User-Centered Security: Stepping Up to the Grand Challenge

Mary Ellen Zurko
IBM Software Group
mzurko@ibm.us.com

Abstract

User-centered security has been identified as a grand challenge in information security and assurance. It is on the brink of becoming an established subdomain of both security and human/computer interface (HCI) research, and an influence on the product development lifecycle. Both security and HCI rely on the reality of interactions with users to prove the utility and validity of their work.

As practitioners and researchers in those areas, we still face major issues when applying even the most foundational tools used in either of these fields across both of them. This essay discusses the systemic roadblocks at the social, technical, and pragmatic levels that user-centered security must overcome to make substantial breakthroughs. Expert evaluation and user testing are producing effective usable security today. Principles such as safe staging, enumerating usability failure risks, integrated security, transparent security and reliance on trustworthy authorities can also form the basis of improved systems.

1. The Problem of User-Centered Security

The importance and challenge of the relationship between human users and security mechanisms has been recognized since the dawn of time in the systems security field. Saltzer and Schroeder [43] defined the principle of psychological acceptability in their seminal 1975 paper on the protection of information in computer systems.

> "It is essential that the human interface be designed for ease of use, so that users routinely and automatically apply the protection mechanisms correctly. Also, to the extent that the user's mental image of his protection goals matches the mechanisms he must use, mistakes will be minimized. If he must translate his image of his protection needs into a radically different specification language, he will make errors."

The mode of interaction with security mechanisms was users applying them consciously and directly as standalone tools in a context they understood. The challenge was to make the security model of the tools consistent with the user's mental model of security, so that undesirable errors would be minimized.

By 1996, humans' relationships to computers had changed dramatically. The World Wide Web, invented in 1989, was popularized with a GUI in 1992, and began its steady rise to ubiquity. The more diverse, distributed, and popular uses of the web, the network, and computers became, the more obvious it became that problems with the usability of existing security mechanisms would compromise their effectiveness. Simon and I [58] defined the term **user-centered security** to refer to "security models, mechanisms, systems, and software that have usability as a primary motivation or goal." We foresaw the following three categories of solutions: (1) applying human-computer interaction (HCI) design and testing techniques to secure systems, (2) providing security mechanisms and models for human collaboration software, and (3) designing security features directly desired by users for their immediate and obvious assurances (for example, signatures). Security researchers pursued the usability in some of the most important and intractable areas, including trust models, encryption and signing, and authentication. HCI researchers began to attack the same problems. Sometimes these even talked to each other.

Two years ago, in November 2003, Computing Research Association held a conference on "Grand Challenges in Information Security & Assurance" [10]. One of the four resulting grand challenges was:

> "Give end-users security controls they can understand and privacy they can control for the dynamic, pervasive computing environments of the future."

In the 28 years since psychological acceptability was defined, the problem has increased in urgency.

While there has been substantial work in usable security in the last nine years, the CRA's grand challenge indicates that the problem is not only

unsolved, but has become more pressing. Our personal and social processes and interactions rely more and more heavily on computers, communications, and applications. The world and the information that feeds it are getting more connected, more available, and more accessible, and all at an increasingly rapid rate. These changes provide the value to people and to society, and cause the difficulties with securing that value.

This essay provides an overview of the user-centered security challenges behind the grand challenge and a discussion of the tools and approaches that have progressed the furthest and hold the most promise for near term results. It attempts to answer the following question: why have we lost ground in usable security since 1975? Throughout this essay I also highlight areas where further research is needed.

2. Opportunities in User-Centered Security

> There is no such thing as problems, there are only opportunities
>
> My boss at Prime Computer, circa 1986

The largest roadblocks to providing user-centered security break down into three categories; (1) human and social relationships to usable security, (2) technical challenges best attacked with research, and (3) further difficulties with implementation and deployment.

2.1. Human and Social Relationship to Security

There is much about the human and social relationship to computer security that we still do not sufficiently understand. What is the best we can hope for when we ask humans to understand a quality of the system so complex that it cannot be understood by any single architect, developer, or administrator? Since humans are part of the system and the system's security, how much responsibility should be assigned to them? Since usable security is so obviously a universally desirable attribute, why aren't we applying resources to it commensurate with its desirability?

2.1.1. Understanding vs. Effectively Using Security Controls.

> If we go on explaining, we shall cease to understand one another.
>
> Talleyrand

What are people's relationship to computer security, as individuals, as a group, as an organization, and as a society? Technology thinkers who understand technical complexity see usable security being enabled by security mechanisms that end users can understand. As computer systems get more complex, it is unfortunate that the security of those systems has also been getting more complex. For example, interpreters that enable active content attacks exist in simple print and display programs. They are at the core of the web technology that forms the basis of most people's interactions with computers today. How can users ever understand anything that complex?

Emphasizing understanding can produce profound changes in the creation and design of security mechanisms, when making them understandable is a primary design goal. This idea is at the heart of the reference monitor concept. Security mechanisms that cannot be understood cannot be effective. Making security features explicable to an imagined, presumed, or tested representative user extends this traditional security design goal.

Attempting to explain what the security expert, architect, designer, or developer understands about a mechanism can be useful. Transparency of security mechanism and their guarantees is at the heart of evaluation and accreditation efforts such as Common Criteria [9]. Evaluation by external experts provides a bridge between the expert understanding and the needs of users. Evaluations enable informed comparisons in those cases where the description language is both consistent and coherent.

Clearly explaining and documenting security mechanisms and their use can produce more usable security, both by communicating what is known, and by providing critical feedback on the degree of explicability of the mechanisms. Security mechanisms that are explicit but incomprehensible and not integrated with the task do not help [61, 20], and the act of documenting them can highlight this problem.

Graphical user interfaces are often meant to be self-documenting. Visualizing security is one method for helping users understand security. Making security information visually available in the context of the task enables users to make the right call, though it does not necessarily give them guidance on determining the right call [12, 13, 14]. Privacy awareness [7] is another form of this approach. In Privacy Bird [11], users liked having the ability to get high level privacy related information at a glance.

Given the richness and complexity of the security currently needed by our systems, it may be that we will never have enough space to explain or visualize (or audio-ize) everything about the security mechanisms that users should understand. For example, the Johnny2 CoPilot study [16] was built around an interface and mechanism specifically designed to make mail encryption and signing understandable. The researchers found that users did not know that digital signatures prevented content modification. While this aspect of the technology could be explained or visualized as well

(for example with some sort of border or other graphic), the mechanisms, their benefits, and their limitations, pile up fast in even a simple scenario. If the mail message could include active content (for example Java or Javascript) which is also a security concern, the complexity of what should be understood has at that point probably exceeded the bounds or interests of most users. The majority of users are unlikely to desire a better understanding of security mechanisms than they currently have. From a panel at Network Distributed System Security Symposium 1999 on "Security and the User" [50] to discussions with very security knowledgeable customers, what I hear from users is "Why can't security just work?"

