

We did not determine if this was a *usability* problem
or an *education* problem.
Evidence for the usability problem: Computers *lie* when users delete data.

DEL removes file names

—but not file contents.

FORMAT claims “ALL DATA ... WILL BE LOST”

—but it’s not.

“...a fundamental mismatch between software capabilities and the user’s mental models.” [SS 75]
Oliver North had a mismatched mental model.

“We all sincerely believed that when we ... pressed the button 'delete' that it was gone forever.

Wow, were we wrong.”
— Oliver North, 1987
Evidence for an educational problem:
There is a huge secondary market for used disk drives.

- Re-used within organizations
- Given to charities
- Sold on eBay

People could just be discarding disk drives without thinking about the consequences.
To be effective, patterns should address the root cause of the problem.

**Usability Problem:**

- Effective audit of information present on drives.
- Make DEL and FORMAT actually remove data. [Bauer & Priyantha 01]
- Provide alternative strategies for data recovery.

**Education Problem:**

- Add training to the interface. [Whitten 04]
- Regulatory requirements. [FTC 05, SEC 05]
- Legal liability.

To determine the root cause, I looked on the drives and contacted the data subjects.
Data on a hard drive is arranged in blocks.

The white blocks indicate directories and files that are visible to the user.
Data on a hard drive is arranged in blocks.

The brown blocks indicate files that were deleted.
Data on a hard drive is arranged in blocks.

The green blocks indicate blocks that were never used (or that were wiped clean).
Stack the disk blocks:

- Files
- Deleted Files
- Zero Blocks
NO DATA: The disk is factory fresh.

All Blocks are Zero
The disk has an empty file system
AFTER OS INSTALL: Temp. files have been deleted

Free Blocks
Deleted temporary files
OS and Applications
AFTER A YEAR OF SERVICE

... 1 year ...

Blocks never written

Deleted files

OS, Applications, and user files
DISK NEARLY FULL!

... 1 year ...

OS, Apps, user files, and lots of MP3s!
FORMAT C:\ (to sell the computer.)

... 1 year ...

Recoverable Data
We can use forensics to reconstruct motivations:

- Training failure
- Usability failure
The 236 drives are dominated by failed sanitization attempts.

But training failures are also important.
But what *really* happened?

To answer this question, I needed to contact the original drive owners.
The Remembrance of Data Passed Traceback Study.

1. Find data on hard drive
2. Determine the owner
3. Get contact information for organization
4. Find the right person inside the organization
5. Set up interviews
6. Follow guidelines for human subjects work

This was a lot harder than I thought it would be.

<table>
<thead>
<tr>
<th>Date</th>
<th>File Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>06/19/1999</td>
<td>dir216/Four H Resume.doc</td>
</tr>
<tr>
<td>03/31/1999</td>
<td>dir216/U.M. Markets &amp; Society.doc</td>
</tr>
<tr>
<td>08/27/1999</td>
<td>dir270/Resume-Deb.doc</td>
</tr>
<tr>
<td>03/31/1999</td>
<td>dir270/Deb-Marymount Letter.doc</td>
</tr>
<tr>
<td>03/31/1999</td>
<td>dir270/Links App. Ltr..doc</td>
</tr>
<tr>
<td>08/27/1999</td>
<td>dir270/Resume=Marymount U..doc</td>
</tr>
<tr>
<td>03/31/1999</td>
<td>dir270/NCR App. Ltr..doc</td>
</tr>
<tr>
<td>03/31/1999</td>
<td>dir270/Admissions counselor, NCR.doc</td>
</tr>
<tr>
<td>08/27/1999</td>
<td>dir270/Resume, Deb.doc</td>
</tr>
<tr>
<td>03/31/1999</td>
<td>dir270/UMUC App. Ltr..doc</td>
</tr>
<tr>
<td>03/31/1999</td>
<td>dir270/Ed. Coordinator Ltr..doc</td>
</tr>
<tr>
<td>03/31/1999</td>
<td>dir270/American College ...doc</td>
</tr>
<tr>
<td>04/01/1999</td>
<td>dir270/Am. U. Admin. Dir..doc</td>
</tr>
<tr>
<td>04/05/1999</td>
<td>dir270/IR Unknown Lab.doc</td>
</tr>
<tr>
<td>04/06/1999</td>
<td>dir270/Admit Slip for Modernism.doc</td>
</tr>
<tr>
<td>04/07/1999</td>
<td>dir270/Your Honor.doc</td>
</tr>
</tbody>
</table>
Ultimately, I contacted 20 organizations between April 2003 and April 2005.
The leading cause of compromised privacy was betrayed trust.

