More than Skin Deep: Measuring Effects of the Underlying Model on Access-Control System Usability

Robert W. Reeder
Microsoft
Redmond, WA
roreeder@microsoft.com

Lujo Bauer
Carnegie Mellon University
Pittsburgh, PA
lbauer@cmu.edu

Lorrie Faith Cranor
Carnegie Mellon University
Pittsburgh, PA
lorrie@cmu.edu

Michael K. Reiter
University of North Carolina, Chapel Hill
Chapel Hill, NC
reiter@cs.unc.edu

Kami Vaniea
Carnegie Mellon University
Pittsburgh, PA
kami@cmu.edu

ABSTRACT

In access-control systems, policy rules *conflict* when they prescribe different decisions (ALLOW or DENY) for the same access. We present the results of a user study that demonstrates the significant impact of conflict-resolution method on policy-authoring usability. In our study of 54 participants, varying the conflict-resolution method yielded statistically significant differences in accuracy in five of the six tasks we tested, including differences in accuracy rates of up to 78%. Our results suggest that a conflict-resolution method favoring rules of smaller scope over rules of larger scope is more usable than the Microsoft Windows operating system's method of favoring deny rules over allow rules. Perhaps more importantly, our results demonstrate that even seemingly small changes to a system's semantics can fundamentally affect the system's usability in ways that are beyond the power of user interfaces to correct.

Author Keywords

access control, security, human factors

ACM Classification Keywords

H.1.2 User/Machine Systems: Human factors; D.4.6 Security and Protection: Access controls; H.5.2 User Interfaces: User-centered design

General Terms

Experimentation, Human Factors, Security

INTRODUCTION

Access-control policies must be specified correctly to ensure that authorized access is allowed while unauthorized access is denied. One obstacle to accurate access-control policies is human error; the people who author and maintain these policies—whom we call "authors" or "policy authors"—are prone to making specification errors that lead to incorrect policies [8, 16, 21].

Rule conflicts are one important area of difficulty for policy authors. Access-control policies consist of a set of rules that dictate the conditions under which users will be allowed access to resources. These rules may conflict with each other. For example, one rule may allow user u access to file f, while a second rule may deny group g, of which u is a member, access to f. Past work has shown that authors have difficulty detecting and resolving rule conflicts [16].

Usability improvements to security systems, such as access-control systems, can come from two sources: improvements to user interfaces and changes to underlying system models. User interfaces for policy-authoring systems have already received a fair amount of attention from researchers [3, 9, 14, 16, 18, 23], so we set out to improve usability in an access-control policy authoring system by varying an aspect of the underlying access-control model. Specifically, we studied the conflict-resolution method, which is the algorithm an access-control system uses for determining which rule, of a set of rules in conflict, will take precedence over others. We describe a conflict-resolution method that we expect will lead to fewer specification errors. Our method is aimed particularly at improving upon the Windows NTFS conflict-resolution method, since Windows is such a widely deployed operating system, and since it has been shown to be prone to policy-specification problems [16, 21].

We implemented our conflict-resolution method and the Windows method in a simulated Windows NTFS file system. For a policy-authoring user interface across both conflict-resolution method conditions, we used the *Expandable Grid*, which has been shown in past work to be more usable than the equivalent Windows interface [18]. An Expandable Grid (see Figure 2) is an interactive visualization that approximates Lampson's access-control matrix [15]. It displays *effective policy* resulting from an underlying set of rules and permits authors to modify effective policy directly.

CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.

We evaluated our conflict-resolution method by running a user study in which participants performed a variety of tasks using one of three different combinations of interface and conflict-resolution method: Grid with Windows conflict-resolution method, Grid with our conflict-resolution method, and Windows interface with Windows conflict-resolution method. Since two of our conditions use the same interface, we can compare the effect of just the conflict-resolution method on usability, with interface held constant. Our results show that the choice of conflict-resolution method can have a profound effect on usability. Whichever conflict-resolution method selects the intended rule for a given task by default (and thus requires no action from the policy author) is the more usable for that task. Moreover, we demonstrate that our conflict-resolution method is far more usable for tasks that require the author to take action — leading to gains of up to 78% in accuracy rates. (Of course, many considerations go into choosing an access-control model for a given usage context, so our method, while better for usability, may not be best for all contexts.)

Our results also illustrate a larger point about designing usable security systems. Designing security systems has traditionally been the work of expert security architects versed in technical arcana like access control and cryptography. The work of making a security system usable is typically pushed to usability experts after the work of the security architects is done, so that a suitable user interface can make the underlying security model usable by humans. Some researchers have suggested this "security-then-usability" approach is suboptimal; they suggest "user-centered security" in which usability is considered early on, as part of the process of designing the security model as well as the interface [1, 13, 24]. Following the logic of user-centered security, the ultimate usability of a system should be affected by both underlying model and interface. Our results provide empirical evidence that this proposition is true, at least for access-control models, and that leaving usability as an end-of-cycle, user-interface-only solution is misguided. In fact, security-model decisions made early in the development lifecycle *can* have a substantial impact on usability, even when those decisions at first appear to bear no relation to usability. We show that while the user interface is clearly an important part of a system's usability, underlying models also matter a great deal.

PROBLEM DESCRIPTION

Our objective is to find a conflict-resolution method that helps authors accurately set the policies they intend. Specifically, we want a conflict-resolution method that helps authors set policies accurately in the presence of rule conflicts.

We define an access-control policy to consist of rules, which are tuples of the form (*principal*, *resource*, *action*, *decision*). Principals may be users or groups containing users; in Windows, groups cannot contain other groups, but different groups may have overlapping memberships. Resources may be files or folders; resources are arranged in a strict hierarchy, so folders cannot have overlapping memberships. READ and WRITE are examples of actions. Decisions can be ALLOW or DENY. *Requests* to an access-control system are tuples of the form (*principal*, *resource*, *action*). A rule *matches* a request if the rule's principal, resource and action contain (or equal) the request's principal, resource, and action, respectively. Rules that match a request *conflict* if they prescribe different decisions for that request. When conflicting rules apply to a request, the conflict-resolution method comes into play to resolve the conflict. The resulting mappings of requests to decisions are known as *effective permissions*.