People do understand something about security controls in the physical world. The understanding usually centers on the threats that are repelled by them. Locks on the house and the car keep burglars out. Strongboxes inside banks and houses make it harder to get to the most valuable objects, and slow down the intruder even more. Hackers and viruses are in the news; they disrupt systems and corrupt data and steal identity information for financial fraud. The intrusions, attacks, hacks, and other incidents are how most users think about security mechanisms. Will it keep my identity safe? Will it keep viruses off my computer? This approach is in direct opposition to how security mechanisms are designed. Security mechanisms are designed to withstand both current and potential future unknown attacks. It is the immediate use and utility of the security mechanism that makes sense to users, not their inner workings.

[15] lists the risk management questions that users ask:

- What could go wrong?
- How likely is it, and what damage would it cause to me or to others if it did?
- How would I know if something went wrong?
- What reason do I have to believe that it won't?
- Who is responsible to ensure that it doesn't, and what recourse do I have if it does?

Humans know that the likelihood and sophistication of an attack may depend on the (perceived) abilities and protections of the person under attack (particularly if social engineering or scamming is involved), the abilities of the attacker (and their tools), and the (perceived) value of the item under attack. They will trade off the short term and long term benefits applying security (or not). A UK news outlet traded candy bars for (ostensible) passwords with commuters [42]. A toy lock box for ages 6 and up has keys and a combination [31]. It shows that people can understand how to use a security control at an early age. Greenwald [21] suggests that primate dominance games that humans engage in at a very early age may show that territorial "security" is hard wired into our brains.

Users need to understand how to **use** the security controls that are directly relevant to their task and context. The desired end result of usable security is that security controls are applied appropriately and effectively to provide protection. The risks of using the features they protect are thus decreased, and are not exacerbated by the use of and belief in mechanisms that are not likely to withstand the most common or risky attacks. The goal is that security controls be effectively used.

I would rephrase the CRA's grand challenge stated above to be:

> "Give all users (including developers, administrators, and end-users) security controls that protect them, their systems, and their privacy, that they can use appropriately in the dynamic, pervasive computing environments of the present and the future."

2.1.2. User Slip-ups Are Not User Errors.

> I didn't do it.
>
> Bart Simpson, cartoon character

When a security breach is said to be caused by "user error", the desired implication is that the breach was not the responsibility of the (computer) system, but of the user. Acceptance of that implication is one of the roadblocks to the grand challenge of usable security. For products that are deployed in both the enterprise and consumer spaces, the community of security experts and society at large should never accept "user error" as a source of a security problem. If a non-malicious, mistaken end user is blamed for a vulnerability or breach, we have to ask, why did the system make the insecure option so easy and attractive? If the error was "skill-based" [39], an automatic and unconscious slip-up, then basic usability techniques should be brought to bear. These will minimize the potential of serious breaches arising from the equivalent of a typographical error. For example, the Therac-25 error [29] which killed several people, was due to a bad editor. If the "user error" is a conscious action (or lack of a conscious action) that was mistaken, then the design problem that needs to be fixed runs deeper.

Tog [49] points out that a security breach is the fault of the security designer (assuming a single product and the actual existence of a security designer). Security professionals, like the stereotype of legal professionals, may run on the first principle of ensuring that nothing bad happens that can be attributed to their area of responsibility. A big worry for any product's security architect is, "What is the likelihood that our product can and will be exploited in a way that makes it to the

cover of the New York Times?" Sometimes product designers and architects believe that if they design the system so that the hard security choices are the responsibility of a user or customer, their company can say to any related breaches a customer might suffer, "I didn't do it".

Difficult-to-use security controls in one place in the system encourage poor security decisions in other parts of the system. Bill Cheswick [8] points out that most systems overuse the setuid to root function which gives a program the privileges of an administrator. I agree with Greenwald [22] that the cause is designing a service that does not require that function is difficult. The system's protections are not easy for the majority of developers to use appropriately (and developers are people too). Initial usability testing of an early enterprise web conferencing system showed that users were immediately confused by the browser's ActiveX trust dialogs. They could not get to the desired functionality because they did not understand they could take the more sophisticated approach that today's users do to such dialogs (click OK [61]). The most effective workaround available was for the product to explain to the user the infrastructure security model that they found confusing. The initialization screen of the product explains the user's choices when faced with a security decisions and the results of those choices. Similarly, in early deployments of multilevel systems, users regularly declassified documents to the lowest level possible. Providing reasonable and usable use cases of security functionality needs to become accepted before system interactions like these will diminish.

Figure 1: IBM Sametime® initialization screen

An area that signals the strong possibility of "user error" is any security procedure that includes a step that is too vague to be precisely documented, even as an example. For example, users are often sent "out of band" to resolve a security or trust question. Brustoloni's work [Brustoloni] shows one way to approach that challenge; ensure that out of band contact information is available in band. Assuming a secured collaboration infrastructure, contact information can be extended to computer based methods of real time communication, including IM or VoIP.

Another architectural area that attracts "user errors" is error cases. Every error message a user sees should be understandable and actionable, but often they are not, particularly in the security area. Consumer operating system error messages will tell users how to increase their paging file size, showing that useful error messages for complex system problems are possible. Messages telling the operating system user how to contact their system administrator are not useful to consumers, but are to users in an enterprise with accessible system administrators. Many security errors and warnings leave users wondering what the problem means and what they should do. Such warnings are really only a defense against blame, not an enhancement to security.

The various warnings about SSL server certificates are a case in point. Xia and Brustoloni's [55] work attempts to make every error message both understandable and actionable. In general, the SSL dialogs he suggests are oriented towards https and the public web. SSL can be used for other protocols (IIOP, SIP) and for enterprise intranet activities. In addition, there are error cases that SSL can encounter that are not covered by this work. If the server certificate is not trusted or the DN does not match the host address, then the potential vulnerability is explicable to the user. But what does it mean if the validity dates of the certificate have not arrived yet? Why should the user care about that? More work like Brustoloni's on handling **security error cases** is needed.

2.1.3. Marketing Usable Security.

Sell when you can: you are not for all markets.
As You Like It, Act 3, scene v

Usable security is obviously a desirable quality in commercial software. Enterprise customers explicitly request software that can be deployed securely with a low Total Cost of Ownership (TCO), which equates directly to the usability of the security. This market pull should increase the technology transfer of user-

centered security into products, or increase the rate of innovation in usable security in product development. The market does not seem to have done so, beyond a small number of companies, including those specializing in security products that are able to create market value from emphasizing usable security [5].

