Making Policy Decisions Disappear into the User's Workflow

Alan H. Karp
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304 USA
alan.karp@hp.com

Marc Stiegler
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304 USA
marc.d.stiegler@hp.com

CHI 2010, April 10–15, 2010, Atlanta, Georgia, USA.
ACM 978-1-60558-930-5/10/04.

Abstract
Complaints of security interfering with getting work done often arise when users are distracted from their tasks to make policy decisions. We have identified what is missing from earlier security interaction designs that leads to these interruptions. Explicitly representing policy decisions in the user interface as items relevant to the application and providing application-specific controls for changing those policies has allowed us to reliably infer users' desired policy decisions from actions they take as they work. This paper describes the underlying principles and how they resulted in an interaction design that does not interfere with the user's work.

Keywords
Usable security

ACM Classification Keywords
D.4.6 [Security and Protection] Access Controls; H.5.2 [User Interfaces]: User-centered design

General Terms
Human Factors, Security

Introduction

The user interfaces we encounter daily often interrupt our work to ask us to make policy decisions, such as changing a Facebook privacy setting to share a photo or Windows 7 asking to elevate privilege to carry out a request. The software asks because only the user is in a position to know what policy to apply. What would be the impact of a system that made security invisible because it could infer the user's wishes from actions taken as part of the application tasks? Before we can answer that question, we need such software for testing. The key contribution of this paper is to present principles for interaction designs that make it possible to infer the user's desired policy without asking.

The principles we propose are not specifically designed to reduce human error [13] or for automating the process of making security decisions [5]. These principles might be used to build better interaction designs for authoring security policies [17], but our work primarily focuses on inferring users' policy decisions as they go about their work.

Making security disappear

Today's systems sacrifice usability when adding security. Warning dialog boxes don't provide sufficient information to allow the user to assess the risk. There is no way on existing systems to express which of a user's rights to apply to a request, which is the root cause of the Confused Deputy [7]. Systems do not support attenuated delegation, leaving users with the dilemma of sharing credentials or not getting their work done. Authenticating users when in the middle of a task interferes with their work.

These observations led us to identify four dimensions to avoiding the need to trade usability for security.

Dimension 1: Information

We should give users the information they need to make intelligent decisions. If we don't, they are likely to be unhappy the result of not understanding the implications of the decision.

Following the 10 guidelines for usable security [21] enhances the usability of the security mechanisms. Most of these principles are related to giving users the information they need. For example, "Present objects and actions using distinguishable, truthful appearances." Surprisingly, most systems in common use implement none of these principles.

Dimension 2: Expressiveness

We need to let users express the modes of sharing they need to get their work done. If we don't, they will find the workarounds required to do their jobs an impediment.

A decade of building systems for collaborative work and observing how people share led to a search for commonalities [20]. (We have been surprised by our inability to find references for such a list in either the computer science or the sociology literature.) The six identified aspects of sharing are

- Dynamic: No admin needed to approve a change.
- Cross-domain: No one party is in charge.
- Attenuated: Take a dollar, but not my wallet.
- Chained: Re-delegating a delegated right.
- Composable: Use rights from different sources.
- Accountable: Who is responsible, not who acted.

Each time the security blocks one of these modes of sharing, the user must find a workaround. Despite the abundance of collaboration software products, it is like-

ly that email is so widely used for collaborating because only it supports all six aspects of sharing.

Dimension 3: Control

We must give users the means to express their decisions. If we don't, they will be frustrated by the inability of the application to carry out their wishes. One approach is to open a dialog box for every decision. Such security by admonition interferes with the user's work, often without enforcing the user's desired policy.

CapDesk [19] uses *capabilities* [4] because that is the only mechanism that has been shown to support all six aspects of sharing while enforcing access control at fine granularity. CapDesk adapts the fundamental property of capabilities, combining designation with authorization, to the interaction design by using acts of designation to denote the desired authorizations. For example, in CapDesk dragging the icon for a file onto the icon for an editor designates that the user wants to use that editor with the file. CapDesk infers that the user wants to grant the process running the editor the authority to read and write the designated file. The result is that the user is not distracted from the task of editing the file to specify the desired policy. Existing systems avoid this problem at the cost of violating the Principle of Least Privilege [18] by granting all the user's rights to every process the user runs. CapDesk demonstrates that this large vulnerability to viruses is unnecessary.

Dimension 4: Time

The interaction design must let users make policy decisions at a time that doesn't interrupt their work. Groove [16] asks users to determine the trustworthiness of a message sender's authentication when the message is received, which interferes with the user's task of reading the message. Instead, we can make this authentication step part of a different user task, that of establishing a new relationship. It's the same work, but it's done at a different time so that it becomes part of the user's workflow.