Conflict-resolution methods

To resolve rule conflicts, there must be a method for unambiguously choosing a decision. Most conflict-resolution methods in practice choose one of the rules in conflict to take precedence over the others. (Other methods are possible, however, such as "majority rules.") Note that it is sufficient to define these methods in terms of their behavior when exactly two rules are in conflict, because the access control system can handle cases of more than two rules in conflict by following a simple algorithm that does paired matches of each ALLOW rule against each DENY rule. This algorithm issues an ALLOW decision if any ALLOW rule wins its matches against every DENY rule, and otherwise issues a DENY decision.

While numerous methods for choosing rules to take precedence are possible, two are relevant to this paper:

- *Specificity precedence*: A rule that applies to a more specific entity (principal or resource) takes precedence over a rule that applies to a more general entity. For example, a rule that applies to a user takes precedence over a rule that applies to a group, or a rule that applies to a subfolder or file takes precedence over a rule that applies to the subfolder's or file's parent.
- *Deny precedence*: DENY rules take precedence over ALLOW rules.

These conflict-resolution methods may be used in combination. It is possible to use different conflict-resolution methods depending on whether conflicting rules differ in the principals they cover, the resources they cover, or both. It may also be necessary to resort to multiple conflict-resolution methods when one method fails to resolve a conflict. For example, when conflicting rules cover groups, but those groups are peers of each other (i.e., neither contains the other), specificity precedence cannot resolve the conflict.

Weaknesses of the Windows NTFS method

The Windows NTFS conflict-resolution method combines specificity and deny precedence. When conflicting rules differ only in their resources, specificity precedence is used. When conflicting rules differ only in their principals, deny precedence is used. When conflicting rules differ in both principals and resources, specificity precedence is used.

The Windows conflict-resolution method has two primary weaknesses we seek to improve upon. First, the use of deny precedence not only leads to specification errors, it also makes certain policy configurations impossible. Second, the Win-

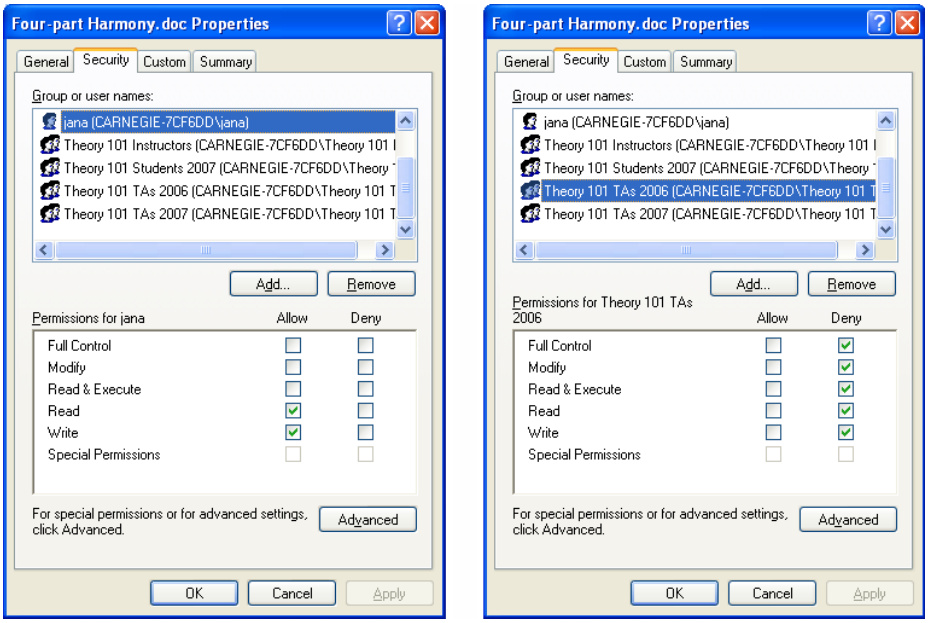

Figure 1. The Windows XP file permissions interface. The left-hand screenshot shows rules applying to Jana that appear to state that Jana is allowed to read and write the operative file, but the right-hand screenshot shows rules applying to a group of which Jana is a member that conflict with Jana's rules. If a policy author does not know that Jana is in the group and that the DENY rules take precedence, the author may have difficulty determining the effective permissions in the presence of a rule conflict.

dows behavior in the presence of a *two-dimensional conflict*, in which one rule is more specific in its principal and another rule is more specific in its resource, is likely to confuse some policy authors' expectations. We discuss each of these weaknesses below.

Deny precedence

By using deny precedence, the Windows conflict-resolution method leads to violations of *direct manipulation*, a user-interface design principle stating that interfaces should allow users to operate directly on objects or data of interest [20]. In policy-authoring interfaces, direct manipulation translates to allowing authors to change effective policy directly whenever possible.

The Jana task, one of the tasks we assigned to participants in our user study (see the "User study method" section), is a good example of where the Windows conflict-resolution method leads to violations of direct manipulation. The goal of the task is to allow Jana read and write access to a file. Jana is initially a member of two groups, one of which is allowed access to the file and the other of which is denied access to that file. Windows uses deny precedence to resolve the conflict, so Jana is effectively denied access to the file. However, many authors do not understand this; they may see only the rule allowing Jana access, and not realize that another rule conflicts with it. Even if they do notice the conflict, it is not easily resolved. There are two ways to grant Jana access: the DENY rule applying to the latter group can be removed, or Jana can be removed from the group. Direct manipulation is violated because the task cannot be completed by simply manipulating rules applying to Jana; the task requires making a change at the group level. Moreover, both potential solutions may lead to undesirable side effects. Removing the DENY rule may change policy for the other members of the group and for members added to the group in the future. Removing Jana from the group may be undesirable since it may affect Jana's permissions on other resources. The Windows conflict-resolution method provides no entirely satisfactory way to complete the Jana task.