When it comes to allocating resources in product development, everything is a cost/benefit tradeoff. For most software products, little attention is paid to usable security until a substantial exploit can be attributed to the lack of usable security, or potentially solved by more of it. For example, most mail is not signed using S/MIME, and could have a forged sender. The difficulties with deploying and understanding most S/MIME implementations and the vulnerability that left did not draw much attention until the use of spam (including ad-spam, scam-spam, and attack-spam) became a substantial problem with email reliance. As we have seen, recognition of exploits is how most users engage with security. Thus exploits increase the ability to justify the resource allocation to usable security when the tradeoff of resources did not seem justified before the widely recognized breaches. In this model, economic roadblocks can be overcome by concrete and visible exploits, stronger explicit customer demand, or decreasing the cost of user-centered security. Let us look at each of these economic drivers in turn.

Practically anyone who has ever worked on the security of a shipping product knows how security vulnerabilities are dealt with by her organization. Organizations that explicitly track them will also triage them, dedicating resources to fixing the worst vulnerabilities first. The list of vulnerabilities comes from internal developers and users, internal (or internally contracted) testers, ethical hackers and advisories, and external sources (customers, advisory organizations, ethical hackers, not-so-ethical hackers). The not-so-ethical hackers stand out from a process point of view because they do not work with the organization to identify and triage the vulnerability. They exploit it, advertise it or sell it for personal reasons.

Once a vulnerability has been exploited or advertised, the resources devoted to fixing it increase. Resources are also dedicated to responding to the exploit. If an organization does not have an internal process for triaging and fixing vulnerabilities, the overall quality of the security in the code base is likely to increase. If they do have such a process, it is likely that the exploit causes resources to be pulled away from vulnerabilities that by objective measures are worse. In either case, resources are pulled away from other security-related activities. From a systemic point of view, exploiting or advertising vulnerabilities is not the most effective way to increase the security quality of our products. A **transparent security quality process** is [30]. That process should include HCI as well as assurance aspects.

More proactively, there are ways to increase the market demand for usable protection before an exploit highlights the gap. Persuasion techniques [53] can be used to make the threats and the risks of lack of usable security clear. These techniques include social marketing, which associates positive qualities (professionalism, loyalty) with the desired behavior (security-aware purchasing) and negative qualities with the lack. Another persuasion technique equates safety with being a less attractive target (through the use of more obviously secure software). The more extreme social persuasion techniques equate fear, uncertainty, and doubt with security oblivious behaviors.

Marketing campaigns can generate positive consumer awareness and pull for previously obscure attributes ("Brown eggs are local eggs, and local eggs are fresh."). Quantifying security risk [6] through insurance, certification, or other means enables more accurate and explicit cost/benefit tradeoffs and can provide a factual basis to marketing efforts that emphasize the desirability of usable security. Insurance can reduce real risk and enhance the feeling of safety that usable security should provide. Since people understand security best as protection against risks and exploits, consumers need to be told explicitly what they can be protected from. Usable security mechanisms must be designed to provide those protections. Checklists are a particularly attractive method for purchasers to compare products and product coverage in various areas. The human emphasis on risks indicates that checklists should be based on or categorized by exposure types. Existing documents that could form the basis of such checklists, such as Common Criteria [cc] and ISO 17799 [28] use complex and dense language, and do not use threats to structure their recommendations.

Low cost techniques that yield useful results and tools that automate the simple tasks are two ways to make usable security cheaper. In the area of usability, Jared Spool [48] pioneered low-cost evaluation methods as a way for many more software projects to incorporate usability testing appropriately. Tool kits such as Visual C++ generated consistent user interfaces, setting a bar on certain types of usability. Usable security techniques need to be reduced to methods that are simple enough for most developers to execute effectively, and turned into checklists and tools. The checklists should not be philosophy or vision statements. They must have specific design criteria that can be actively evaluated against concrete functions and design elements in a system.

2.2. Technology's Relationship to User-Centered Security

Much of the existing literature focuses on challenges to user-centered security that require breakthroughs in our approach to the technology. I focus on three. How can we incorporate models of user behavior into models of security, so that real user behavior is taken into account? How do we design systems so that security related decisions and actions are minimized, and always made by the person who has the ability to make them? How do we design systems so that all the parts that determine the user's ability to interact with them securely are actually secured?

2.2.1. Users As Part Of The System.

> You're either part of the solution or part of the problem.
>
> Eldridge Cleaver

Classic security models [17] situate the end user outside of the system boundary (with the administrator inside). They provide mechanisms for very attentive and obedient users to behave securely. For example, security critical operations can only be invoked through a trusted path. While almost any computer user is likely to know when they must use ctrl-alt-del, whether or not they know what security the use of that key sequence provides them is an open question.

Classic security models also acknowledge that computers systems cannot prevent users from giving away information they have. They ignore the fact that computer system interfaces can make mistaken security breaches more or less likely. Some of the most difficult and worrying current attacks rely on social engineering, attacking the human processor, to either extract something directly from the human (i.e. spam and phishing) or to use the human to overcome the technical barriers to the attack proceeding on the computer processor (i.e. virus propagation). These attacks are akin Schneier's "semantic attacks" [46]. Ignoring the user's active participation in the security model enables attacks on the user through the computer system, and makes the interface to the user a weak link. Computer systems can be used to fool users into giving away what they do not want to.

Security models can include users' beliefs and knowledge in terms of protocol states or secrets, thought they are often encoded in the user agent, not the user's brain. Using existing modeling capabilities, a user-centered security modeling technique would be to structure the security of the system and its data so that what users could easily and mistakenly give away would not compromise them or the system alone. Security techniques implementing this generally put of additional barriers to user actions, including two factor authentication and two person control. This approach has its own usability challenges.

Existing work on user models does not map well to existing security models. Specific human capabilities such as memory or error behavior have models. There is extensive literature on human trust and applying it to computer systems. Targeted models of password security and usability are based on very specific aspects of a very limited task [40].

We do not have an appropriate approach to **modeling human security behavior abstractly**. In a sense, security modeling is an abstract concern while human use is driven by pragmatic details. Such models may need to abstract the details that matter to humans. Existing user models are driven by concrete tasks and interfaces. In that context, the user's knowledge and actions can be modeled in a process where the modeler thinks like the user [5]. Another approach is to drive the entire security model from the user's model. [57] shows an active content security model that is centered around user actions and intentions. Threat based models are the class of security models that map most closely to what we can models about users today. They can be extended to include the **risks of unusable security**.

