Effects of Access-Control Policy Conflict-Resolution Methods on Policy-Authoring Usability

Robert W. Reeder, Lujo Bauer, Lorrie Faith Cranor, Michael K. Reiter, Kami Vaniea

March 17, 2009
CMU-CyLab-09-006

CyLab
Carnegie Mellon University
Pittsburgh, PA 15213

Effects of Access-Control Policy Conflict-Resolution Methods on Policy-Authoring Usability

Robert W. Reeder* Lujo Bauer† Lorrie Faith Cranor† Michael K. Reiter‡ Kami Vaniea†

ABSTRACT

Access-control policies can be stated more succinctly if they support both rules that grant access and rules that deny access, but this introduces the possibility that multiple rules will give conflicting conclusions for an access. In this paper, we compare a new conflict-resolution method, which uses first specificity and then deny precedence, to the conflict-resolution method used by Windows NTFS, which sometimes uses deny precedence before specificity. We show that our conflict-resolution method leads to a more usable policy-authoring system compared with the Windows method. We implemented both conflict-resolution methods in a simulated Windows NTFS file system and built a state-of-the-art policy authoring interface on top of the simulated file system. We ran a user study to compare policy authors' performance with each conflict-resolution method on a range of file-permissions policy-authoring tasks. Our results show that the conflict-resolution method has a significant effect on usability, and that, though no conflict-resolution method can be optimal for all tasks, our specificity-based conflict-resolution method is generally superior, from a usability perspective, to the Windows deny-based method. Ours is the first user study we are aware of that demonstrates empirically the effect that an access-control semantics can have on usability, independent of the graphical user interface.

1. 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 refer to as "authors" or "policy authors"—are prone to making specification errors that lead to incorrect policies [13, 20, 29].

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 in access-control policies [20].

The conflict-resolution method 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 [27], and since it has been shown to be prone to policy-specification problems [20, 29]. In short, our method resolves conflicts by having specific rules take precedence in cases where the Windows method has deny rules taking precedence.

We implemented our conflict-resolution method in a simulated Windows NTFS file system. Since the Windows policy-authoring utilities are already documented as error-prone [20, 29], we implemented our conflict-resolution method in an *Expandable Grid* interface [24], instead. An Expandable Grid (see Figure 2) is an approximation of Lampson's access-control matrix [19] that illustrates *effective policy* resulting from an underlying set of rules, and that permits authors to modify effective policy directly. 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: the Grid with the Windows conflict-resolution method, the Grid with our conflict-resolution method, and the native Windows interface with the native Windows conflict-resolution method. Our results show that 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, but that our conflict-resolution method is far more usable (leading to gains of up to 78% in accuracy rates) for tasks that require the author to take action. Since no method can always select the intended rule by default, we conclude that our method is likely superior from a usability perspective.

We make two primary contributions. First, we demonstrate empirically that the semantics (in our case, the conflict-resolution aspect of the semantics) underlying an access-control system can have a large effect on usability, even when controlling for the graphical user interface presented to policy authors. Second, we demonstrate the benefits of our specific conflict-resolution method over the Windows method. While we have previously demonstrated the utility of the Expandable Grid [24] as a presentation technique, that work did not thoroughly investigate the effects of different policy semantics independent of interface. This paper builds on our

*Microsoft, Redmond, WA, USA; roreeder@microsoft.com
†Carnegie Mellon University, Pittsburgh, PA, USA; {lbauer,lorrie,kami}@cmu.edu
‡University of North Carolina, Chapel Hill, NC, USA; reiter@cs.unc.edu

prior work by addressing the important question of how the choice of conflict-resolution method affects policy-authoring usability. Our results can help access-control model designers determine what conflict-resolution method is most usable for their appliction.

2. 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.

In this paper, 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. Rules conflict when their principals, resources, and actions overlap but their decisions differ. *Requests* to an access-control system are tuples of the form (*principal*, *resource*, *action*). The access-control system takes requests and returns decisions according to the rules in the policy. The resulting mappings of requests to decisions are known as *effective permissions*. When conflicting rules apply to a request, the conflict-resolution method comes into play to resolve the conflict.