Figure 1 shows the Windows file permissions user interface. The screenshot on the left shows the approach many participants in our study took to try to resolve the conflict; they set a rule applying specifically to Jana indicating that she can read and write the file. However, under the Windows conflict-resolution method, Jana is still denied access because of the group rule, shown in the right-hand screenshot of Figure 1, that denies access to the file. Because it is difficult to view effective permissions in the Windows interface (the display of effective permissions is buried three mouse clicks away), many authors assumed they had correctly completed the task when in fact they had not.

The Expandable Grid interface, shown in Figure 2, was designed to show effective permissions prominently and to allow the author to directly manipulate them. In the Expandable Grid screenshot shown in the figure, the cells at the intersection of "jana" and the "Four-part harmony.doc" file are red, indicating that Jana cannot read or write the file. To stay faithful to the principle of direct manipulation, an author should be able to click on the red squares to allow Jana to access the file. The author should see the squares turn green, which indicates an effective permission of ALLOW. Unfortunately, under the Windows conflict-resolution method, direct manipulation is still impossible, even in the Grid inter-

face. Clicking on the red squares can set ALLOW rules that state that Jana can read and write the file, but the rules will not take effect until the conflicting group DENY rule is removed, or Jana is removed from the group. Even when the user clicks on them, the red squares stay red. The conflict-resolution method we propose in the "Proposed solution" section enables direct manipulation of effective permissions in the Grid interface for the Jana task.

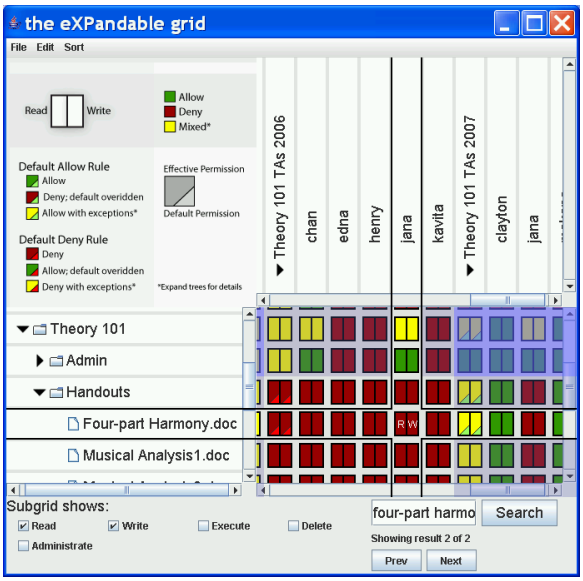

Figure 2. Our Expandable Grid interface for setting file permissions in Windows XP. The interface shows principals along the upper axis, resources along the left-hand axis, and the effective permissions applying to the principals and resources in the colored squares in the grid itself.

Two-dimensional conflicts

Besides violations of direct manipulation in the presence of rule conflicts, we seek to improve upon a second aspect of the Windows conflict-resolution method: behavior in the presence of a two-dimensional conflict. A *two-dimensional conflict* occurs when two rules are in conflict and one rule is more specific in the principal dimension (i.e., the dimension of users and groups) while the other is more specific in the resource dimension. For example, in the Lance task we describe in the "User study method" section, one rule denies Lance access to an Admin folder, but another rule allows a group he is in access to the gradebook.xls file contained in the Admin folder. Since neither rule is strictly more specific than the other, specificity precedence cannot resolve this conflict. Windows resolves such conflicts by favoring the resource dimension over the principal dimension, so the rule that is more specific in the resource dimension will take precedence. Since Windows uses deny precedence to resolve other conflicts, some authors may expect deny precedence to also resolve two-dimensional conflicts, so this aspect of the Windows conflict-resolution method is likely to be inconsistent with some authors' expectations. Moreover, some authors may expect the principal dimension to be favored, rather than the resource dimension.

PROPOSED SOLUTION

We propose a conflict-resolution method that we believe will address the weaknesses of the Windows conflict-resolution method. We propose to use specificity precedence, in both principals and resources, to resolve conflicts when possible and to resort to deny precedence only when specificity precedence fails. By using specificity precedence, we address the rule-conflict weakness in Windows in which the conflict-resolution method violates direct manipulation. Specificity ensures support for direct manipulation for rules that cover users and files. Thus, tasks like the Jana task are easily completed using the Grid with our specificity-based conflict-resolution method; clicking Jana's red squares turns them green. The rule-conflict weakness in Windows where either a group access rule has to be changed or Jana has to be removed from the group is solved by allowing a specific rule that applies to Jana to take effect.

To address the weakness of the Windows conflict-resolution method in the case of a two-dimensional conflict, our proposed method uses deny precedence for two-dimensional conflicts. Since there is no natural way to resolve such conflicts using specificity, we ensure fail safety by using deny precedence. We believe the Windows use of specificity in the resource dimension is likely to be confusing to policy authors, who may expect that deny precedence would apply in these situations. Our choice of deny precedence may or may not be confusing, but at least it is safe and, in any case, it is easily overridden by creating a rule that is equally or more specific in both resources and principals than the rules in conflict. Note that while we resort to deny precedence in two-dimensional conflicts, the fact that our method uses specificity precedence first means that overriding the two-dimensional conflict with a more specific rule is easy.

Table 1 shows the differences between our specificity-based conflict-resolution method and the Windows method. It lays out the entire space of rule conflicts with respect to the structural relations between principals and resources of the rules in conflict. In a conflict, there will be an ALLOW rule in conflict with a DENY rule. The static differences between our method and the Windows method can be seen in two of the table cells: first, where the ALLOW rule's principal is contained by (and thus is more specific than) the DENY rule's principal, but the two rules' resources are the same; second, where the ALLOW rule's principal contains the DENY rule's principal, but the ALLOW rule's resource is contained by the DENY rule's resource. However, the most significant difference between the two methods is in some of the conflicts where both methods give a DENY decision, but the author intends an ALLOW decision. Our specificity method makes these cases easier to fix than does the Windows method.