The vast majority of users do not interact with computers in isolation. User-centered security models will need to take into account relationships between system users, including **authorities** and **communities of users**. In multi-user, distributed, and collaborative systems, what the user population "knows" can be leveraged for protective purposes. Human authorities can set policy. Established relationships within and between organizations and communities legitimately form the basis for trust. Conversely, there may be information that needs to be hidden from other members of a community of system users. **Deception**, plausible deniability, and ambiguous information are part of the model for applications that are designed to share very personal information, such as social location disclosure applications [27]. Early work in multilevel databases [54] recognized the need for cover stories or cover information in places where users who should not see some information would expect some.

2.2.2. Who Makes The Security Decisions.

> What, me worry?
>
> Alfred E. Neuman, Mad Magazine

Security problems can come from bugs and flaws in the design and implementation of the system software, firmware, or hardware, unanticipated use of the system for attacks (on either the computer processor or the human processor), and mismatches between computer activities and human expectations. In the latter case, the mismatch may occur when the user is explicitly

given a security decision to make. Making a security decision correctly is not easy. One of the most frustrating and difficult things in security, and one of the most desired, is detecting an intrusion attempt accurately [32]. Even determining after the fact that a breach occurred is difficult.

Some activities have a high likelihood of being a security problem. Firewalls repel the many casual and not-so-casual attempts to break into systems on the Internet, and can catch malicious code trying to contact the Internet from the host machine. Many incorrect restrictive security decisions can be recovered from, with more or less ease. The problem of incorrect denial of access is dealt with by several approaches. Training wheels on access control mechanisms teaches the system the current access patterns before actively enforcing the policy. More commonly, the person desiring access notifies the owner/manager (assuming that person can be identified). Optimistic access control [38] allows new access and lets the organization impose penalties after the fact for privileges that were abused. Recovering from incorrect security decisions around active content may be the most difficult challenge. Disabled Java or Javascript in web forms or collaborative applications can cause business logic to break in opaque and inscrutable ways. Mail from someone I have never communicated with before that was blocked may be scam-spam or may carry a virus, or might just be a co-worker I have never met who I now have business with. Security research and technology makes strides against all of these, but the recovery process from a wrong decision, either too restrictive or too permissive, puts a human in the loop.

A range of user roles are responsible for security decisions. The developer, the administrator, and the end user all have different views, information, knowledge, and context. In many cases, none of them knows whether or not an actual security problem exists. The lines of communication from one role to the next are mostly unidirectional. Each role uses whatever context or information the technology carries to determine what hints the previous roles might be sending them. Documentation and education rarely fill that gap. Developers create software to protect and detect, with points of variability to allow for differing configurations, policies, and tradeoffs. The default values of the configuration options determine their initial assumptions. Administrators can change those options, and determine default policy for the end users. And the end user can (in many cases) override policy with personal preferences and specific actions.

Most end users will not want to override defaults and policy, since it represents the received wisdom from the theoretically more knowledgeable authorities. Usable security research shows that the majority of users will neither take the time to configure their settings properly (even when told directly how to do so) nor be able to process security interrupts that disrupt their task at hand [61, 52]. Security problems involving deep technical detail, which is often the case with active content attacks, are not something about which most users can provide an informed response. The option to alter security settings by the end user is still important, not only when specific incidents give additional insight to the end user, but to satisfy power users, group thought leaders, and evaluators (in the popular press, for an enterprise, and for specific criteria). However, it is not effective as the primary means of defense.

Providing a security model such as code signing is not enough when the model does not enable usefully secure default policies or understandable choices for the user roles. User decisions, when they are imposed or required, must be structured around a model the user can understand. If the user is asked to trust a signing entity (for executing code or for receiving SSL protected communications, for example), the user has to have some model of who's being trusted for what. The developer must provide that model and the developer and administrator must provide reasonable defaults so that active decision making is not a requirement for daily operation.

Constraints beyond those provided by the "pure" security mechanisms can be useful in making security decisions understandable. Within an organization or enterprise, recovery options are available that do not apply at the individual or consumer level, through reliance on administration and service groups. The Notes PKI ties certificate distribution and trust to the enterprise context and naming scheme for individuals and organizations [62]. As in [18], trust follows the name hierarchy. This structure provides a natural set of trust defaults, and limits the damage an untrustworthy authority can do. Physical security is being leveraged in innovative ways by research in portable devices and wireless connectivity. Physical gestures to a trustworthy CA [3] provide a natural and secure way to specify trust. Users bring their mobile devices into visual range of a CA, and use the device to point to the CA to tell it to trust that authority. There are obvious limits to the scalability of that approach. "Think locally, act locally" can most easily be accomplished within small structures. Useful constraints may also be derivable from the "upper layers" of specific applications and their use of the infrastructure.

2.2.3. Assurance For The User.

> But yet I'll make assurance double sure
>
> Macbeth, Act IV, scene i

The user's special knowledge of security comes in part from their ability to look at the specific interaction

at hand in the context of how it relates to the entire system they're working with. This broad and specific view is in tension with the componentized assurance view that states that the security surface of the system must be minimized to ensure that it is accurate and bug free [2]. Developers or other experts evaluating the system determine its accuracy and assurance. Intrusion detection systems have a challenge similar to that of users, since they can take input from a wide array of sources across the system. The trustworthiness of the sources can vary. The logic of determining whether or not an intrusion has occurred is kept compact, sophisticated and explicit. User processing will not necessarily reflect any of those attributes.

Studies on trust indicate that users decide whether to trust a system based on all the information immediately available to them [37]. This includes non-security aspects that might reflect the trustworthiness of humans associated with a service, such as how professional a web site design is. Information on past history, like eBay's reputation service, can also promote trust. When considered from a classic assurance perspective, many of the things people rely on to determine trust can easily be manipulated orthogonally from how truly trustworthy a web site is. They are traditional indicators of social trustworthiness, applied to social situations where computers are used to mediate.

While users may rely on components not meant to provide security, the flip side can occur where there are components in a platform providing useful security functions (encryption, signing, trust root storage) but not actually securing the system in any meaningful fashion. This can happen in standards and other efforts that begin by specifying the security subsystems that are needed before determining how the subsystems secure the overall system. Current examples are frameworks such as OSGi and Eclipse [23]. Security, like other qualities such as usability and performance, is a system-wide concern, requiring system wide thinking to be effective.

How can we integrate human assurance with classic security assurance? Making all user visible aspects part of the security kernel increases the complexity of the part of the system that should be given the most rigorous attention. That approach does accurately apply security assurance techniques to the interfaces upon which security relevant decisions will be made. Firefox's approach [41] to simplifying the security surface for users and minimizing false end user alerts is to rely on rapid, timely updates to the code base in response to future attacks (much as virus protection does today). We need more work on security assurance for the end user, including mechanisms that can take hints based on data not part of the core security processing functions.

2.3. Implementation and Deployment Are The Golden Spike

Much of the information in the current literature focuses on how user-centered security is or can be achieved using specific designs or technologies. There are not yet tools or best practices that allow a larger body of practitioners or researchers to incorporate user centered security into their system. There are no criteria or checklists for evaluating how usably secure a system or approach is likely to be.