Trust Failure: 5 cases

- Home computer; woman’s son took to “PC Recycle”
- Community college; no procedures in place
- Church in South Dakota; administrator “kind of crazy”
- Auto dealership; consultant sold drives he “upgraded”
- Home computer, financial records; same consultant

This specific failure wasn’t considered in [GS 03]; it was the most common failure.
Poor training or supervision was the second leading cause.

Trust Failure: 5 cases

Lack of Training: 3 cases

✔ California electronic manufacturer
✔ Supermarket credit-card processing terminal
✔ ATM machine from a Chicago bank

Alignment between the interface and the underlying representation would overcome this problem.
In two cases, the data custodians simply didn’t care.

Trust Failure: 5 cases
Lack of Training: 3 cases
Lack of Concern: 2 cases

✔ Bankrupt Internet software developer
✔ Layoffs at a computer magazine

Regulation on resellers might have prevented these cases.
In seven cases, no cause could be determined.

Trust Failure: 5 cases  
Lack of Training: 3 cases  
Lack of Concern: 2 cases  

Unknown Reason: 7 cases

✘ Bankrupt biotech startup  
✘ Another major electronics manufacturer  
✘ Primary school principal’s office  
✘ Mail order pharmacy  
✘ Major telecommunications provider  
✘ Minnesota food company  
✘ State Corporation Commission

Regulation might have helped here, too.
I have identified five distinct patterns for addressing the sanitization problem.

Visibility

- Users
- User Audit

Sanitization

- Users
- Explicit Item Delete
- Reset to Installation
- Complete Delete
- Delayed Unrecoverable Action

Document Files, Applications, and Media
Complete Delete: assure that deleting the visible representation deletes the hidden data as well.

Naming this pattern lets us discuss its absence in modern operating systems.
**Delayed Unrecoverable Action:** give the users a chance to change their minds.

[Norman 83] and [Cooper 99] both suggest this functionality, but they do not name or integrate it.
Two ways to delete information. #1: Explicit Item Delete

“Provide a means for deleting information where the information is displayed.”
Reset to Installation: Get rid of everything

Sanitization

Users

Explicit Item Delete

Reset to Installation

Complete Delete

Delayed Unrecoverable Action

Document Files, Applications, and Media

Reset/reinstall functionality is common (Windows; PalmOS; etc.). This pattern framework clarifies Reset’s security property.
User Audit: If the information is present, make it visible.

With files, this happens automatically when the Complete Delete pattern is implemented.
The power of these patterns is that they apply equally well to other sanitization problems.

- Document Files

- Web Browsers
Information is left in document files.

• The New York Times published a PDF file containing the names of Iranians who helped with the 1953 coup. [Young 00]

• US DoJ published a PDF file “diversity report” containing embarrassing redacted information. [Poulsen 03]

• SCO gave a Microsoft Word file to journalists that revealed its Linux legal strategy. [Shankland 04]
The information leaked because two patterns were not implemented.
Microsoft has tried to solve this problem with “Remove Hidden Data” tool.

RHD doesn’t integrate into the flow of document preparation. The patterns-based analysis predicts that RHD will fail in many cases.
Information is left behind in web browsers.

Two key problems: ① Deleted files; ② The cache
In fact, a lot of information is left behind in web browsers.

MIT Humanities Library, April 25, 2005
4 out of 4 computers inspected had significant quantities of personal email in their browser caches.

The American Library Association recommends software that automatically purges caches on a daily basis. [ALA 05] (It would be better to purge after each use.)
Applying the patterns, an obvious solution is to unify the history and cache:

The patterns make it easy to explain this concept to the browser developers—and users, too!
The patterns also suggest opportunities for further promoting HCI-SEC within the browser.

Without *Complete Delete* the data can still be recovered. This demonstrates the need for the complete pattern set.
1. Introduction to patterns.

2. Prior work in HCI-SEC.

3. Patterns for sanitization.

4. Patterns for secure messaging.
My thesis presents eight patterns for enhancing secure messaging.

- Leverage Existing Authentication
- Email Based Identification & Authorization
- Distinguish Internal Senders
- Create Keys When Needed
- Send Signed
- Track Recipients
- Track Keys
- Key Continuity Management

Users

Web-based Services
My thesis presents eight patterns for enhancing secure messaging.

I am going to discuss five of the patterns.
Secure Messaging — email that is *signed* and *sealed* — seems to be the grand challenge of usability and security.

- Public key cryptography was developed for secure messaging.