From a usability perspective, a good conflict-resolution method is one that gives the decision the author intends, or, if it cannot give the intended decision by default, makes it easy to change the policy to get the right decision. No method can always make the intended decision by default, since it cannot know the author's intention. Thus, it is on the latter point—the ease of changing the policy—that we expect our

Table 1. Table showing which rule, of an ALLOW rule and a DENY rule in conflict, will take precedence for our specificity-based conflict resolution method (S) and the Windows method (W). The table shows the relevant cases defined by the rules' principals and resources. Each cell shows which rule takes precedence and the conflict-resolution method in play: specificity in the *resources*, specificity in the *principals*, specificity in *both* resources and principals, or *deny* precedence. In the case where both rules' resources and principals are the same, there will be *no conflict*, since the more recently set rule will have overwritten the other.

		ALLOW rule's principal is ____ DENY rule's principal			
		Less specific than	Peer of	Same as	More specific than
ALLOW rule's resource is ____ DENY rule's resource	Less specific than	DENY (both)	DENY (resources)	DENY (resources)	S:DENY (deny) W:DENY (resources)
	Same as	S:DENY (principals) W:DENY (deny)	DENY (deny)	no conflict	S:ALLOW (principals) W:DENY (deny)
	More specific than	S:DENY (deny) W:ALLOW (resources)	ALLOW (resources)	ALLOW (resources)	ALLOW (both)

method to have the greatest usability gains over the Windows method. To resolve a conflict for which the conflict-resolution method is not already giving the intended decision by default, an author must (1) notice the problem; and (2) fix it. The Expandable Grid, by virtue of showing effective permissions directly, makes noticing the problem much easier than does the Windows interface, and our specificity conflict-resolution method makes fixing the problem easier by allowing user-level exceptions to group rules, instead of requiring that a user be removed from a group or requiring an entire group's permissions to change. In the Jana task, for example, both methods will result in a DENY decision, but specificity allows an author to specify an exception for Jana to the group-level DENY decision, while Windows simply does not allow such an exception, and instead forces the author to make a group-level policy change.

USER STUDY METHOD

We have argued that our conflict-resolution method should be more usable than the Windows method for tasks that require action from the policy author because our method enables direct manipulation in the Grid interface. To test this argument empirically, we ran a laboratory user study to measure the effects of conflict-resolution method on policy-authoring usability, as measured by task-completion accuracy. Our study had 54 participants, who each performed 12 policy-authoring tasks in one of three experimental conditions. Experimental conditions were combinations of interface and conflict-resolution method. We used a between-participants design, so 18 participants were in each of the three conditions. We measured accuracy for each task. The three conditions we compared were:

- the Grid interface with our specificity-based conflict resolution method, a combination henceforth called *GS*;
- the Grid interface with Windows conflict-resolution method, a combination henceforth called *GW*; and
- the Windows interface with the Windows conflict-resolution method, a combination henceforth called *WW*.

The WW condition served as the control in our study; it was the condition on which we hoped to improve. The GS condition is our best effort at improving upon WW, because GS has both the visual presentation advantages of the Expandable Grid and the expected advantages of our conflict-resolution method. The GW interface serves to help us separate the effects of the Grid as a presentation technique from the effects of the conflict-resolution method. When we observe differences between the GW and WW conditions, we attribute them to the Grid presentation (i.e., the user interface), and when we observe differences between the GW and GS conditions, we attribute them to the conflict-resolution method (i.e., the underlying model). Differences between the GS and WW conditions are the cumulative effect of the Grid presentation and the specificity-based conflict-resolution method. (Note that, while results from a fourth combination of the Windows interface with the specificity conflict-resolution method would have been enlightening, it was infeasible to implement such a combination for this study.)

We recruited 54 undergraduate and graduate students from technical disciplines (science, engineering, or mathematics) to participate in the study. Eighteen were female. Participants were all daily computer users, but had never served as system administrators. As in prior work evaluating the Expandable Grid interface [18], our participant pool was consistent with a target demographic of novice and occasional policy authors. The policy authoring population largely consists of information technology professionals with some technical education who have so many job responsibilities that access-control is an occasional activity. Other populations, like managers and staff, author policy but were out of scope for this study.

Task design

Of the 12 tasks in which each subject participated, six were designed to test the advantages and disadvantages of each of the two conflict-resolution methods, as well as the overall usability of each interface/conflict-resolution-method combination. We discuss only these six tasks here. (The other six tasks are fully reported elsewhere [17].) All tasks were based on a teaching assistant (TA) scenario, in which the participant is put in the role of a TA maintaining the file server for a hypothetical music department. The hypothetical file server contained roughly 500 principals and 500 resources. Each task is defined by its task statement (i.e., the text we presented to participants in the study) and its initial configuration, including existing access rules, group memberships, and file locations. Task statements asked participants to make changes to the initial configuration.

The tasks were designed in pairs: the Charles/Kent pair, the Lance/Adria pair, and the Jana/Pablo pair. For the Charles/Kent and Lance/Adria pairs, the tasks are structured similarly except that goals are inverted. This way, each pair of tasks has a task that favors our conflict-resolution method and a task that favors the Windows conflict-resolution method. The Jana and Pablo tasks share essentially the same structure, though they differ superficially in the specific users and files involved. We used two tasks with the same structure because this structure best illustrates the advantage of specificity precedence, and we wanted to show that any usability differences between conflict-resolution methods were due to the underlying task structure and not the superficial aspects of the task. Also, pairing the Jana and Pablo tasks gave us a fair balance of tasks: two tasks in which our conflict-resolution method completes the task goal by default (Charles and Lance), two tasks in which the Windows method completes the task goal by default (Kent and Adria), and two tasks in which neither method completes the goal by default and the author has to take action (Jana and Pablo).