HCI made great strides as a discipline through the promotion of guidelines, tool kits, and processes for incorporating usability into products at a reasonable cost by anyone willing to take the time to learn to use them. Security assurance has not made similar advances in consumability. Security assurance comes from use of algorithms and techniques that have been shown to provide security in practice, have proven to be secure (in the formal sense), or that are backed by process and assurances that demonstrate their strength.

How can we integrate the lessons from practice into our research thinking so that we achieve usable security in practice? And how can we specify and implement reusable security components that support a user-centered security model in the system they're integrated into?

2.3.1. Integrating research and practice.

> In theory, there is no difference between theory and practice. In practice, there is.
>
> Yogi Berra

The establishment of best practices relies on some history of feedback loops between research, development, deployment, and use. There is some communication built into the system from research to development. In commercial software companies, it is often called "technology transfer." Development feeds naturally into deployment, which feeds naturally into use.

Communication up this chain is rarer. As [34] points out, the security weaknesses of text passwords were revealed only by their use in practice. Those weaknesses are so well established that they have been communicated back to research, sparking solid work on looking for suitable alternatives. Changing practice could also have changed the degree of usable security provided by passwords. Generated passwords were a reasonable solution 20 years ago when only professionals needed them and they only had one password, and it was only used at a computer in an office with a lock on the door. Now users deal with many passwords with many different but overlapping strength and management policies, rendering almost all forms of deployed passwords unusably insecure. Users

cannot create and recall many difficult passwords for a multitude of systems. They will reuse them and write them down, both of which can be exposures.

[5] recommends that products know their audience, and that responsible parties interact directly with customers and users. This is classic business advice; there is no substitute for understanding the customer and their goals and business. It provides feedback from deployment and use to development. Many HCI techniques provide feedback from use to other stages. User advocates can provide related information. If a technical writer cannot explain how to use a security mechanism in a practical and safe fashion, it's a safe bet that users won't know how to do so.

Tradeoffs that are critical in practice must inform research if research is to successfully transfer to practice and products. Some of these tradeoffs are surprisingly mundane. For example, in product development, screen space is often at a premium, with many important pieces of information vying for a place in the sun. CoPilot [16] takes a substantial amount of primary screen real estate to get its point across; more than would be likely to be allotted to security in a general purpose email product. This squeeze is not encountered in special purpose dialogs [62]. To advance usable security, research needs to actively seek development, deployment, and use experience, and development needs to actively seek deployment and use experience.

2.3.2. Components Contributing To Usable Security.

> With these kinds of proposals, the devil is in the details.
>
> John B. Larson

Reuse of security component allows concentration on the assurance of the algorithms and the code. It supports centralization of security concerns, which makes it possible to simplify security mechanisms, potentially increasing both assurance and understandability [60]. Standards provide another form of reuse, in algorithms, mechanisms, and APIs. They make security usable by developers; they can use the security mechanisms developed by others instead of inventing them. Abstraction of the security specific functions allows mechanisms such as authentication credentials to change over time when the infrastructure changes, without disrupting the rest of the system.

Both reuse and abstraction pose a challenge to usable security. Security is often most obvious to the user when things go wrong, when an error or alert is raised by a security mechanism. Very directed users will even ignore alerts, until they are unable to accomplish their task, because of either the security problem or its solution. Exceptions or errors from security components are often either very low level or abstracted. Low level error messages are likely to require more detailed knowledge to understand than most users possess. Abstracted error messages remove the security situation from the specific context the user is in, stripping them of useful clues.

The security use of protocols can also change when their use changes [36]. Repurposing a security mechanism, such as a protocol, can change the security properties because of the changed context and threats. For example, we considered in some detail the usability of using SSL as provided by JSSE to protect rich client protocols such as IIOP, HTTP, and SIP used to access backend servers. In the browser, the URL defines the desired target, and the protocol action (get or submit) maps directly to a user action. When a form is submitted by a button press, users can get confused about what protections are available. There have been several reports of web forms which were themselves SSL protected mistakenly submitting data in an unprotected URL.

The connection between user action and protocol activity is even more opaque in a rich client. The client programs actions may be any one of a number of housekeeping operations, including initial access of data for a particular application, synchronization of data between local and remote stores, or getting server updates of code or metadata. If there is a problem with some SSL server's certificate, the user will want to know what server the network layer was connecting to, for what purpose, and with what data. That data is not readily available to callbacks that process exceptions and errors.

[55] encapsulated several likely browser-based scenarios in his recommendations on enhancing the usability of the SSL security dialogs. They develop the general principle of some sessions being more sensitive than others, and allowing the user to trust the certificate for just a session. This approach can apply to rich client use as well, though the definition of a session is dependant on the structure of the calling application. They stop short of addressing the question of how sensible the security model is. For example, just what threat should the user consider if the certificate's validity time period has not yet begun?

Rich clients can provide additional tools that when used will eliminate the occurrences of some security errors. These include tools to notify administrators when certificates are going to expire and to easily update trust roots across the client base.

In general, no error message or exception should be specified or implemented in a security component unless it is linked with an action the user can take when they receive it, or an action that an administrator or user can be told to take to ensure the error does not occur. Security modules that can be reused need to

consider what information will be needed to process these errors or alerts, should they reach the user.

3. Effective User-Centered Security Today

Despite the many substantial challenges, there is a solid body of work on user-centered security that we can use today. Most of the work addresses technology's relationship to user-centered security, but some address the human and social aspects, and some covers implementation and use. References to existing work may be found in the bibliography of this essay, and on the HCISEC reference list [24]. There is now a symposium dedicated to usable security and privacy [47], a newly published book security and usability [25], and an email community on the topic [26].

The two best tools we have mastered so far are applying certain HCI techniques to security mechanisms, and distilling and applying some principles of usably secured systems established by work so far. Each of these is discussed in more detail below.

There are many other areas that can yield results that are less well explored. Much work in the area of useable security gives process advice, or shows how experts in the area can apply usable security to a specific problem or domain. In the former case, the process advice is almost always to security people, and is often some variation of "Think about the user" [49] or "Use established CHI methods and principles" [44, 40, 56, 34]. Specific problem or domain advice is mostly in the area of authentication and passwords.

3.1. Human Computer Interface Techniques and User-Centered Security

Expert evaluations of both usability and security can enhance those attributes. Both disciplines have tools and processes to provide input from experts, and both support a small industry of consultants (for example, Nielsen Norman Group, @stake). Expert reviews of the usability of security have mixed success [62]. UI experts can help with the usability of every day concepts (such as passwords) and visual design. They may not be able to give deep advice on the usability of security aspects that are only invoked when a problem or error occurs, or which surface existing security mechanisms in their current complexity. Individuals with expertise in both security and usability can provide richer advice, but are far scarcer. Because they don't yet have processes or checklists these evaluations can be very inconsistent, and cannot be done on even a simple scale by others.