2.1 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"—choosing the decision of the majority of the rules in conflict.) We list several possible conflict-resolution methods below. Note that it is sufficient to define them 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.

Some of the possible conflict-resolution methods for choosing rules to take precedence are:

- *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.
- *Order precedence*: Rules are totally ordered, usually by the policy author, so the author can explicitly state which rules take precedence over others. This is the method commonly used in firewall policies.
- *Recency precedence*: Rules specified more recently in time take precedence over rules specified less recently in time. Note that recency precedence is equivalent to order precedence where order is determined by the time at which each rule was set.

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. For example, the Windows NTFS semantics uses deny precedence if conflicting rules differ in principals, but specificity precedence if conflicting rules differ in resources or in both resources and principals. 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, specificity precedence cannot resolve the conflict.

2.2 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 Windows 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 authors' expectations. We discuss each of these weaknesses below.

2.2.1 Deny precedence

By using deny precedence, the Windows conflict resolution 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 [26]. 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 (Section 4.1.3), 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 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

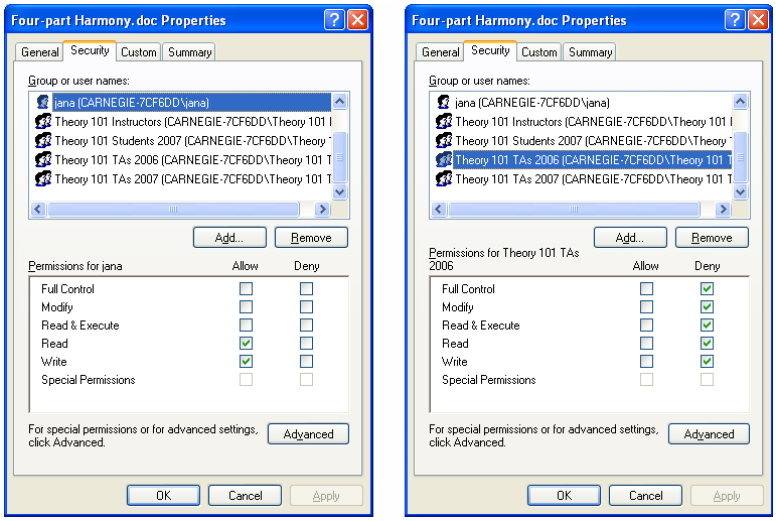

Figure 1: The Windows XP file permissions interface. The left-hand screenshot shows rules applying to Jana that appear to state that Jana is allowed to read and write the operative file, but the right-hand screenshot shows rules applying to a group of which Jana is a member that conflict with Jana's 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.

conflict-resolution method provides no entirely satisfactory way to complete the Jana task.

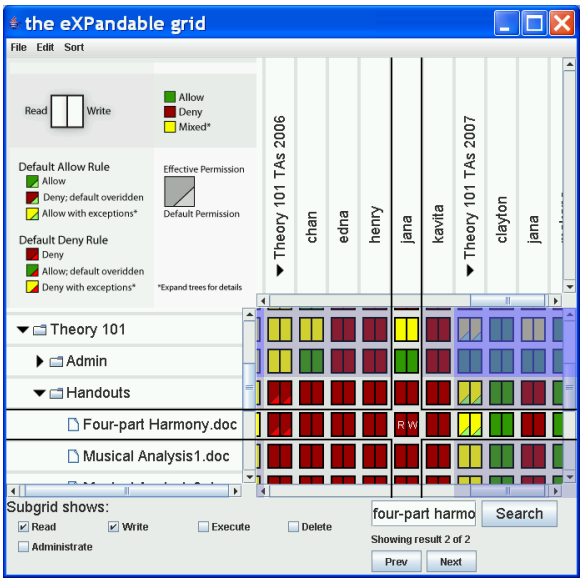

Figure 2: Our Expandable Grid interface for setting file permissions in Windows XP. The interface shows principals along the upper axis, resources along the left-hand axis, and the effective permissions applying to the principals and resources in the colored squares in the grid itself.

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.