The six tasks can be classified by their initial configuration. Each task's initial configuration included a rule conflict that fit into one of the rule-conflict structures shown in Table 1. The tasks only cover the three table cells corresponding to interesting rule conflict structures, i.e., those for which the two conflict-resolution methods yield different decisions or for which they lend themselves to different means of resolving the conflict. Specifically, the Charles and Kent tasks present cases where the ALLOW rule's principal is more specific than the DENY rule's principal and the resources are the same. The Lance and Adria tasks present 2-dimensional conflicts, where the DENY rule's principal is more specific, but the ALLOW rule's resource is more specific. The Jana and Pablo tasks present cases where the principals are peers of each other and the resources are the same. We describe each task pair in more detail below.

Charles/Kent

We designed the Charles task to show the advantage a specificity conflict-resolution method has over the Windows method when ALLOW rule exceptions to a group DENY rule are desired. The Charles task involves adding a user to a group; the user has several ALLOW permissions on some files, the group has DENY permissions on those files, and the goal is to keep the user's ALLOW permissions. The Charles task statement presented to participants was:

> Charles has just graduated, but is going to come back to sing in the choir with his friends.
>
> Add **Charles** to the **Alumni** group, but make sure he can still **read** the same files in the **Choir 1\Lyrics** folder that his good friend Carl can read.

In the initial configuration, there are rules stating that Charles is allowed READ access to four files in the Choir 1\Lyrics folder. These are the same files that Carl can read, so in the final state, we want Charles to be allowed READ access to the four files. There are rules stating that the Alumni group is denied READ access to the same files. When Charles is moved into the Alumni group, the group's DENY rules will apply to him, and under the Windows deny precedence, he will be denied access to the files. However, under specificity precedence, his rules are more specific than the group access rules, so he will still be allowed to read the files. Thus, specificity precedence makes this task easier, and we expected participants in the GS condition to perform better than GW and WW in the Charles task.

We designed the Kent task to show a drawback our specificity-based conflict-resolution method has when ALLOW exceptions to a group DENY rule are not desired. The Kent task was structured similarly to the Charles task, but the goal was inverted to give the advantage to the Windows conflict resolution method.

The Kent task statement presented to participants was:

> Kent was a terrible TA for Choir 1 so the instructor demoted him to the level of student. While Kent previously had permissions to read and write the attendance and gradebook files, as a student he should no longer have access to that information.
>
> Remove **Kent** from **Choir 1 TAs 2008** and add him to **Choir 1 Students 2008**. For files in the **Classes\Choir 1\Admin** folder, make sure he only has the same permissions as the other Choir 1 students.

As in the Charles task, Kent inherits a DENY permission from a group to which he is moved. This time, however, Kent is to keep the DENY permission, so we expect participants in the GW and WW conditions to perform better on the Kent task.

Lance/Adria

We designed the Lance task to test the behavior of our conflict-resolution method in the presence of a two-dimensional conflict when a DENY decision is desired. The Lance task introduces a two-dimensional conflict, in which there are conflicts in both the principals and resources. The Lance task statement presented to participants was:

> Lance was hired by the New York Philharmonic and can no longer serve as Head TA this year.
>
> Remove **Lance** from the group **Head TAs 2008**, but make sure you don't remove him from **Head TAs 2007**. Then make sure he is not allowed to access any files in the **Music 101\Admin** folder.

In the initial configuration, there is a rule stating that Lance is denied READ access to the Music 101\Admin folder, and there are rules stating that the Head TAs 2007 group is allowed READ access to the Music 101\Admin\gradebook.xls file, but that the Head TAs 2008 group is denied READ access to the file. When Lance is removed from Head TAs 2008, a two-dimensional conflict is revealed, since the rule applying to Lance on the Admin folder is more specific in its principal, but the rule applying to Head TAs 2007 on the gradebook.xls file is more specific in its resource. In the Windows conflict-resolution method, the rule that is more

specific in the resource takes precedence, so Lance will be allowed to read the gradebook.xls file, but in the specificity-based conflict-resolution method, the DENY rule takes precedence, so Lance will not be allowed to read the gradebook.xls file. Since the task calls for Lance not to be allowed to access any files in the Admin folder, our conflict-resolution method requires less work to complete the task correctly, and we expected participants in the GS condition to perform best.

We designed the Adria task to test the behavior of our conflict-resolution method in the presence of a two-dimensional conflict when an ALLOW decision is desired. Just as the Kent task is similar to the Charles task with the inverse goal, the Adria task is similar to the Lance task with the inverse goal.

The Adria task statement presented to participants was:

> Adria, an Opera Instructor, was not getting along with the other instructor and left the class. You need to remove her from the Opera Instructors group. She is still a Music 101 instructor, though, and the Music 101 instructors need access to the Music 101 Lecture Notes.
>
> Remove **Adria** from the **Opera Instructors** group. Make sure she has the same permissions on the files in the **Music 101\Lecture Notes** folder as the other Music 101 instructors.

As in the Lance task, when Adria is removed from a group, a two-dimensional conflict involving another group is revealed. However, the goal is inverted in the Adria task, so that now, we expect the advantage to lie with the GW and WW conditions.

Jana/Pablo

The Jana task involves a rule conflict at the group level that must be resolved to give Jana access to a file. The Jana task statement presented to participants was:

> Jana, a Theory 101 TA, complained that when she tried to change the Four-part Harmony handout to update the assignment, she was denied access.
>
> Set permissions so that **Jana** can **read and write** the **Four-part Harmony.doc** file in the Theory 101\Handouts folder.

We expect specificity precedence to enable authors to perform better for the Jana task compared to deny precedence. Thus, we expected participants in the GS condition to outperform participants in the GW and WW conditions for the Jana task.

The Pablo task was structured similarly to the Jana task, and we expected better performance in the GS condition compared to the GW and WW conditions. The Pablo task statement presented to participants was:

> Pablo, a student in Music 101, tried to download the homework file, assignment4.pdf, but couldn't.
>
> Set permissions so that **Pablo** can **read** the file **assignment4.pdf** in the **Music 101\Handouts** folder. Make sure you don't change any other students' permissions. (Hint: If you need to, you can add Pablo to a new group or remove Pablo from a group he's already in.)

As previously discussed, the Pablo task was superficially different from Jana in the names and numbers of principals and resources involved in the task, but structurally the same.