- This project is nearly thirty years old:
  - 1976 — Diffie-Hellman
  - 1977 — RSA
  - 1987 — RFC 989 (PEM)
  - 1991 — PGP Released
  - 1998 — S/MIME

- Today most people who engage in Internet mail have S/MIME-enabled clients, but there’s virtually no secure email.

Either it’s really hard to get this right, or nobody really cares.
People do care about email security. *(Garfinkel et al., FC05)*

In our study of 470 Amazon.com merchants:

- 59% thought that receipts from *online merchants* should be digitally signed.
- 47% thought receipts should be sealed.

And they have the tools — sort of.

- 54% could handle S/MIME-signed messages.
- 60% didn’t know if they could or not!
- 45% would upgrade their email client for more security.
Software for three public-key based communication security systems have been widely deployed.

<table>
<thead>
<tr>
<th></th>
<th>SSH</th>
<th>SSL</th>
<th>S/MIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secures</td>
<td>remote login</td>
<td>web pages</td>
<td>email</td>
</tr>
<tr>
<td>Protects Against</td>
<td>eavesdropping</td>
<td>eavesdropping</td>
<td>eavesdropping</td>
</tr>
<tr>
<td></td>
<td>spoof servers</td>
<td>spoof servers</td>
<td>spoof senders</td>
</tr>
<tr>
<td>3rd Party Certificates Needed</td>
<td>none</td>
<td>servers</td>
<td>sender &amp; recipients</td>
</tr>
<tr>
<td>Protection Mechanism</td>
<td>Warns when key changes</td>
<td>CA trustworthiness</td>
<td>CA trustworthiness</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Success</td>
<td>High</td>
<td>Somewhat</td>
<td>None</td>
</tr>
</tbody>
</table>

Success of these systems was inversely correlated with the need for third-party interactions.
Today’s S/MIME systems use third-party certificates to assert identity.
Signature-only S/MIME mail is automatically verified by most email clients.

Amazon.com is sending signed VAT invoices to its European merchants. No usability problems reported.
Signature-only S/MIME eliminates the burden on the recipient, but loses protection against eavesdropping.

<table>
<thead>
<tr>
<th></th>
<th>full S/MIME</th>
<th>signature-only S/MIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protects Against</td>
<td>eavesdropping spoof servers</td>
<td>spoof senders</td>
</tr>
<tr>
<td>3rd Party Certificates Needed</td>
<td>senders recipients</td>
<td>senders only</td>
</tr>
<tr>
<td>Protection Mechanism</td>
<td>CA trustworthiness</td>
<td>CA trustworthiness</td>
</tr>
<tr>
<td>Success</td>
<td>None</td>
<td>Good, when used</td>
</tr>
</tbody>
</table>
Signature-only S/MIME eliminates the burden on the recipient, but loses protection against eavesdropping.

Attacks that rely on spoofed senders:

- Worms that forge “From:” address
- Some kinds of spam
- Many “phishing” attacks

<table>
<thead>
<tr>
<th>Protects Against</th>
<th>full S/MIME</th>
<th>signature-only S/MIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>3rd Party Certificates Needed</td>
<td>eavesdropping spoof servers</td>
<td>spoof senders</td>
</tr>
<tr>
<td>Protection Mechanism</td>
<td>senders recipients</td>
<td>senders only</td>
</tr>
<tr>
<td>Success</td>
<td>CA trustworthiness</td>
<td>CA trustworthiness</td>
</tr>
<tr>
<td></td>
<td>None</td>
<td>Good, when used</td>
</tr>
</tbody>
</table>

Signature-only S/MIME protects against the security problems facing E-mail today.
This is the motivation behind the *Send Signed* and *Track Recipients* patterns.

Typical candidates for *Send Signed* are high-volume “do not reply” senders:

- EBay and PayPal notifications.
- Domain expiration notices.
- Advertisements.

Removing AOL and Webmail users, between 80% and 90% of Internet email users in our sample could decode S/MIME-signed messages.

[Garfinkel *et al.* 2005]

The technology for *Send Signed* is already deployed. Articulating this pattern will create the reality.
We can do even better by directly applying the SSH trust model to email:

<table>
<thead>
<tr>
<th></th>
<th>SSH</th>
<th>signature-only S/MIME</th>
<th>KCM S/MIME</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Secures</strong></td>
<td>remote login</td>
<td>email</td>
<td>email</td>
</tr>
<tr>
<td><strong>Protects Against</strong></td>
<td>eavesdropping spoof servers</td>
<td>spoof senders</td>
<td>eavesdropping spoof senders</td>
</tr>
<tr>
<td><strong>3rd Party Certificates Needed</strong></td>
<td>none</td>
<td>servers</td>
<td>none</td>
</tr>
<tr>
<td><strong>Protection Mechanism</strong></td>
<td>Warns when key changes</td>
<td>CA trustworthiness</td>
<td>Warns when key changes</td>
</tr>
<tr>
<td><strong>Success</strong></td>
<td>High</td>
<td>Somewhat</td>
<td>High in lab</td>
</tr>
</tbody>
</table>
Key Continuity Management is a strategy for managing untrusted certificates.