We designed the Expandable Grid interface, shown in Figure 2, 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 interface. 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 Section 3 enables direct manipulation of effective permissions in the Grid interface for the Jana task.

2.2.2 *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 while the other is more specific in the resource dimension. For example, in

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.

		Relationship between ALLOW rule's principal and DENY rule's principal			
		Contains	Peer	Same	Contained by
Relationship between ALLOW rule's resource and DENY rule's resource	Contains	DENY (both)	DENY (resources)	DENY (resources)	S:DENY (deny) W:DENY (resources)
	Same	S:DENY (principals) W:DENY (deny)	DENY (deny)	no conflict	S:ALLOW (principals) W:DENY (deny)
	Contained by	S:DENY (deny) W:ALLOW (resources)	ALLOW (resources)	ALLOW (resources)	ALLOW (both)

the Lance task we describe in Section 4.1.2, 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 that the principal dimension to be favored, rather than the resource dimension.

3. 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 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.

The differences between our specificity-based conflict-resolution method and the Windows method can be seen in Table 1, which 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 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

Table 2: Table showing the rule-conflict structure of tasks in our user study. The Charles and Kent tasks and the Lance and Adria tasks were structured the same, but presented inverse goals to participants. The Jana and Pablo tasks were superficial variants on the same structure.

		Relationship between ALLOW rule's principal and DENY rule's principal			
		Contains	Peer	Same	Contained by
Relationship between ALLOW rule's resource and DENY rule's resource	Contains				
	Same		Jana Pablo		Charles (goal: ALLOW) Kent (goal: DENY)
	Contained by	Lance (goal: DENY) Adria (goal: ALLOW)			

change.

4. 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 which we henceforth call *GS*;
- the Grid interface with Windows conflict-resolution method, a combination which we henceforth call *GW*; and
- the Windows interface with the Windows conflict-resolution method, a combination which we henceforth call *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, and when we observe differences between the GW and GS conditions, we attribute them to the conflict-resolution method. 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 Reeder et al.'s work evaluating the Expandable Grid interface [24], our participant pool was consistent with a target demographic of novice and occasional policy authors.

4.1 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 [23] and described in Section A.2.) 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 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. Table 2 shows each task in the table cell corresponding to the type of conflict in the task's initial configuration. 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 (as in the Jana task, where our method allows an author to add a simple user-level exception to the group-level DENY rule, while Windows requires a group-level change).

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 gains from our conflict-resolution method 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). We describe each task pair in more detail below.

4.1.1 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. So, we expected participants in the GW and WW conditions to perform best in the Kent task. The complete Kent task description can be read in Section A.1.1.

4.1.2 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 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. So, we expected better performance from participants in the GW and WW conditions for the Adria task. The complete Adria task description can be read in Section A.1.2.

4.1.3 Jana/Pablo

The Jana task, described in Section 2.2, 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 was superficially different from Jana in the names and numbers of principals and resources involved in the task, but structurally the same. We included the Pablo task for additional assurance that any effect observed in the Jana task is due to the specificity precedence's benefits for the conflict structure, and not merely due to the particular superficial aspects of the Jana task. The complete Pablo task description can be read in Section A.1.3.

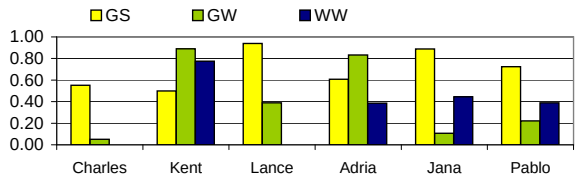

Figure 3: Accuracy results, showing proportion of participants correctly completing each task with GS, GW, and WW interfaces.

4.2 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 (described in Section A.4) 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.

5. RESULTS

Our results are in the form of accuracy rates for each interface condition and for each task. 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 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.

