Not One Click for Security?

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

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

Tyler Close
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304
tyler.close@hp.com

1. INTRODUCTION

Most people agree with the statement, "There is an inevitable tension between usability and security." We don't, so we set out to build a useful tool to prove our point. Since people in our line of work often share work on documents, such as this paper, we decided to build a file sharing tool. An informal survey revealed that everyone we asked used email attachments to share versions of the document when firewalls caused problems. Based on this result, we adopted an email metaphor for SCoopFS (The "F" is silent.), a system for Simple Cooperative File Sharing.

SCoopFS allows users to manage read and write authorities on single files using only the actions they commonly take when sharing files. At the same time, SCoopFS provides server authentication, so there is no vulnerability to DNS redirection attacks, client authorization, in a way that reduces administrative overhead, and is designed for end-to-end encryption, but the user never sees the security infrastructure. In fact, the goal of the user interface is to include nothing specific to security.

This poster will include a live demo showing how users set up and manage sharing relationships. Participants will also be able to install SCoopFS on their own machines. We will keep a running tally of the answer participants give to the question in the title of this abstract.

2. SIX ASPECTS OF SHARING

*Due to an emergency (**dynamic**), Bob asks Alice to have her son (**cross-domain, chained**) put Bob's car in Carol's garage (**composable**), all while being unable to open the car's trunk (**attenuated**) yet being held responsible for mishaps (**accountable**).*

This story illustrates the six aspects of sharing that people rely on in the physical world, some of which they have been forced to give up online because virtually all of today's systems decide whether or not to honor a request based on the authentication of the requester. We refer to this approach as autheNtication-Based Access Control (NBAC), whether it is identity (IBAC), role (RBAC), or attributes (ABAC) being authenticated.

Dynamic: As our relationships change, so too should the sharing of our resources, both to add new sharings and remove existing ones. Changing access rights in NBAC systems often requires adding users to the system or manipulating security critical data structures, work often done by a system administrator.

Chained: After Bob delegates to Alice, Alice must be able to further delegate to her son. NBAC systems often make chained delegation hard because of the special nature of the "owner" of a resource.

Cross-Domain: Operating across administrative boundaries is difficult in NBAC systems because of the need to federate identities or reach prior agreement on the meaning of roles or attributes. Firewalls introduce additional challenges in such systems

Composable: Alice's son needs to combine the right represented by Bob's car key with the right represented by Carol's garage key. In common NBAC approaches, Alice's son is able to use either Bob's authentication or Carol's, but not both at the same time.

Attenuated: Bob grants Alice attenuated authority by giving her his car keys instead of his entire key ring. Alice further attenuates by giving her son only the valet key to Bob's car. In NBAC systems, the ability to attenuate a permission is usually associated with the ability to increase the rights of that permission.

Accountable: If Bob delegates to Alice, and Alice delegates to her son, Bob must be able to hold *Alice* accountable for her son's actions. NBAC systems record who did something bad but often lose track of who is responsible for that person having access.

The key to SCoopFS supporting all six aspect of sharing is to base access decisions on explicit, delegatable authorizations [1]. Our approach, which we call authoriZation-Based Access Control (ZBAC), bundles the authorization for an access with the request itself. While subject authentication is one way to decide what rights to grant when using ZBAC, no examination of user authentication credentials is required at the time of access.

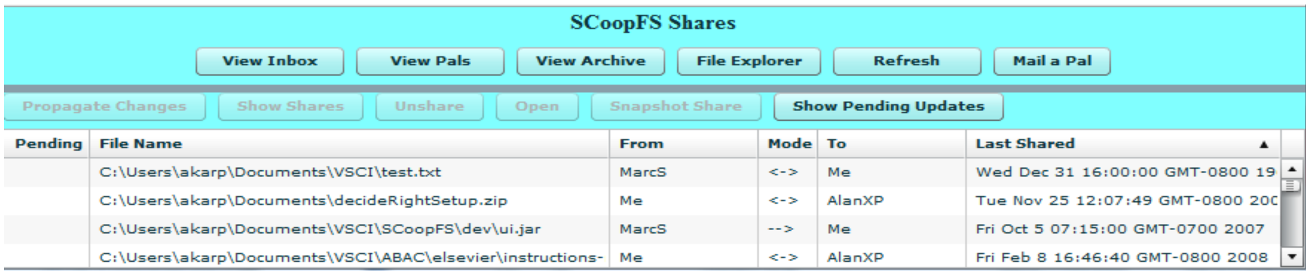
One advantage of such authorization-carrying access requests is that the user interface can represent each authorization in the application context so that the user need not be aware of the underlying security mechanisms. Manipulations in the user interface can then be mapped directly to access control decisions. This approach extends the fundamental property of capabilities, combining designation with authorization [2], to the user interaction, inferring authorization from designation [3].

3. ZBAC WITH WEBKEYS

SCoopFS uses ZBAC in the form of webkeys [4], which are designed specifically to operate across the Web. A webkey, *e.g.*,

https://y-yyx3b54gke3qg2dd.hp.com/#4ca26jq2quiugd

is a standard URL constructed to be used as a capability [2]. The webkey is an HTTPS URL, which protects both data and the URL while in transit. The domain name contains the fingerprint of the public key used for the TLS/SSL connection, allowing webkey-aware servers and browsers to authenticate the server, eliminating concerns about DNS redirection and man-in-the-middle attacks. Other browsers can still use these URLs, though without this security guarantee. The page name contains an unguessably large random string. Knowledge of the URL is proof that a request to the object it designates is authorized.