Traditional:

KCM:

KCM makes it possible to easily use S/MIME with self-signed certificates. (*Create Keys When Needed* pattern.)
Unfortunately, KCM creates a number of possible attacks:

Normal Communications
Unfortunately, KCM creates a number of possible attacks:

New Key Attack: (Forged From:, New Cert)
Unfortunately, KCM creates a number of possible attacks:

New Identity Attack (From Hotmail, New Cert)
Unfortunately, KCM creates a number of possible attacks:

3. Unsigned Message Attack

Unsigned Message Attack (Forged From:, No Cert)
Unfortunately, KCM creates a number of possible attacks:

1. New Key Attack
   - Key 42214
   - Maria Page
   - mpage@campaign

2. New Identity Attack
   - Key 123456
   - mpage@campaign

3. Unsigned Message Attack
   - Key 123456
   - mpage@hotmail.com

Can untrained end-users resist these attacks?
The Johnny 2 Experiment:

Designed to test KCM model:

- Subject plays the role of a political campaign worker.
- Enemy campaign tries to steal documents through a spoofing attack.
- Three attack messages.

Experimental Details:

- 43 subjects aged 18–63 ($\bar{x} = 33, \sigma = 14.2$)
- 19 Men, 24 Women
- 17 to 57 minutes ($\bar{t} = 41, \sigma = 10.32$)

Earn $20 and help make computer security better!

I need people to help me test a computer security program to see how easy it is to use. The test takes about 1 hour, and should be fun to do.

If you are interested and you know how to use email (no knowledge of computer security required), then call Simson at 617-876-6111 or email simsong@mit.edu
The Johnny 2 Results:

We compared KCM with no KCM and found:

<table>
<thead>
<tr>
<th>Attack</th>
<th>Attack Rate Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Key Attack</td>
<td>81% drop***</td>
</tr>
<tr>
<td>New Identity Attack</td>
<td>43% drop**</td>
</tr>
<tr>
<td>Unsigned Message Attack</td>
<td>24% drop</td>
</tr>
</tbody>
</table>

\[ ***p < .001; **p < .05 \]
The KCM patterns can increase mail security by promoting the use signing and sealing.

KCM clients must:
- Create keys when needed.
- Track capabilities of correspondents.
- Maintain database of correspondents and certificates.
This talk has presented a few of my original contributions. Here is the complete list:

On Sanitization:

- Novel hypothesis for the HCI-SEC conflict
- Comprehensive literature review and critique
- Analysis of 236 hard drives
- Traceback study of 20 organizations
- Cross-drive forensics
- Study of operating systems sanitization issues
- Study of web browser sanitization issues
- Study of Word and Acrobat sanitization issues

On Regulatory techniques:

- A “Bill of Rights” for RFID labeling.
- A proposal for software labeling.
- A novel analysis of how ANSI Z535.4-2002 could be applied to software.

On PKI and secure messaging:

- Survey of 470 Amazon.com merchants
- Technique for embedding invisible digital signatures in MIME messages
- Application of Key Continuity Management model to email
- User study of KCM with Outlook Express
- A meta-analysis of the E-Soft SecuritySpace study.

On HCI-SEC Patterns:

- Four original principles and more than 20 original patterns for aligning security and usability.
- An analysis showing why inconsistent vocabulary in the field of security damages usability.
In Summary

✔ Patterns are a promising technique for aligning security and usability.

✔ Sanitization can be made automatic and natural in many cases.

✔ Significant progress can be made on mail security with technology that is already deployed.
Acknowledgments

Thesis supervisors:
Robert Miller and David Clark

Thesis readers:
Ronald Rivest and Daniel Weitzner

Collaborators:
Abhi Shelat, Ben Gelb, Erik Nordlander

Presentation Critics:
Karen Sollins, Sian Gramates

Slide format:
Michael Alley, Virginia Tech

And special thanks to Beth Rosenberg, Sonia, Jared and Draken...
In Summary

✔ Patterns are a promising technique for aligning security and usability.

✔ Sanitization can be made automatic and natural in many cases.

✔ Significant progress can be made on mail security with technology that is already deployed.

Questions?