

# Three Challenges for Embedding Security into Applications

Rebecca E. Grinter and D. K. Smetters

Palo Alto Research Center (PARC)

3333 Coyote Hill Road

Palo Alto, CA 94304 USA

{surname}@parc.com

## ABSTRACT

In order to use them securely, current computer systems require end users to understand both the threats they are subject to and the details of technology necessary to protect against them. This is a requirement they have as of yet been unable to meet. We present thoughts about a new approach to building systems that are both usable and secure. Critically, we argue that this cannot be accomplished merely by providing a better interface on top of existing security technology and systems, that instead the underlying technology itself will need to change. We have begun a research program to build new, more usable security technologies and security-conscious applications. In this paper we present 3 interface design challenges we have found as a result of our experiences, and invite the HCI community to participate in addressing them.

## Keywords

Usability, usefulness, security

## INTRODUCTION

Work on improving the usability of security has concentrated on providing better interfaces to existing systems and security technologies. For example, making PGP more intuitive to use through the improvement of feedback and interface design [7]. Or work that looks at the failings of password systems, and attempts to figure out how to make them more effective [1]. We agree that many contemporary security systems could be improved with techniques such as these. Good interface layout, thoughtful prompts, and helpful feedback are central tenets of HCI design. However, we also believe that these approaches may not be sufficient to make security usable. They may also not extend well to mobile devices and other systems outside of the traditional office desktop metaphor.

We suggest instead improving the usability of security will require starting from the user — designing secure systems, and even the security technologies that underlie them in

terms of what tasks users are attempting to accomplish. We propose the use of a design technique that we term *implicit security* — embedding security directly in applications, so that when a user undertakes a task that requires changes to her security state to accomplish, the application can automatically make those, and only those, changes necessary to accomplish the task at hand [6]. This removes the cumbersome “separate but parallel” approach to security in most traditional applications, where the “security settings” have to be correctly set in a vacuum divorced from application actions. This puts an unnecessary burden on users, and at the same time, removes even the minimal context provided by an application setting to help a user decide whether a security change is appropriate for the task at hand.

Building such systems in practice requires not only new security building blocks but also poses several critical HCI design challenges. First, to align security actions with end-user needs, we must identify users’ tasks and context of action and understand the security implications, and identify what typical users “expect” should happen to their security state as a result. Second, we must design interfaces capable of inferring these goals from user actions and context, and security technologies that make it easier for application designers to “make it so”. And third, we must feed back information about that security state to the end user — make those security-related aspects of user tasks directly visible to the end user, in the terms they understand.

## THE PROBLEM

Many applications have some type of security built into them. For example, Microsoft's Internet Explorer has two configuration dialogs that allow end-users to configure a number of security settings. It also often pops up additional dialogs asking users to make other security decisions on-the-fly. Unfortunately, these dialogs and settings do not pose security questions in any way that users are prepared to answer. Instead, they require end-users to understand both the technical details of an attack that they could be subject to (e.g. cross-site scripting of active X controls), and the details of the technologies available in the browser to increase their security (e.g. the option to decide whether to permit downloading of any

*Copyright Remains with Authors*

active X controls or to ask on each occasion). The browser attempts to hide some of the complexity of these decisions using a set of preconfigured suggested settings applied to different source web sites depending on the “zone” they are in — the local, trusted, Internet or restricted zone. Even this simplified structure is difficult for users to grasp. Similarly, all browsers attempt to provide interface feedback as to when information is being sent over a secure (encrypted) connection. Unfortunately, studies of users’ conceptions of web security show that these attempts are unsuccessful — users can not always tell when a connection from their browser is secure [4].

Other applications may not present the user with explicit security settings to configure, but may still present multiple options to users for different applications, where the option chosen can have significant security implications. For example, Apple’s iChat technology allows users to interact with colleagues on local networks in two ways. If everyone has an IM account they can chat through an external server; however, if they are all on the same local subnet they can also interact using Apple’s peer-to-peer technology, Rendezvous. Users take identical steps to start conversations via IM or Rendezvous, but in the latter case do not expose those conversations to the external network, making the peer-to-peer alternative more secure for corporate confidential discussions. It is not clear to users which network mechanism they will be using to communicate the data they send as part of any particular IM conversation, and therefore how secure that conversation will be.

An illuminating approach to understanding why these systems fail can be seen in the recent work of Yee [8], which gathers in one concise list ten important principles for user interaction design for secure systems. Looking at the IM example in these terms, we can see that one major problem with this interface is one of *identifiability* — it’s not apparent to the user what connection mechanism she is using to communicate with another user, not to mention the underlying security consequences of the method chosen.