5.1 Experiment-wide results

We performed an experiment-wide test of the hypothesis that GS performed better than GW and WW over all tasks. The accuracy rates over all tasks do show that GS performed best (overall accuracy 70.4%), followed by GW (overall accuracy 41.7%) and then WW (overall accuracy 33.3%). We used logistic regression to test for statistical significance of the effects of interface on accuracy. The fitted model gave an intercept of 0.86 for the GS condition and gave coefficients of -1.20 for the GW condition and -1.56 for the WW condition. Wald tests of the hypotheses that the intercept ($Z = 4.10, p < 0.001$) and the coefficients ($Z = -4.18, p < 0.001; Z = -5.31, p < 0.001$) were not equal to zero were all strongly significant at the 0.05 level, suggesting that the GS condition led to statistically significantly greater accuracy than GW or WW. The difference between the GW and WW accuracy rates (8.4%) shows an improvement in usability due to the Grid interface, and the difference between the GS and GW accuracy rates (28.7%) shows another, much larger improvement from the conflict-resolution method.

5.2 Task-by-task results

We followed up our experiment-wide test with 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 3.

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 simply 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 method to have an advantage by default. However, we expected the specificity-based method to make it quite easy to overcome the conflict by simply setting a specific rule allowing Jana or Pablo to access the relevant files. As we have explained, the Windows

Table 3: 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**

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. This better performance was statistically significant in all four cases.

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.

6. DISCUSSION

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.

The evidence from our study strongly supports our hypothesis that our specificity-based conflict-resolution method is more usable in many situations than the Windows method. 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.

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 [24]. 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 semantics 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 [4].

Our study design had some limitations that 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.

7. RELATED WORK

We discuss two relevant areas of related work: access-control models and policy-authoring usability.

7.1 Access-control models

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. Most of these works 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.

Works addressing the general access-control conflict-resolution methods we have addressed here include Fundulaki and Marx [10], Jajodia et al. [15], Fisler et al. [9], the XACML 1.0 standard [11], Dougherty et al. [7], and Goldberg et al. [12]. Fundulaki and Marx describe an access-control model for eXtensible Markup Language (XML) documents. They acknowledge three conflict-resolution methods: priority precedence, in which a policy author specifies a priority for each rule; deny precedence (which they call "deny overwrites"); and allow precedence (which they call "grant overwrites"). Their semantics uses deny precedence. Jajodia et al. describe an access-control policy framework that accommodates a number of conflict-resolution methods, including specificity, deny, and allow precedence. Their framework also accommodates "no conflicts allowed," in which policies are statically checked for conflicts and conflicts are flagged as errors. Fisler et al. describe Margrave, an RBAC policy-analysis tool, that supports the three conflict-resolution methods outlined in the XACML standard: allow precedence (called "permit-overrides" in the standard), deny precedence (called "deny-overrides" in the standard), and order precedence (called "first-applicable" in the standard). The standard can also be extended to accommodate application-specific conflict-resolution methods. Dougherty et al. are concerned with proving properties of safety and availability in access-control policies, comparing policies to each other, and understanding the effects of dynamic changes to the policy environment. They discuss "combiners," the term used in the XACML standard for conflict-resolution methods, as a factor contributing to policy and policy-authoring complexity. Goldberg et al. describe a module they built for restricting access to system calls by untrusted browser helper applications, like document viewers. They use specificity precedence as the conflict-resolution method in their module.

Several works present policy analysis tools that detect conflicts with the goal of helping policy authors resolve them. Al-Shaer and Hamed describe the Firewall Policy Advisor, a tool that detects firewall policy anomalies, including conflicts [1]. They state that firewall policies traditionally use order precedence to resolve rule conflicts. Yuan et al. present FIREMAN, a toolkit for performing static analysis of firewall policies to check for misconfigurations [31]. Jaeger et al. [14] present the Gokyo policy analysis tool for constraint conflict detection, analysis, and resolution.

7.2 Policy-authoring usability

Past work in policy-authoring usability has mostly focused on interface design or policy visualization, rather than how the underlying access-control model (of which the conflict-resolution method is a part) could be best designed for usability. Kapadia et al. [16], described below, is an exception that does address an underlying model issue, and two works have noted that the underlying model may affect usability, but they do not present user studies to show how [5, 30]. Work in policy-authoring usability includes work on policy authoring for domains such as enterprise privacy policies and firewalls as well as traditional file-system access control.