Procedure

At the start of each study session, participants filled out a demographic survey so that we could ensure they were students in technical disciplines. Following the survey, our experimenter read instructions explaining our teaching-assistant scenario to participants. After reading these instructions, our experimenter read interface training materials. For each interface, training covered how to perform the following operations: viewing files and folders; moving a file; viewing group memberships; adding a user to a group; removing a user from a group; creating a new group; checking an access rule; checking effective permissions; creating an access rule; and searching for a file or principal. During training, the experimenter also explained that effective permissions may differ from access rules because of the way rules combine, but did not explain the precise workings of the conflict-resolution methods. After these operations had been explained to participants, the experimenter walked them through a training task. The training task gave participants practice with some of the basic operations covered during training, but did not involve a rule conflict. Participants received the same training task in all three experimental conditions. Training took about 10 minutes.

Participants then began completing tasks. Before each task, the experimenter brought up the interface in a preconfigured state tailored to each task. Task statements were then presented to participants in a Web browser on screen. Participants indicated they were done with each task by clicking a button in the Web browser. Participants were asked to think aloud while they worked on the tasks. Task order was counterbalanced across participants using a pseudo-random Latin square design to guard against ordering and sequence effects.

RESULTS

Our results are in the form of accuracy rates for each condition and task. While we also measured times-to-task-completion, we do not present them in detail due to space constraints. Time-to-task-completion data resulted in few statistically significant differences between conditions and the results that were significant corroborate the accuracy results, so the data show that the accuracy results we present here were not merely due to a time-accuracy tradeoff. Complete time-to-task-completion data can be found elsewhere [17].

Accuracy rates represent the proportion of participants in each condition who correctly completed the task. We scored a task as correct if the participant's final effective permissions matched the task goals and did not introduce any extraneous changes not called for by the task statement or otherwise required to complete the task. Tasks in which the

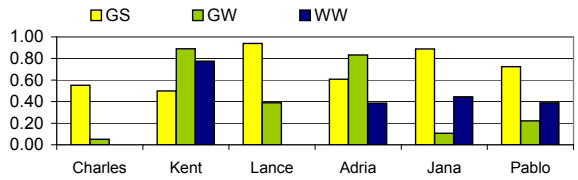

Figure 3. Accuracy results, showing proportion of participants correctly completing each task with GS, GW, and WW interfaces.

goals of the task were not met or in which the goals were met but extraneous changes were introduced were scored as incorrect. Accuracy results can be seen in Figure 3.

We conducted post-hoc task-by-task tests. Each of our tasks tested a specific hypothesis about the effects the conflict-resolution methods would have on usability, as measured by accuracy. We tested hypotheses regarding accuracy rates by using one-sided Fisher's exact tests with the null hypothesis that the difference in accuracy rates was zero. We used one-sided tests since our hypotheses are directional (i.e., we expect better performance from one condition than another). Results of all hypothesis tests of accuracy rates can be seen in Table 2.

In the Charles, Kent, Lance, and Adria tasks, one of the two conflict-resolution methods yielded the goal decision by default, so had an advantage over the other conflict-resolution method in helping participants complete tasks correctly. Thus, for those four tasks, either our conflict-resolution method or the Windows method had a natural advantage, and we expected the conditions using the advantaged conflict-resolution method in each task to lead to better performance. For the Charles and Lance tasks, in which our specificity-based conflict-resolution method had the advantage, the accuracy rate for the GS condition was statistically significantly higher than that for the GW and WW conditions, as expected. For the Kent task, in which the Windows conflict-resolution method had the advantage, the accuracy rate for the GS condition was statistically significantly lower than that for the GW condition, as expected, but the result for GS compared to WW was not statistically significant. For the Adria task, in which the Windows method had the advantage, there was no statistically significant difference between the GS condition and the GW and WW conditions. This somewhat surprising result in favor of GS is likely due to the combination of the Grid's presentation aspects, which allow authors to easily notice the discrepancy between the effective policy and their goal policy, and the ease of adding a specific rule applying to Adria to overcome the conflict with specificity precedence.

For the Jana and Pablo tasks, both conflict-resolution methods yield a DENY decision in the presence of the conflict, so we did not expect either to have an advantage by default. However, we expected the specificity-based method to make it easy to overcome the conflict by setting a specific rule allowing Jana or Pablo to access the relevant files. As we have explained, the Windows method makes it cumbersome to overcome the conflict at the group level. As expected, the GS condition led to better performance over both the GW and WW conditions in both the Jana and Pablo tasks.

In summary, we observed GS to be statistically significantly better than both GW and WW in the four tasks in which GS had the default advantage or in which there was no default advantage. In the two tasks where the default advantage went the other way, GS was statistically significantly worse compared to GW in the Kent task, and was behind GW and WW in both tasks, though not significantly so. The comparisons between GS and GW illustrate the substantial usability gains that can be had just from our conflict-resolution method; even if the Charles/Kent and Lance/Adria accuracy rates are viewed as a wash, GS gave a gain of 78% in accuracy rate compared to GW for Jana and 50% for Pablo.

DISCUSSION

We discuss three aspects of our results: comparison of the two conflict-resolution methods, broader implications for designing usable security systems, and limitations of our study.

Comparing conflict-resolution methods

Our results show that the policy conflict-resolution method underlying a file-permissions interface has a significant effect on task-completion accuracy. The precise effect of the conflict-resolution method depends on what the task goals are. If task goals are aligned with a conflict-resolution method, such that the method yields the desired effective policy by default, accuracy rates are likely to be higher than if the method requires an author to take additional action to change the decision. In the Charles, Kent, and Lance tasks, we saw that the method that yielded the correct effective policy by default led to the highest accuracy rates.