Applying the concepts

Policy decisions are subject to change, and we don't want to interfere with the user's work when they do. CapDesk replicates an existing user experience, that of a conventional desktop. That constraint limits the policy decisions that can be expressed directly in the user interface. We had more freedom with the design of SCoopFS [11], (Simple Cooperative File Sharing, the "F" is silent). We used this freedom to design an interaction that lets us infer users' policy decisions.

ScooFS gets a high score on the information axis because of the way it displays application elements. SCoopFS also gets a high score on the expressiveness axis because it uses capabilities, which lets it support all six aspects of sharing. Like CapDesk, SCoopFS uses acts of designation in the user interface to infer acts of authorization. However, SCoopFS scores higher on the control axis because it represents policy decisions as elements in the application's user interface and provides application-specific mechanisms for manipulating these policies. SCoopFS also scores well on the time axis because care was taken to make sure policy decisions are made when they fit the user's activity.

We proposed interaction designs based on earlier versions of these principles for two HP products. Halo [8] is HP's video conferencing system, which gives people in different locations the sense that they are at a common table. While the physical components met that goal, the user interface for such things as dial-in participants did not. Our proposal [15], which was par-

tially adopted for the product, uses a visual metaphor to represent the shared elements in the room, such as the telephone and overhead cameras.

Another part of our proposal, which was not adopted, allowed control over who could book which rooms. Each user would have a web page listing the rooms that user could book. Each page would have a means to delegate the right to book a subset of those rooms and a means to revoke such delegations.

We also proposed a similar system for managing the physical resources in the HP-Intel-Yahoo! Cloud offering, Open Cirrus [9]. Users would manage their allocations of CPUs and storage from a web page showing what was in use, what they had delegated to whom, and what was currently available. Widgets in the interface were to provide for managing policy decisions made on these resources.

Policy decisions as controllable objects

We identified four principles that let the system infer the desired policy from the user's workflow.

- Every object separately controllable by the user should be represented in the application user interface by a capability that is uniquely distinguishable to the user.
- Every possible policy decision on an object should appear as a unique affordance in the application user interface.
- Every policy decision the user has made should be represented in the application user interface by a capability that is uniquely distinguishable to the user.
- Every possible change to a previously made policy decision should appear in the application user interface as a unique affordance.

If the system follows these principles, every action taken in the user interface that affects policy will be unique. Since there is no ambiguity in determining the user's intent, there is no need to interrupt the user's work with a question.

Figure 1 shows an application of the last two of these principles. The second line in the grid shows that "Me," a pseudonym for the user, granted read and write permission (the double headed arrow under "Mode") for the file "decideRightSetup.zip" to "AlanXP" on November 25. The grayed buttons show the actions that the

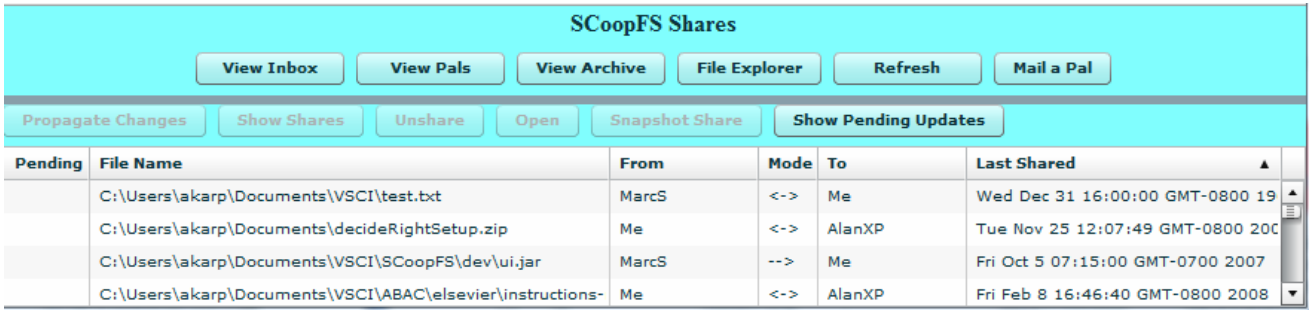

Figure 1: Shares view in SCoopFS.

user can take on this sharing relationship. In this case they are only "Unshare," which revokes the privileges, and "Snapshot Share," which makes a private copy of the file in its current state.

Figure 1 also shows how we use filtering to simplify dealing with large numbers of policy decisions. Clicking the "Show Shares" button when an item is selected results in a view showing all sharing relationships involving that file. Another view shows all the sharing relationships with a given party.

Other considerations

We have shown how these principles apply to access control, but there are other kinds of security policy. For example, deciding whether or not to encrypt communications is often left to the user. One approach that lets the user decide is to make the communication channel a "separately controllable object" and include separate buttons for sending encrypted or not.

The danger with all these affordances is an overly complex user interface. How to avoid this problem depends on the application space. SCoopFS attaches some properties to the communication channel and others to the sharing relationship, as well as eliminating some choices. These design decisions limit the user's options in ways that make sense for the application domain.