The HP Select Access Policy Builder is an application for authoring enterprise security and privacy policies [22]. It includes a user interface that provides a visualization of a policy in a matrix indexed by principals on one axis and resources on the other. The idea is similar to the idea of the Expandable Grid [24], but no user study evaluating the HP Policy Builder has been published.

Karat et al.'s SPARCLE system is concerned with the problem of enterprise privacy policy authoring [18, 17]. SPARCLE provides a natural-language input mechanism for authors to write privacy policy rules in English, and also provides a mechanism for authors to select rule elements from structured lists.

Zurko et al. designed the Adage system, a policy-based access control system for distributed computing systems [28, 32]. The Adage designers put a premium on usability, so Adage included the Visual Policy Builder (VPB), a graphical user interface for supporting policy authoring.

Cao and Iverson presented Intentional Access Management (IAM) systems for access control [6]. They define IAM systems as systems that take a user's intention for a specific policy decision, convert that intention into possible rules that would result in the decision, and present the possible rules to the user. Thus, the user is freed from having to figure out when a rule conflict will occur. Cao and Iverson demonstrate that an implementation of IAM for WebDAV, a Web-based distributed file system, is an advance over the standard WebDAV Access Control List editor.

Balfanz developed the ESCAPE Web server, which provided an easy-to-use access control system for content published on the Web [2]. His system allowed a person to put content on the Web and send email announcements to people who were allowed to view the content. The ESCAPE server would automatically give access to the recipients of the emails. The ESCAPE server provided a user interface for viewing directories of Web content and the people who had access to those directories.

The Role Control Center (RCC) is an impressive application for managing role-based access control policies [5, 8]. It includes a graphical user interface that shows the assignment of users to roles in a directed graph and allows for viewing policy either by principal or by resource. However, it does not, as far as we know, include a visualization of the full, effective policy like the Expandable Grid.

In the domain of firewall policy-authoring interfaces, Mayer et al. developed Fang, a firewall analysis tool, and Bartal et al. (including some of the Fang authors) developed Firmato, a firewall management toolkit [21, 3]. Fang includes a user interface for querying the effective policy of a firewall, but

it does not include a visualization of the full policy. Firmato includes a user interface that the authors call a "rule illustrator." The rule illustrator presents a firewall policy in a graph in which host-groups are the nodes and rules are indicated by edges.

Rode et al. designed Impromptu, an file sharing application [25]. Impromptu includes a visualization-based user interface for specifying file sharing policies. The Impromptu visualization depicts users, files, and actions on a pie chart. Its primary limitation seems to be that it was designed for small groups of users (on the order of six), and cannot scale to show policies with many users.

Kapadia et al.'s Know system addresses usability for policy-based systems [16]. Know sits within an access control system and provides feedback when access is denied and explains how access might be obtained.

8. CONCLUSION

We have argued that significant usability improvements can be had from a file-system access-control conflict-resolution method based on specificity precedence compared to a method based on deny precedence. We have implemented a specificity-based conflict-resolution method in a simulated Windows file system and run the Expandable Grid interface on top of it. We have shown in a user study 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.

Our results show the large effects that an aspect of an access-control semantics can have on usability, even when controlling for user interface, and suggest that designers of access-control systems should consider the effects of their design decisions on policy-authoring usability. We have shown that the choice of a specificity-based conflict-resolution method can be a substantial measure toward reducing specification errors. While we have yet to consider factors such as degree of policy-authoring expertise and other variants on conflict-resolution methods, what we have done is a significant first step towards a more comprehensive evaluation that would include those factors.

9. REFERENCES

[1] E. S. Al-Shaer and H. H. Hamed. Firewall Policy Advisor for anomaly discovery and rule editing. In *Proceedings of the IFIP/IEEE Eighth International Symposium on Integrated Network Management*, IFIP International Federation for Information Processing, pages 17–30, New York, NY, March 2003. Springer.

[2] D. Balfanz. Usable access control for the World Wide Web. In *Proceedings of the 19th Annual Computer Security Applications Conference*, pages 406–415, Los Alamitos, CA, December 2003. IEEE Computer Society. Available at http://www.acsac.org/2003/papers/43.pdf.