No conflict-resolution method can always yield the desired effective permissions, as we designed our inverse task pairs (Charles/Kent, Lance/Adria) to show. However, we have argued that our specificity-based conflict-resolution method is superior to the Windows deny-based method because, even in a situation where specificity is not yielding the desired effective permission, it is easy to change the effective permission with a specific rule. Contrast this to the Windows deny-precedence method, which forces an author in some conflict situations to remove a user from a group or change rules applying to the whole group. The Adria task shows a situation in which, although our specificity-based conflict-resolution method does not yield the desired decision by default, getting the task right is a simple matter of adding a rule more specific than those in conflict. The Pablo and Jana tasks further illustrate the advantage that a specificity-based conflict-resolution method has over a deny-based method in overcoming rule conflicts. Thus, our argument that specificity is more usable when the author is required to take action to fix a rule conflict is borne out by our empirical results.

While our empirical results are based on holding only one user interface, the Expandable Grid, constant across conditions, we note that a different interface could not have negated the difference in usability between the two conflict-resolution methods. No user interface can overcome the advantage of making the right decision by default; doing so would require making it easier for the author to do something than to do nothing. And no user interface can make it

Table 2. Summary of post-hoc statistical tests for significant differences in accuracy rate for Grid with our specificity-based conflict-resolution method (a_{GS}), Grid with Windows conflict-resolution method (a_{GW}), and Windows (a_{WW}). For each task, the table shows accuracy rates for the three interfaces, hypotheses tested, and p-values from one-sided Fisher's exact tests; p-values at or below the $\alpha = 0.05$ rejection threshold are shaded and highlighted in bold, indicating significant tests.

Task	a_{GS}	a_{GW}	a_{WW}	GS vs. GW hypothesis	p-value	GS vs. WW hypothesis	p-value
Charles	0.56	0.06	0.00	$a_{GS} > a_{GW}$	**0.001**	$a_{GS} > a_{WW}$	**< 0.001**
Kent	0.50	0.88	0.78	$a_{GS} < a_{GW}$	**0.014**	$a_{GS} < a_{WW}$	0.082
Lance	0.94	0.39	0.00	$a_{GS} > a_{GW}$	**< 0.001**	$a_{GS} > a_{WW}$	**< 0.001**
Adria	0.61	0.83	0.39	$a_{GS} < a_{GW}$	0.13	$a_{GS} < a_{WW}$	0.95
Jana	0.89	0.11	0.44	$a_{GS} > a_{GW}$	**< 0.001**	$a_{GS} > a_{WW}$	**0.006**
Pablo	0.72	0.22	0.39	$a_{GS} > a_{GW}$	**0.003**	$a_{GS} > a_{WW}$	**0.046**

possible to do the Jana and Pablo tasks under the Windows semantics without making a group-level change.

One concern with adopting a specificity-based conflict-resolution method is that it introduces a risk that, when setting a group-level DENY rule, the DENY rule will not apply to members of the group who already have ALLOW rules applying to them. Specifically, in a situation like the one in the Kent task, where a group-level DENY rule is meant to override any user-level ALLOW rule exceptions, specificity does not give the desired decision by default. If a policy author fails to notice the undesired ALLOW exceptions to the DENY rule, unauthorized access may be allowed. However, the Expandable Grid interface can help mitigate this issue by making it much easier for authors to see anomalies in their policies [18]. Thus, if using such an interface, the usability gains from specificity are likely worth this slight security risk for many applications. Moreover, an unusable policy model may itself be a security risk, as humans frustrated that they are unable to implement the policies they want may set policies that are more liberal than necessary [2], or circumvent access-control systems altogether [11].

Implications for designing security systems

The substantial differences in accuracy rates due to relatively small changes to the access-control model in our study are striking. Differences in accuracy rates between the GS and GW conditions were 50%, 38%, 55%, 22%, 78%, and 50% for our six tasks. In five of six cases, these differences were statistically significant. Since we held interface constant between GS and GW and only varied conflict-resolution method, these substantial differences are due only to changes in the underlying model. In real-world security system design, these kinds of details of underlying models are usually determined by security experts, who may not consider their effects on usability. Our results suggest they should, since the effects can be large, and since, as we have just pointed out, an unusable system that users circumvent is itself a security risk. Other researchers have called for "user-centered security," in which usability is a prime consideration during the design of underlying models and in which security and usability experts work side-by-side [1, 13, 24]. Our results lend empirical evidence to back up these calls for user-centered security.

Limitations

Some limitations of our study are worth noting. Our results tell us that the success of any conflict-resolution method depends largely on the requirements of the particular tasks that an author performs. Since we do not have data on the frequency with which particular tasks are performed, we cannot say conclusively that a specificity-based conflict-resolution method is always superior to deny precedence. However, our results showed that over the six tasks in our balanced set (balanced to be fair in alternating the advantage they gave to the two conflict-resolution methods) the specificity-based method was superior. We attribute its superior performance largely to the ease with which it enables authors to fix situations in which its default behavior is not what is desired.

Our study created scenarios that participants could complete within a few minutes in our lab. We were not able to recreate certain real-life scenarios in which policy-authoring may happen periodically over long periods of time. Particularly difficult to simulate in the lab are scenarios in which an author must remember the intention behind a setting that may have been made weeks or months before.

RELATED WORK

There is an extensive body of work describing formal access-control models and languages. The authors of these models and languages necessarily describe a semantics and usually address conflict resolution. The conflict-resolution methods we have discussed here have been covered in the access-control literature (e.g., [5, 7, 10]), but rarely from a usability perspective. Most of these works, as well as others in the general access-control model literature, are concerned with describing the formal aspects of their models and proving theoretical properties about them. A few of these works address ease of policy-authoring under their semantics, but none that we are aware of actually ran user studies to evaluate their semantics empirically as we have.

Others have noted that the underlying model may affect usability [22]. However, past work in policy-authoring usability has mostly focused on interface design or policy visualization [3, 4, 9, 14, 18, 19], rather than how the underlying access-control model (of which the conflict-resolution method is a part) could be best designed for usability. Papers by Fisler and Krishnamurthi [6] and by Kapadia et al. [12]

are exceptions that do address usability through underlying models. Fisler and Krishnamurthi introduce an access control model that captures information about the broader organizational context in which an access-control policy exists and uses this information to sanity check the actual policy itself. Kapadia et al. discuss a system that enables users to debug denied access errors.