There are other mismatches with applying existing usability techniques to security. Many usability techniques center on the user's goal or task [4, 45]. For the vast majority of users impacted by security, it's an attribute of their tools, not their goal. Many security goals can only be stated anti-goals; someone guesses your password, business critical information is leaked. As I have additionally pointed out, the interactions with security mostly come from error conditions, not normal processing (authentication being an important exception).

Security literature and experience is rife with examples of security that did not provide the promised protection in the face of real users. Both security and usability have a tradition of **testing** mechanisms either in **laboratory** setups or **actual use**. Security is tested through red teaming or ethical hacking of deployed systems. Usability testing can be either lab testing with a structured set of tasks, or *in situ* testing through contextual analysis or logs [45, 4].

Examples of user testing of security functionality are still modest in number in the literature [59, 51, 1, 53, 27, 16], although there are some very early examples of applying usability techniques to security messages [23]. Some of the existing studies are lab-based studies, emulating attacks in that context. Others are detailed interviews with people about their use of security and privacy technology (for example, passwords, location finders). A recent phishing attack study attempting to show how successful modest social engineering could be as an attack approach garnered some heated complaints, even though it had been cleared by the university's board beforehand [33].

The body of experience testing the usability of security both in the lab and in context will define the techniques and tools we need and can use. It will also generate a body of best practice we can begin to systematize in checklists and expert evaluations. Taking that best practice and making it visible to users and purchasers will apply pressure to raise the level of usable security in systems and products.

3.2. Principles of Usably Secured Systems

[43] used their experience with security and Multics to formulate eight principles of secure systems. One of these was "psychological acceptability", which is usable security. Today we have enough experience to turn the existing body of knowledge into a small set of principles for systems that can be made usably secure.

Whitten [52] lays out two design techniques for usable security: **safe staging** and **metaphor tailoring**. Safe staging is "a user interface design that allows the user freedom to decide when to progress to the next stage, and encourages progression by establishing a context in which it is a conceptually attractive path of least resistance." The intuition is that users should not be forced to make security decisions when they're

trying to get something else done and don't have enough information to make them yet anyway. Many active content and trust dialogs do not provide safe stages. Some of the more usable dialogs for setting security policy information for access control and active content allow the user to use large granularity defaults, and to proceed to configurable or fine grained security settings as needed.

Metaphor tailoring starts with a conceptual model specification of the security related functionality, **enumerates the risks of usability failures** in that model, and uses those risks to explicitly drive visual metaphors. A more general restatement of the core of the principle is "Incorporate risks of usability failures into the security model". This principle can be applied even to security relevant parts of a system when the actual context of use is undetermined, such as security standards. Use case scenarios may be required before usability failures can be described.

Additional principles can be derived from the work on presenting and visualizing security, from metaphor tailoring through work on visualization [12, 13, 14] and explanation and enforcement [55]. I will call these **integrated security** and **transparent security**. Integrated security aligns security with user actions and tasks so that the most common tasks and repetitive actions are secure by default. Inability to design, implement, and deploy integrated security is an indication that a task or action is not securable. A task or action that is not securable needs to be redesigned so that it is securable. Producing integrated security can lead to an iterative process between the task's functionality and security aspects, until the integration is occurs.

Transparent security provides information about security state and context in a form that is useful and understandable to the user, in a non-obtrusive fashion. For example, in contexts such as email, where social engineering attacks such as scam-spam and phishing are a concern, proactive display of the reliability of the sender information (digital signature, all mail headers indicating a full transmission path behind the enterprise firewalls) is warranted, as long as it is well designed to fit within the context of all the other per mail message information displayed to the user. Conversely, immediate indicators that the current web page was delivered encrypted via SSL are likely to be of secondary importance. The security the user needs to know about for their next action is whether any submission of data from that web page will be encrypted.

Transparent security might be easiest to achieve in the subcategory of protection mechanisms that provide privacy. The desired privacy protections on personal information are by definition determined by the user themselves. The user can understand the risk of exposure and misuse of the information, since the risk is directly to and about them.

Transparent security can do more than explain security. It can highlight anomalies that indicate problems making them more obvious or understandable to non-technical users. It can reassure the user and promote trust in the vigilance of the software. Going one step further, Bill Cheswick suggests software that lets out a groan whenever a preventable problem is detected can train the user to use security more effectively [8]. Persuasion literature [37] teaches that if a task is important or particularly engaging, users will apply intense analytical processing. Otherwise they will apply simple heuristics. Even though security researchers and developers all believe security to be both important and engaging, since it is almost never the user's primary task, users do not. An open research question is how effectively **play and humor** can be used to bring the user's level of engagement in line with the importance of any additional actions required to ensure security.

As an aside, an unexplored area of research is how visible security can be used to **dissuade attackers**. By analogy, "Neighborhood Watch" signs and very visible (sometimes fake) cameras are used as deterrents to vandalism and robbery of physical goods.

A principle implicit in many approaches is **reliance on trustworthy authority**. Implementing usable security in the first place relies on architects, designers, and developers to provide it. Giving administrators the ability to configure security policy for users and resources in their domain puts the responsibility on them to make the right choices (or accept the defaults from developers). These knowledgeable and responsible people can be relied on. Approaches that integrate visualization of community information [13] or security decisions based on evaluation of the activities of other users [19] presume that information about how a community or group or organization is making security decisions can reliably inform similar personal decisions. As the authority being relied on becomes more diffused, trust and security decisions may be susceptible to "flash crowds" [35]. If instead it provides a damping effect, community information may be the best weapon we develop to resist social engineering, since it uses one social process to counteract the abuse of others.

4. Conclusion

As the CRA conference found, the challenge of usable security is grand. We need work at the social, technical and production levels to meet the challenge. We have some HCI techniques and some usable security principles to take us to the next level. Expert evaluation and user testing are producing effective

usable security today. Principles such as safe staging, enumerating usability failure risks, integrated security, transparent security and reliance on trustworthy authorities can also form the basis of improved systems.

There are many open research problems. How do we build security mechanisms that are usable, with no additional education or explanation? How do we set the tone for explanations of security breaches so that blaming a user is not an option? How do we extend HCI techniques to security error cases? How can we encourage marketing pull of usable security? What low impact processes can be used soon to raise the bar on the lower end of usable security? How do we model users as part of the system security? How can constraints simplify the decisions thrust upon users? How do we get feedback from development, deployment and use into the research process rapidly? How do we design reusable security components and specifications that participate in usable security?