It is clear that the problems with the IM example go farther than just informing the user about the security level of her current communication medium. An effective IM system would at least employ a *path of least resistance* approach to security — it should simply send all data by the most secure method available at all times. Even better, it would endeavor *a priori* to provide a secure method to transfer data. At the same time, it has to do this without ending up with a set of Internet Explorer-style dialog boxes asking the user to make decisions about trusted root certificates.

We suggest that using many, if not most, of Yee’s principles to build secure and usable systems will require reconsidering how security technologies are integrated into applications, and even building new security technologies that integrate more effectively with application tasks.

## THE THESIS: EMBED SECURITY INTO APPLICATIONS

Security cannot be considered in the abstract, separate from a particular application and context of use. This is well understood by the security community — but only to a point. The design of a security-relevant system or technology begins with the construction of a “threat model” — an assessment of the potential threats to that system and the resources of the adversaries that might carry out those threats, and a decision about which of those threats the system will be designed to protect against (the others may be too expensive, or too unlikely to be deemed worthy of the cost of prevention).

Such an analysis, however, doesn’t consider the user as part of the equation. The work of Sasse and others (e.g. [1,5]) has argued that to be effective, security has to be designed in terms of 1) the user, 2) the task that they are using the technology to accomplish, and 3) the context of that task activity (for example, whether it’s in the office or at home).

We believe that designing security from this point of view actually improves the prospect for making security usable. Knowing the task the user is currently attempting, a system designer can infer that the user wants all security-related steps necessary to accomplish that task securely to take place as well — but nothing more. By taking the user, the context, and the task together, it also becomes possible to structure an interface that makes security-related concepts more visible to the user by framing them in task terms. This requires, however, embedding security directly into the context of an application.

We have termed this approach to application design *implicit security* [6], and have been attempting to build usable, secure systems along these lines. For example, in [3] we describe Casca, a system designed to let users securely share arbitrary resources (data, services, and devices) with particular groups of other users. Users can construct “spaces” into which they can invite other members. They then can add resources to that space that they wish other members, and only other members, to be able to access.

These efforts have led us to two conclusions. First, many existing security technologies do not map easily onto the task-centered approach. We have therefore been attempting to design new security “building blocks” that can be used to more easily build secure applications (for an example, see [2]). Second, and more important for this context, we have identified a number of HCI design challenges that arise from this approach. In the rest of this position paper we describe three challenges in more detail.

## CHALLENGE 1: STARTING WITH A USER-CENTERED THREAT MODEL

Design of an appropriate threat model is an important step in building a secure system. We suggest, however, that system design must first begin with a user model — an analysis of what tasks the user is trying to accomplish, and what the concepts are that user has to work with, before beginning the design of a traditional threat model. The

resulting system must actually be capable of accomplishing the user's desired tasks, even if the system as a whole ends up less secure as a result.

In addition, such an analysis can also attempt to understand the user's expectations about security. For instance, a user of IM or email may simply expect that a message sent to Bob will reach Bob, and only Bob. Identifying such expectations helps us to determine what the path of least resistance should be — instead of attempting to inform the user that their expectations are incorrect, attempt to build a system that meets the user's expectations. For example, in contrast to the IM client, the Casca application endeavors to meet the user's naïve security assumptions — only other members of the shared space can indeed see the resources placed therein, and encryption and strong authentication is automatically used to protect all data exchange between space members from undesired observation or manipulation.

Identifying these expectations can often be difficult, as it is hard for users to identify and articulate them as security concerns rather than system expectations. More work is necessary to understand what expectations users have of the applications that they use (in context of their tasks) and how they express those assumptions. Moreover, dialog between HCI and security researchers is necessary to understand how to take user-centered findings and integrate them into the threat model.

## **CHALLENGE 2: INFERRING SECURITY ACTION FROM USER INTENT**

Another, even larger challenge in injecting effective HCI design into the security-related components of applications is to make *implicit security* real — to infer what necessary changes in security state are implied by the user's actions. If we want to make applications that don't require users to understand and perform a myriad of apparently unrelated security tasks in order to achieve their application goals, we need to make those applications clever and responsive enough to know what necessary “security tasks” are implied by a given user action as termed in the context of the application at hand. This requires identifying the user tasks embodied an application, and determining what security actions must take place in order to let the user accomplish her task at hand.

System design can then attempt to make sure that those required security actions, and only those, take place *implicitly* as part of the task actions that make them necessary. In the context of the Casca application, construction of a shared “space” causes the system to create a set of cryptographic credentials necessary to control access to that space. Adding another user as a member of the space implicitly causes the application to issue that new member a credential that allows them to access that space, and that also allows all communication with them to be encrypted and authenticated.