CONCLUSION

We have argued that significant usability improvements can be had from an access-control conflict-resolution method based on specificity precedence compared to a method based on deny precedence. We implemented a specificity-based conflict-resolution method in a simulated Windows file system and ran the Expandable Grid interface on top of it. Our user study showed that the specificity-based method provides substantial usability gains for tasks that require a policy author to make changes to a default decision issued by the conflict-resolution method. That is, when the conflict-resolution method gets the decision wrong, specificity-precedence helps the author fix it.

More generally, our results illustrate the impact that underlying system models can have on usability. Our results strengthen the argument for user-centered security, in which usability is a prime consideration during the formative stages of security system design, and in which security and usability experts work side-by-side to ensure a security system that will work as intended when humans interact with it.

ACKNOWLEDGMENTS

This work was supported in part by NSF grants CNS-0756998 and CNS-0627513, and by CyLab at Carnegie Mellon under grants DAAD19-02-1-0389 and W911NF-09-1-0273 from the Army Research Office.

REFERENCES

1. A. Adams and M. A. Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):41–46, December 1999.
2. L. Bauer, L. F. Cranor, R. W. Reeder, M. K. Reiter, and K. Vaniea. A user study of policy creation in a flexible access-control system. *ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 543–552, April 2008.
3. S. Brostoff, M. A. Sasse, D. Chadwick, J. Cunningham, U. Mbanaso, and S. Otenko. 'R-What?' Development of a role-based access control policy-writing tool for e-Scientists. *Software Practice & Experience*, 35(9):835–856, June 2005.
4. X. Cao and L. Iverson. Intentional access management: Making access control usable for end-users. *2nd Symposium on Usable Privacy and Security*, pages 20–31, 2006.
5. D. J. Dougherty, K. Fisler, and S. Krishnamurthi. Specifying and reasoning about dynamic access-control policies. *3rd International Joint Conference on Automated Reasoning*, Lecture Notes in Computer Science, Vol. 4130, pages 632–646, August 2006.
6. K. Fisler and S. Krishnamurthi. A model of triangulating environments for policy authoring. *15th ACM Symp. on Access Control Models and Technologies*, p. 3–12, June 2010.
7. I. Goldberg, D. Wagner, R. Thomas, and E. A. Brewer. A secure environment for untrusted helper applications: Confining the wily hacker. *6th USENIX Security Symposium*, July 1996.
8. N. S. Good and A. Krekelberg. Usability and privacy: a study of Kazaa P2P file-sharing. *ACM SIGCHI Conf. on Human Factors in Computing Systems*, pages 137–144, April 2003.
9. P. Inglesant, M. A. Sasse, D. Chadwick, and L. L. Shi. Expressions of expertness: The virtuous circle of natural language for access control policy specification. *2008 Symposium on Usable Privacy and Security*, July 2008.
10. S. Jajodia, P. Samarati, V. S. Subrahmanian, and E. Bertino. A unified framework for enforcing multiple access control policies. *1997 ACM SIGMOD International Conference on Management of Data*, pages 474–485, 1997.
11. M. Johnson, S. M. Bellovin, R. W. Reeder, and S. E. Schechter. Laissez-faire file sharing: Access control designed for individuals at the endpoints. *2009 New Security Paradigms Workshop*, pages 1–10, 2009.
12. A. Kapadia, G. Sampemane, and R. H. Campbell. KNOW why your access was denied: Regulating feedback for usable security. *11th ACM Conference on Computer and Communications Security*, pages 52–61, 2004.
13. C.-M. Karat, C. Brodie, and J. Karat. Usability design and evaluation for privacy and security solutions. In L. F. Cranor and S. Garfinkel, editors, *Security and Usability*, chapter 4, pages 47–74. O'Reilly, Sebastopol, CA, 2005.
14. C.-M. Karat, J. Karat, C. Brodie, and J. Feng. Evaluating interfaces for privacy policy rule authoring. *ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 83–92, 2006.
15. B. W. Lampson. Protection. *Operating Systems Review*, 8(1):18–24, January 1974.
16. R. A. Maxion and R. W. Reeder. Improving user-interface dependability through mitigation of human error. *International Journal of Human-Computer Studies*, 63(1-2):25–50, July 2005.
17. R. W. Reeder. *Expandable Grids: A user interface visualization technique and a policy semantics to support fast, accurate security and privacy policy authoring*. PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, May 2008. Available as technical report number CMU-CS-08-143.
18. R. W. Reeder, L. Bauer, L. F. Cranor, M. K. Reiter, K. Bacon, K. How, and H. Strong. Expandable grids for visualizing and authoring computer security policies. *ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 1473–1482, April 2008.
19. J. Rode, C. Johansson, P. DiGioia, R. S. Filho, K. Nies, D. H. Nguyen, J. Ren, P. Dourish, and D. Redmiles. Seeing further: Extending visualization as a basis for usable security. *2nd Symp. on Usable Privacy and Security*, pages 145–155, 2006.
20. B. Shneiderman. Direct manipulation: A step beyond programming languages. *Computer*, 16(8):57–69, August 1983.
21. U.S. Senate Sergeant at Arms. Report on the investigation into improper access to the Senate Judiciary Committees computer system. Available at http://judiciary.senate.gov/testimony.cfm?id=1085&wit_id=2514, March 2004.
22. T. Whalen, D. Smetters, and E. F. Churchill. User experiences with sharing and access control. *Conference on Human Factors in Computing Systems Extended Abstracts*, pages 1517–1522, April 2006.
23. M. E. Zurko, R. Simon, and T. Sanfilippo. A user-centered, modular authorization service built on an RBAC foundation. *1999 IEEE Symposium on Security and Privacy*, pages 57–71, May 1999.
24. M. E. Zurko and R. T. Simon. User-centered security. *Workshop on New Security Paradigms*, pages 27–33, Lake Arrowhead, CA, September 1996. Available at http://www.memesoft.com/adage/.