It's an area where many types of people are needed for us to make progress. We need researchers and developers attracted to issues that are both system wide and pragmatic, practitioners who can synthesize multiple disciplines, and innovative thinkers and doers of all sorts.

Acknowledgements

This essay profited from review by Steve Greenwald and Serge Egelman. Charlie Payne volunteered his time to help with IEEE guidelines. While this would not have been possible without support from IBM and Doug Wilson, all opinions expressed are those of the author.

Bibliography

[1] Anne Adams and Martina Angela Sasse, "Users Are Not The Enemy", Communications of the AM, vol. 42, issue 12, December 1999, pp 40 – 46.

[2] J. P. Anderson, Computer Security Technology Planning Study, ESD-TR-73-51, Bedford, MA, USAF Electronics Systems Division, October 1972.

[3] Dirk Balfanz, Glenn Durfee, and D. K. Smetters, "Making the Impossible Easy: Usable PKI", Security and Usability: Designing Secure Systems that People Can Use, Lorrie Faith Cranor and Simson Garfinkle, ed., O'Reilly, August 2005.

[4] H. Beyer and K. Holtzblatt, Contextual Design: Defining Customer Centered Systems, Morgan Kaufmann Publishers, 1998.

[5] Jordy Berson, "ZoneAlarm: Creating Usable Security Products for Consumers", Security and Usability: Designing Secure Systems that People Can Use, Lorrie Faith Cranor and Simson Garfinkle, ed., O'Reilly, August 2005.

[6] Bob Blakley, Ellen McDermott, and Dan Geer, "Information Security is Information Risk Management", New Security Paradigms Workshop, ACM Press, Cloudcroft, New Mexico, 2001, pp. 97 – 104.

[7] Benjamin Brunk, "A User-Centric Privacy Space Framework", Security and Usability: Designing Secure Systems that People Can Use, Lorrie Faith Cranor and Simson Garfinkle, ed., O'Reilly, August 2005.

[8] Bill Cheswick, "My Dad's Computer, Microsoft, and the Future of Internet Security", http://cups.cs.cmu.edu/soups/2005/program.html#ches

[9] Common Criteria for Information Technology Security Evaluation, version 2.2, January 2004, http://www.commoncriteriaportal.org/public/expert/index.php?menu=2

[10] CRA Confernence on Grand Research Challenges in Information Security & Assurance, http://www.cra.org/Activities/grand.challenges/security/home.html

[11] Lorrie Faith Cranor, Privacy Policies and Privacy Preferences, Security and Usability: Designing Secure Systems that People Can Use, Lorrie Faith Cranor and Simson Garfinkle, ed., O'Reilly, August 2005.

[12] Rogerio de Paula, Xianghua Ding, Paul Dourish, Kari Nies, Ben Pillet, David Redmiles, Jie Ren, Jennifer Rode, Roberto Silva Filho, Two Experiences Designing for Effective Security, Proceedings of the 2005 Symposium On Usable Privacy and Security, Pittsburgh, Pennsylvania, pp. 25 – 34.

[13] Paul DiGioia, Paul Dourish, "Social Navigation as a Model for Usable Security", Proceedings of the 2005 Symposium On Usable Privacy and Security, Pittsburgh, Pennsylvania, pp. 101 – 108.

[14] Paul Dourish and David Redmiles, "An Approach to Usable Security Based on Event Monitoring and Visualization", Proceedings of the 2002 workshop on New Security Paradigms, Virginia Beach, Virginia, pp. 75 – 81.

[15] Scott Flinn and Steve Stoyles, "Omnivore: Risk Management Through Bidirectional Transparency", Proceedings of the 2004 Workshop on New Security Paradigms, pp 97 – 105.

[16] Simson L. Garfinkel and Robert C. Miller, "Johnny 2: A User Test of Key Continuity Management with S/MIME and Outlook Express", Proceedings of the 2005 Symposium On Usable Privacy and Security, Pittsburgh, Pennsylvania, pp. 13 – 24.

[17] Morrie Gasser, Building a Secure Computer System, Van Nostrand Reinhold, New York, NY, USA, 1988.

[18] Virgil D. Gligor, Shyh-Wei Luan, and Joseph N. Pato, On Inter-Realm Authentication in Large Distributed Systems, Proceedings of the 1992 IEEE Symposium on Security and Privacy, page 2.

[19] Jeremy Goecks and Elizabeth D. Mynatt, Social Approaches to End-User Privacy Management, Security and Usability: Designing Secure Systems that People Can Use, Lorrie Faith Cranor and Simson Garfinkle, ed., O'Reilly, August 2005.

[20] Nathaniel Good, Rachna Dhamija, Jens Grossklags, David Thaw, Steven Aronowitz, Deirdre Mulligan, and Joseph Konstan, "Stopping Spyware at the Gate: a User Study of Privacy, Notice and Spyware", Proceedings of the 2005 Symposium On Usable Privacy and Security, Pittsburgh, Pennsylvania, pp 43 – 52.

[21] Steven J. Greenwald, personal communication.