A challenge in making this mapping from application tasks to implicit security actions requires making sure that such a

mapping is possible — that the security technologies used are modular in such a way that they can map onto well-defined application tasks. An even more important challenge is to design the user experience in such a way that it is easy to infer from user actions what tasks, and hence what security actions, they are attempting to accomplish at any one time. While the Casca application provides an example where the mapping from user action to security consequence is very easy to determine, taking this approach into different contexts (e.g. mobile environments) may make this both easier, by giving more direct expression to user intent, and more difficult.

### **Subchallenge: “Trusted Path”**

An important additional challenge for any system that wants to infer user intent from user action is to make sure that “the user” performing that action is really the user we think it is. We can easily infer when a user, Alice, emails a document to another user, Bob, that she wants Bob to be able to read that document, and to receive it without being tampered with. She's also made no statement that she wishes anyone else to be able to see the document, so that we can assume we should take steps to prevent it from getting into any hands other than Bob's.

On the other hand, with current end-user mail clients, the act of Alice's mail client sending a document to Bob may also mean that an email worm that has infected Alice's computer is emailing arbitrary documents to individuals selected at random from Alice's address book. In the security world, great care is put into building “trusted paths” — channels of information from the user to the system that cannot be manipulated en route. As computer systems become more scriptable and do more and more on their own, a new interface design challenge will arise — differentiating direct user input from other software actions, or in essence, building a trusted path.

## **CHALLENGE 3: REFLECTING SECURITY STATE BACK TO THE USER**

If we have identified user goals with security consequences, we ought to present them as such to the user — provide the user visibility into their security state, and in that context, make it possible to provide interface elements that control and change that state without requiring them to understand arbitrary security technology. Consider the Casca application described above. The user interface of that system is designed entirely around the shared “spaces” constructed by users. It is easy to immediately see both what resources are shared, and with whom. The goal of the Casca application is cooperative work, but the primary task used to achieve that goal is “sharing”. “Sharing” is an understandable user goal, one that users can reason about, and one that is directly reflected in the Casca interface.

Approaches to reflect the state of more traditional security systems back to their users have not been successful [4]. Our approach attempts to move an understandable security concept (“sharing”) to the center of the organization of an application interface — basically, to turn security's usual

“back-room” role on its head. These two alternatives seem to be two ends of a continuum. Work going forward will need to look into how to effectively handle presenting security state to users in the context of applications that necessary fall more into the middle of that continuum — that have their own, non-security related presentation goals to meet.

## REFERENCES

1. Adams, A. and Sasse, M.A. Users Are Not The Enemy. *Communications of the ACM*, 42 (12). 1999. 41-46.
2. Balfanz, D., Smetters, D.K., Stewart, P. and Wong, H.C., Talking to Strangers: Authentication in Ad-Hoc Wireless Networks. In *Proceedings of 2002 Network and Distributed Systems Security Symposium (NDSS '02)*, (San Diego, CA, 2002), Internet Society, 23-38.
3. Edwards, W.K., Newman, M.W., Sedivy, J.Z., Smith, T.F., Balfanz, D., Smetters, D.K., Wong, H.C. and Izadi, S., Using Speakeasy for Ad Hoc Peer-to-Peer Collaboration. In *Proceedings of ACM Conference on Computer Supported Cooperative Work (CSCW 2002)*, (New Orleans, LA, 2002), New York, NY: ACM Press, 256-265.
4. Friedman, B., Hurley, D., Howe, D.C., Felten, E. and Nissenbaum, H., Users' Conceptions of Web Security: A Comparative Study. In *Proceedings of Extended Abstracts of Conference on Human Factors in Computing Systems (CHI 2002)*, (Minneapolis, MN, 2002), ACM Press, 746-747.
5. Sasse, M.A., Brostoff, S. and Weirich, D. Transforming The 'Weakest Link' — A Human/Computer Interaction Approach to Usable and Effective Security. *BT Technology Journal*, 19 (3). 2001. 122-131.
6. Smetters, D.K. and Grinter, R.E., Moving from the Design of Usable Security Systems to the Design of Useful and Secure Applications. In *Proceedings of ACM New Security Paradigms Workshop*, (Norfolk, VA, 2002), ACM Press.
7. Whitten, A. and Tygar, J.D., Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. In *Proceedings of 8th USENIX Security Symposium*, (Washington, D.C., 1999), USENIX, 169-184.
8. Yee, K.-P., User Interaction Design for Secure Systems. In *Proceedings of 4th International Conference on Information and Communications Security*, (Singapore, 2002).