[3] Y. Bartal, A. Mayer, K. Nissim, and A. Wool. Firmato: A novel firewall management toolkit. *ACM Transactions on Computer Systems*, 22(4):381–420, November 2004.

[4] 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. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems(CHI 2008)*, pages 543–552, New York, NY, April 2008. ACM Press.

[5] R. Bobba, S. Gavrila, V. Gligor, H. Khurana, and R. Koleva. Administering access control in dynamic coalitions. In *Proceedings of the 19th Large Installation System Administration Conference (LISA '05)*, pages 249–261, Berkeley, CA, December 2005. USENIX Association.

[6] X. Cao and L. Iverson. Intentional access management: Making access control usable for end-users. In *Proceedings of the Second Symposium on Usable Privacy and Security (SOUPS 2006)*, pages 20–31, New York, NY, 2006. ACM Press.

[7] D. J. Dougherty, K. Fisler, and S. Krishnamurthi. Specifying and reasoning about dynamic access-control policies. In *Proceedings of the Third International Joint Conference on Automated Reasoning (IJCAR 2006)*, Lecture Notes in Computer Science, Vol. 4130, pages 632–646, New York, NY, August 2006. Springer.

[8] D. F. Ferraiolo, G.-J. Ahn, R. Chandramouli, and S. I. Gavrila. The role control center: Features and case studies. In *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT 2003)*, pages 12–20, New York, NY, June 2003. ACM Press.

[9] K. Fisler, S. Krishnamurthi, L. A. Meyerovich, and M. C. Tschantz. Verification and change-impact analysis of access-control policies. In *Proceedings of the 27th International Conference on Software Engineering (ICSE '05)*, pages 196–205, Los Alamitos, CA, May 2005. IEEE Computer Society Press.

[10] I. Fundulaki and M. Marx. Specifying access control policies for XML documents with XPath. In *Proceedings of the 9th ACM Symposium on Access Control Models and Technologies (SACMAT 2004)*, pages 61–69, New York, NY, June 2004. ACM Press.

[11] S. Godik, T. Moses, A. Anderson, B. Parducci, C. Adams, D. Flinn, G. Brose, H. Lockhart, K. Beznosov, M. Kudo, P. Humenn, S. Godik, S. Andersen, S. Crocker, and T. Moses. eXtensible Access Control Markup Language (XACML) Version 1.0. OASIS Standard, February 2003.

[12] I. Goldberg, D. Wagner, R. Thomas, and E. A. Brewer. A secure environment for untrusted helper applications: Confining the wily hacker. In *Proceedings of the 6th USENIX Security Symposium*, Berkeley, CA, July 1996. USENIX Association.

[13] N. S. Good and A. Krekelberg. Usability and privacy: a study of Kazaa P2P file-sharing. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems(CHI 2003)*, pages 137–144, New York, NY, April 2003. ACM Press.

[14] T. Jaeger, R. Sailer, and X. Zhang. Resolving constraint conflicts. In *Proceedings of the 9th ACM Symposium on Access Control Models and Technologies (SACMAT 2004)*, pages 105–114, New York, NY, June 2004. ACM Press.

[15] S. Jajodia, P. Samarati, V. S. Subrahmanian, and E. Bertino. A unified framework for enforcing multiple access control policies. In *Proceedings of the 1997*

ACM SIGMOD International Conference on Management of Data, pages 474–485, New York, NY, 1997. ACM Press.

[16] A. Kapadia, G. Sampemane, and R. H. Campbell. KNOW why your access was denied: Regulating feedback for usable security. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, pages 52–61, New York, NY, 2004. ACM Press.

[17] C.-M. Karat, J. Karat, C. Brodie, and J. Feng. Evaluating interfaces for privacy policy rule authoring. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems(CHI 2006)*, pages 83–92, New York, NY, 2006. ACM Press.

[18] J. Karat, C.-M. Karat, C. Brodie, and J. Feng. Privacy in information technology: Designing to enable privacy policy management in organizations. *International Journal of Human-Computer Studies*, 63(1-2):153–174, July 2005.