Pending	File Name	From	Mode	To	Last Shared ▲
	C:\Users\akarp\Documents\VSCI\test.txt	MarcS	<->	Me	Wed Dec 31 16:00:00 GMT-0800 19
	C:\Users\akarp\Documents\decideRightSetup.zip	Me	<->	AlanXP	Tue Nov 25 12:07:49 GMT-0800 20C
	C:\Users\akarp\Documents\VSCI\SCoopFS\dev\ui.jar	MarcS	-->	Me	Fri Oct 5 07:15:00 GMT-0700 2007
	C:\Users\akarp\Documents\VSCI\ABAC\elsevier\instructions-	Me	<->	AlanXP	Fri Feb 8 16:46:40 GMT-0800 2008

Webkeys have the properties needed to support the six aspects of sharing. Webkeys are shared as text strings, which makes their use dynamic and chained, while using the *caretaker pattern* [5] allows delegating fine-grain, temporary authority to a resource. Since webkeys are HTTPS URLs, references cross domains without changing firewall settings. Webkeys can be passed as parameters to methods invoked with webkeys, making them composable. A webkey can be constructed to grant access to a subset of the methods of an object, making them attenuatable, and granting a unique webkey for each sharing allows accountability.

4. USER INTERFACE

The SCoopFS user interface consists of a number of views [6], none of which has any buttons or fields specifically related to security. The Figure shows one of these views. The section above the gray bar lists the other views. The section below the gray bar is used to manage sharing relationships. Each filename is associated with the webkey of its synchronizer object. The From and To fields list the Pals involved in this sharing. Each Pal is associated with the webkey granting permission to send messages.

Two items in this view may be considered security related, but they are presented in terms related to the user's work flow. First, the arrows in the Mode column denote whether or not the sharer will accept updates from the sharee. That important information is often lost when sharing documents as email attachments. Second, is the Unshare button, which is activated when the user selects a row in the table. Unsharing ends the sending (receiving) of updates to (from) this Pal. While that action has security implications, it is a natural part of the user's activities.

5. SCORECARD

For this poster we score SCoopFS on 10 design guidelines for secure interaction design [7]. The reasoning behind the scores can be found in our tech report [6].

1. *Make the easy way the least authority way: 0.5*
2. *Use user actions to grant authorities*: 1.0
3. *Let the user to reduce granted authorities: 1.0*
4. *Show the user other's authorities to the user's resources*: 1.0
5. *Make the user aware of the user's own authorities*: 1.0
6. *Protect the how the user manipulates authorities*: 0.8
7. *Express policy in terms relevant to the user's task*: 1.0
8. *Distinguish objects and actions per the user's task.*: 0.5
9. *Clearly distinguish different objects and actions*: 1.0
10. *Make the implications of the user's actions clear*: 0.8:

These are common sense guidelines, yet it is hard to think of a single system in common use that implements even one of them. The previous highest score went to CapDesk [3], which gets a score of 7. Although we are happy with our score of 8.6 out of 10, there are improvements we can make in a future version.

6. CONCLUSIONS

We started the SCoopFS project with three goals in mind. The first was to produce a useful tool. We find that SCoopFS simplifies our work, but we're biased. The second goal was to demonstrate that security need not interfere with the user experience. Telling is the quote from one of our early users. "This tool would be a lot better if it had some security. Is there any way I can turn some on?" While that question shows we achieved our goal, it also indicates that achieving our goal is not enough. While "security reality" is necessary, the "feeling of security" is important, too [8]. We need to find a way to make people feel secure without making security interfere with their work

The success in meeting our third goal of demonstrating that it is easier to manage rights at a fine granularity than in big chunks is hard to quantify. We believe that we've shown that it is easy for users to manage rights to individual files. We also believe that existing systems for managing rights in big chunks, such as NFS and shared drives in Windows, require a lot of work on the part of the user or an administrator to avoid serious violations of Least Privilege. Quantifying the difference will be a challenge.

References

[1]. *Authorization Based Access Control for the Services Oriented Architecture.* **Karp, Alan H.** Berkeley, CA : IEEE Press, 2006. Proc. 4th Int. Conf. on Creating, Connecting and Collaborating through Computing (C5 2006).

[2]. *Programming Semantics for Miltiprogrammed Computations.* **Dennis, Jack B. and Van Horn, Earl C.** 3, s.l. : ACM, March 1966, Comm. ACM, Vol. 9, pp. 143-155.

[3]. **Stiegler, Marc D.** E and CapDesk: POLA for the Distributed Desktop. [Online] 2002. [Cited: December 16, 2008.] http://wiki.erights.org/wiki/CapDesk.

[4]. *web-key: Mashing with Permission.* **Close, Tyler.** Oakland : IEEE, 2008. IEEE W2SP 2008: Web 2.0 Security and Privacy.

[5]. **Redell, D. D.** *Naming and Protection in Extendible Operating Systems.* Project MAC TR-140. s.l. : MIT, 1974. Ph. D. Thesis.

[6]. **Karp, Alan H., Stiegler, Marc and Close, Tyler.** *Not One Click for Security.* Palo Alto, CA : Hewlett-Packard, 2009. HPL-2009-53. http://www.hpl.hp.com/techreports/2009/HPL-2009-53.html.

[7]. **Yee, Ka-Ping.** Guidelines and Strategies for Secure Interaction Design. [book auth.] Lorrie Faith Cranor and Simson Garfinkel. *Security and Usability: Designing Secure Systems That People Can Use.* Sebastopol, CA : O'Reilly Media, Inc., 2005, pp. 247-273.

[8]. *Reconceptualizing Security.* **Schneier, Bruce.** San Diego : Usenix, 2008. LISA '08: 22nd Large Installation System Administration Conf.