[22] Steven J. Greenwald and Richard E. Newman, The Distributed Compartment Model for Resource Management and Access Control, University of Florida Computer and Information Sciences Department technical report, October 1994.
[23] O. Gruber, B. J. Hargrave, J. McAffer, P. Rapicault, and T. Watson, "The Eclipse 3.0 platform: Adopting OSGi technology", IBM Systems Journal, v. 44, # 2, page 289, 2005.
[23] Clare-Marie Karat, Iterative Usability Testing of a Security Application Computer Systems: Approaches to User Interface Design, Proceedings of the Human Factors Society 33rd Annual Meeting, 1989, v. 1, pp. 273 – 277.
[24] HCISec Bibliography, http://www.gaudior.net/alma/biblio.html.
[25] Security and Usability: Designing Secure Systems that People Can Use, Lorrie Faith Cranor and Simson Garfinkle, ed., O'Reilly, August 2005.
[26] hcisec – Computer Security and Usability, http://groups.yahoo.com/group/hcisec/.
[27] Giovanni Iachello, Ian Smith, Sunny Consolvo, Mike Chen, and Gregory D. Abowd, "Developing Privacy Guidelines for Social Location Disclosure Applications and Services", Proceedings of the 2005 Symposium On Usable Privacy and Security, Pittsburgh, Pennsylvania, pp. 65 – 76.
[28] Information technology – Security techniques – Code of practice for information security management, http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=39612&ICS1=35&ICS2=40&ICS3=
[29] Nancy Leveson and Clark S. Turner, "An Investigation of the Therac-25 Accidents", IEEE Computer, Vol. 26, No. 7, July 1993, pp. 18 – 41.
[30] Steven B. Lipner, "Practical Assurance: Evolution of a Security Development Lifecycle", Proceedings of 20th Annual Computer Security Applications Conference, Tucson, Arizona, December 2004.
[31] Lock Box and Easy Savings Bank with Key Combination, http://www.babyscholars.com/loboxwikeyan.html
[32] John McHugh, "Intrusion and Intrusion Detection", IJIS, 2001, v. 1, pp 14 – 35.
[33] Robert Miller, Simson Garfinkel, Filippo Menczer, Robert Kraut, "When User Studies Attack: Evaluating Security By Intentionally Attacking Users", 2005 Symposium On Usable Privacy and Security, Pittsburgh, Pennsylvania.
[34] Fabian Monrose and Michael K. Reiter, Graphical Passwords, Security and Usability: Designing Secure Systems that People Can Use, Lorrie Faith Cranor and Simson Garfinkle, ed., O'Reilly, August 2005.
[35] Hilarie Orman and Richard Schroeppel, "Positive Feedback and the Madness of Crowds", Proceedings of the 1996 Workshop on New Security Paradigms, Lake Arrowhead, California, United Stages, pp. 134 – 138.
[36] Susan Pancho, "Paradigm Shifts in Protocol Analysis", Proceedings of the 1999 Workshop on New Security Paradigms, pp 70 – 79.
[37] Andrew S. Patrick, Pamela Briggs, and Stephen Marsh, "Designing Systems That People Will Trust", Security and Usability: Designing Secure Systems that People Can Use, Lorrie Faith Cranor and Simson Garfinkle, ed., O'Reilly, August 2005.
[38] Dean Povey, "Optimistic Security: A New Access Control Paradigm", Proceedings of the 1999 Workshop on New Security Paradigms, Caledon Hills, Ontario, Canada, pp. 40 – 45.
[39] J. Reason, Human Error, Cambridge, UK; Cambridge University Press, 1990.
[40] Karen Renaud, "Evaluating Authentication Mechanisms", Security and Usability: Designing Secure Systems that People Can Use, Lorrie Faith Cranor and Simson Garfinkle, ed., O'Reilly, August 2005.
[41] Blake Ross, "Firefox and the Worry-free Web", Security and Usability: Designing Secure Systems that People Can Use, Lorrie Faith Cranor and Simson Garfinkle, ed., O'Reilly, August 2005.
[42] Anne Saita, "Password protection no match for Easter egg lovers", http://searchsecurity.techtarget.com/originalContent/0,289142,sid14_gci960468,00.html
[43] Jerome H. Saltzer and Michael D. Schroeder, "The Protection of Information in Computer Systems", Fourth ACM Symposium on Operating System Principles, October 1973.
[44] M. Angela Sasse and Ivan Flechais, "Usable Security ", Security and Usability: Designing Secure Systems that People Can Use, Lorrie Faith Cranor and Simson Garfinkle, ed., O'Reilly, August 2005.
[45] Ben Schneiderman, "Designing the User Interface", Addison-Wesley Publishing Company, 1997.
[46] Bruce Schneier, Crypto-Gram Newsletter, October 15, 2000, http://www.schneier.com/crypto-gram-0010.html.
[47] Symposium On Usable Privacy and Security, http://cups.cs.cmu.edu/soups/
[48] Jared M. Spool, Tara Scanlon, and Carolyn Snyder, "Product Usability: Survival Techniques", CHI Extended Abstracts 1997, pp 154 – 155.
[49] Bruce Tognazzini, "Design for Usability", Security and Usability: Designing Secure Systems that People Can Use, Lorrie Faith Cranor and Simson Garfinkle, ed., O'Reilly, August 2005.
[50] Win Trees, Peter Neumann, Mary Ellen Zurko, and Mark Ackerman, "Security and the User", Proceedings of the 1999 Network and Distributed System Security Symposium.
[51] Alma Whitten and J. D. Tygar, "Why Johnny Can't Encrypt: A Usability Case Study of PGP 5.0", Proceedings of the 8th USENIX Security Symposium", August 1999.
[52] Alma Whitten and J. D. Tygar, "Safe Security Staging", CHI 2003 Workshop on Human-Computer Interaction and Security Systems, Ft. Lauderdale, Florida.
[53] Dirk Weirich and Martina Angela Sasse, "Pretty Good Persuasion: A First Step Towards Effective Password Security in the Real World", Proceedings of the 2001 Workshop on New Security Paradigms, Cloudcroft, New Mexico, pp. 137 – 143.
[54] Simon R. Wiseman, "Security Properties of the SWORD Secure DBMS design", Database Security, VII: Status and Prospects, Proceedings of the IFIP WG11.3 Working Conference on Database Security, Lake Guntersville, Alabama, USA, September 1993, pp 181 – 196.
[55] Haidong Xia and Jose Carlos Brustoloni, "Hardening Web Browsers Against Man-In-The-Middle Eavesdropping Attacks", The 14th International World Wide Web Conference, ACM Press, Japan, 2005, pp. 489 – 498.

[56] Jeff Yan, Alan Blackwell, Ross Anderson, and Alasdair Grant, "The Memorability and Security of Passwords", Security and Usability: Designing Secure Systems that People Can Use, Lorrie Faith Cranor and Simson Garfinkle, ed., O'Reilly, August 2005.
[57] Ka-Ping Yee, "Aligning Security and Usability", IEEE Security and Privacy, vol. 02, no. 5, pp. 48 – 55, September – October 2004.
[58] Mary Ellen Zurko and Richard T. Simon, "User-Centered Security", Proceedings of the 1996 Workshop on New Security Paradigms, Lake Arrowhead, California, United States, pp. 27 – 33.
[59] Mary Ellen Zurko, Rich Simon, and Tom Sanfilippo, "A User-Centered, Modular Authorization Service Built on an RBA Foundation", IEEE Symposium on Security and Privacy, 1999, pp. 57 – 71.
[60] M. E. Zurko, J. Wray, I. Morrison, M. Shanzer, M. Crane, P. Booth, E. McDermott, W. Macek, A. Graham, J. Wade, and T. Sandlin, "Jonah: Experience Implementing PKIX Reference Freeware", Proceedings 8th USENIX Security Symposium", Washington, DC, August 1999, pp. 185 – 200.
[61] Mary Ellen Zurko, Charlie Kaufman, Katherine Spanbauer, Chuck Bassett, Did You Ever Have To Make Up Your Mind? What Notes Users Do When Faced With A Security Decision, Proceedings of 18th Annual Computer Security Applications Conference, Las Vegas, Nevada, December 2002.
[62] Mary Ellen Zurko, "IBM Lotus Notes/Domino: Embedding Seurity in Collaborative Applications", Security and Usability: Designing Secure Systems that People Can Use, Lorrie Faith Cranor and Simson Garfinkle, ed., O'Reilly, August 2005.