[19] B. W. Lampson. Protection. *Operating Systems Review*, 8(1):18–24, January 1974. Reprint of the original from *Proceedings of the Fifth Princeton Symposium on Information Sciences and Systems* (Princeton University, March, 1971), 437-443.

[20] 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.

[21] A. Mayer, A. Wool, and E. Ziskind. Fang: A firewall analysis engine. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy (S&P 2000)*, pages 177–187, Los Alamitos, CA, May 2000. IEEE Computer Society Press.

[22] M. C. Mont, R. Thyne, and P. Bramhall. Privacy enforcement with HP Select Access for regulatory compliance. Technical Report HPL-2005-10, HP Laboratories Bristol, Bristol, UK, January 2005. Available at http://www.hpl.hp.com/techreports/2005/HPL-2005-10.pdf.

[23] 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.

[24] 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. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems(CHI 2008)*, pages 1473–1482, New York, NY, April 2008. ACM Press.

[25] 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. In *Proceedings of the Second Symposium on Usable Privacy and Security (SOUPS 2006)*, pages 145–155, New York, NY, 2006. ACM Press.

[26] B. Shneiderman. Direct manipulation: A step beyond programming languages. *Computer*, 16(8):57–69, August 1983.

[27] R. Stevenson. Windows XP sales touches 210m copies. Available at http://www.ciol.com/content/news/2004/104050307.asp, May 2004. Report that 210 million copies of XP have been sold.

[28] The Open Group Research Institute. Adage system overview. Available at http://www.memesoft.com/adage/SystemSpec.ps. Accessed via HTTP on September 20, 2006.

[29] U.S. Senate Sergeant at Arms. Report on the investigation into improper access to the Senate Judiciary Committee's computer system. Available at http://judiciary.senate.gov/testimony.cfm?id=1085&wit_id=2514, March 2004.

[30] T. Whalen, D. Smetters, and E. F. Churchill. User experiences with sharing and access control. In *Conference on Human Factors in Computing Systems(CHI 2006) Extended Abstracts*, pages 1517–1522, New York, NY, April 2006. ACM Press.

[31] L. Yuan, J. Mai, Z. Su, H. Chen, C.-N. Chuah, and P. Mohapatra. FIREMAN: A toolkit for FIREwall Modeling and ANalysis. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P 2006)*, pages 199–213, Los Alamitos, CA, 2006. IEEE Computer Society Press.

[32] M. E. Zurko, R. Simon, and T. Sanfilippo. A user-centered, modular authorization service built on an RBAC foundation. In *Proceedings 1999 IEEE Symposium on Security and Privacy*, pages 57–71, Los Alamitos, CA, May 1999. IEEE Computer Security Press.

APPENDIX

A. ADDITIONAL TASK DESCRIPTIONS

This appendix gives the text and structure for the nine of our 12 tasks that were not described in the body of the paper due to space constraints and for our training task. The first three of the tasks described here (Kent, Adria, and Pablo) were reported on in this paper; the following six tested aspects of the Grid interface, and of the semantics we defined for it, other than the conflict-resolution method. The last (Carolyn) was the training task used in all three experimental conditions.

A.1 Tasks reported on in this paper

A.1.1 Kent

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.

In the initial configuration, there are rules stating that Kent is allowed READ and WRITE access to the attendance.xls and gradebook.xls files in the Classes\Choir 1\Admin folder, but there are rules stating that the group Choir 1 Students 2008 is denied READ and WRITE access to those files. So, when Kent is added to Choir 1 Students 2008, that group's DENY rules will apply to him. Under the Windows semantics, the group DENY rules will take precedence, but under specificity semantics, he will still be allowed to read and write the files. Thus, the task is done after moving Kent out of the TA group and into the student group in the GW and WW conditions, but extra action is required to complete the task in the GS condition. So, we expect the Windows semantics, embodied in GW and WW, to perform better on the Kent task. However, we expect the Grid's visual presentation advantages over the Windows list-of-rules presentation at least partially to offset the semantics disadvantage.