An interface that lets the user specify dozens of actions on each of millions of objects will necessarily be complex. Following the guidelines presented here reduces complexity by not interrupting the user's work to make policy decisions.

Related work

There are numerous guidelines for designing for usability, *e.g.*, [6], but that work does not mention including the policy decisions in the interaction design. Other work, *e.g.,* [1], makes interruptions of the user's work less onerous, while our goal is to avoid those interruptions entirely. A key goal of Chameleon [12] is not to "interfere too much with the primary task" nor "intrude on the ordinary activities that people want to perform." Our principles go beyond those goals by attempting to avoid any interference with the user's primary task.

Many systems don't support rich sharing. A spouse can't get to the employee's electronic pay stub because it's behind the company firewall (cross-domain). Often, managers are forced to share their Windows domain credentials with those who take care of minor budgeting and personnel matters (attenuated). The implementation of simple service chaining done for the US Navy can achieve either the desired functionality (chained) or the required security (composable), but not both at the same time [10]. Microsoft Live Mesh [14] only fully supports two of the six aspects (dynamic, cross-domain).

Conclusion

We didn't start out by dreaming up a set of principles and building tools using them. Instead, we built tools and discovered that we weren't bothering our users. The articulation of the principles came from asking ourselves how we did it, a form of *post-hoc* synthesis [2].

The primary contribution of this paper is to show that making policy decisions explicitly controllable objects makes it possible to give the user the desired control without needing to leave the task at hand. We are now applying these principles as we build prototypes, such as a secure shell that supports rich sharing, so that we have enough applications to study the implications of making security invisible. The danger is illustrated by the user who asked how to turn on security, which led

us to ask, "Will users accept an application that is secure if they can't `see` the security?"

Acknowledgements
We thank Jhilmil Jain and April Mitchell for help in writing of this paper and the referees of an earlier version for their guidance.

Citations

[1] Cao, X. and Iverson, L. Intentional access management: making access control usable for end-users. Proc. SOUPS'06 (2006)

[2] Cockton, G. Getting There: Six Meta-Principles and Interaction Design. CHI 2009, Boston, MA. (2009)

[3] Cranor, L. F. and Garfinkel, S. *Security and Usability: Designing Secure Systems That People Can Use*. Sebastopol, CA: O'Reilly Media, Inc., (2005)

[4] Dennis, J. B. and Van Horn, E. C. 1966. Programming semantics for multiprogrammed computations. *Comm. ACM* 9, 3, pp. 143-155 (1966).

[5] Edwards, W. K., Poole, E. S., and Stoll, J. 2008. Security automation considered harmful? Proc. NSPW '07, pp. 33-42

[6] Gould, J. and Lewis, C. Designing for Usability: Key Principles and What Designers Think. CACM, 28(3), 300-311, (1985)

[7] Hardy, N., "The Confused Deputy", *Operating Systems Reviews*, **22**, #4, 1988

[8] Hewlett-Packard Company, http://www.hp.com/halo/introducing.html

[9] Hewlett-Packard Company, Intel, Yahoo!, Open Cirrus, https://opencirrus.org/

[10] Karp, A. H. and Li, J. Solving the Transitive Access Problem for the Services Oriented Architecture, HP Labs Technical Report, HPL-2008-204R1 (2008), accepted for ARES 2010

[11] Karp, A. H., Stiegler, M., Close, T., Not One Click for Security, HP Labs Tech Report, HPL-2009-53 (2009)

[12] Long, C. A. and Moskowitz, C. Simple Desktop Security with Chameleon. In [3] pp. 335-356.

[13] Maxion, R. A., and Reeder, R. W. Improving user-interface dependability through mitigation of human error. Int. J. of Human-Computer Studies, vol. 63, *2005, pp. 25-50*

[14] Microsoft Corp. What's inside Live Mesh? https://www.mesh.com

[15] Mitchell, A. S. and Karp, A. H. Improving Usability by Adding Security to a Video Conferencing Collaboration System, LNCS, #4886, pp. 378-382, 2008.

[16] Moromisato, G., Boyd, P., and Asthagiri, N. Achieving Usable Security in Groove. In [3] pp. 623-636.

[17] Reeder, R. W., Bauer, L., Cranor, L. F., Reiter, M. K., Bacon, K., How, K., and Strong, H. 2008. Expandable grids for visualizing and authoring computer security policies. In Proc. CHI '08. pp. 1473-1482, 2008.

[18] Saltzer, J. H and Schroeder, M. D. The Protection of Information in Computer Systems. *Comm. ACM* 17, 7, 1974.

[19] Stiegler, M. A Capability Based Client: The DarpaBrowser. http://www.combex.com/papers/darpa-report/darpaBrowserFinalReport.pdf (2002)

[20] Stiegler, M. Rich Sharing for the Web. HP Labs Technical Report, HPL-2009-169 (2009)

[21] Yee, Ka-Ping. "Guidelines and Strategies for Secure Interaction Design." In [3] pp. 247-273