A.1.2 Adria

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. In the initial configuration, there is a rule stating that Adria is denied READ access to the Music 101\Lecture Notes folder and there is a rule stating that the Music 101 Instructors group is allowed READ access to the Music101\Lecture Notes\Bach.ppt file. This latter group access rule is initially suppressed in both semantics by a rule stating that the Opera Instructors group is denied READ access to the Bach.ppt file, but when Adria is removed from Opera Instructors, a two-dimensional conflict is revealed. In the Windows semantics, the group access rule, which is more specific in its resource, will take precedence and Adria will be allowed to read the Bach.ppt file as soon as she is removed from Opera Instructors. In our specificity semantics, deny-precedence is used to resolve two-dimensional conflicts, so Adria will not be allowed to read the Bach.ppt file without some extra work. We thus expect the GW and WW conditions to perform better than the GS condition for the Adria task. However, the extra work required in the GS condition is quite simple. With the specificity semantics, simply setting an access rule explicitly allowing Adria to read the Bach.ppt file is sufficient, as the rule applying to Adria and Bach.ppt will be more specific than the rules in conflict. We thus expect that the GS's presentation advantages over the WW condition combined with the ease of overcoming the two-dimensional conflict might counteract the WW semantics advantage.

A.1.3 Pablo

The Pablo task, like the Jana task described in Section 2.2, presents a group conflict in which a user is a member of two groups, one of which has a rule allowing it access to a file and the other of which has a rule denying it access to the same file. 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.)

In the initial configuration, Pablo is a member of the group Music 101 Students 2008, for which there is a rule allowing the group READ access to the assignment4.pdf file, and he is a member of the group Troublemakers, for which there is a rule denying the group READ access to the assignment4.pdf file. In both semantics, deny-precedence is in effect for conflicts between groups. The difference between the semantics comes into play when participants try to resolve the conflict. In the specificity semantics, authors can simply create a rule allowing Pablo to read assignment4.pdf, and it will take precedence because it will be more specific than either of the group access rules. In the GS interface, creating such a rule is a simple matter of clicking on the square corresponding to Pablo, assignment4.pdf, and READ. In the Windows semantics, such a rule would not take precedence; completing the task requires either removing Pablo from the Troublemakers group or removing the rule that denies Troublemakers READ access to the file. In either solution, care must be taken not to change the effective permissions of other members of the Troublemakers group. Because there are more steps involved in the solution for the Windows semantics, we expect better performance with the GS interface for the Pablo task.

A.2 Tasks included in the present study but reported on elsewhere

A.2.1 Piano

The Piano task tests the interfaces' group-creation functionality.

The Piano task statement presented to participants was:

> A new seminar on piano performance was just started.
>
> Create a group called **Piano Students 2008** and add the following students to it: **Aaron, Camilla, Thor, and Uzi**. Set permissions so that these students and any students added to Piano Students 2008 in the future will be able to **read** the **Piano\Handouts** folder.

A.2.2 Troublemakers

The Troublemakers task tests the ability of the interfaces to reveal an undesired exception to a rule.

The Troublemakers task statement presented to participants was:

> The music department is full of pranksters. These people have been put in the Troublemakers group.
>
> Set permissions so that **no one** in the **Troublemakers** group has access to anything.

In the initial configuration, there are rules denying all access to the root Classes folder to members of the Troublemakers group, but there is one user who has access to a folder two levels below the root folder. The Windows interface only allows the author to check the effective policy for one user for one resource at a time. In our configuration, there are seven users in the Troublemakers group and 29 folders one or two levels below the root, so it could take up to 7x29=203 operations to ensure that no members of the Troublemakers group have access to any of those folders using the Windows interface. The Grid interfaces show the aggregate effective policy for READ and WRITE for the Troublemakers at the root folder as mixed (yellow squares with red dog-ears), thereby providing the author a clue that some member of Troublemakers has access to something. Expanding the group reveals that Marie is the user with access to something within the Opera folder, and expanding the Opera folder reveals two rules stating that Marie has access to read and write the Opera\Admin folder. Because of the relative ease of spotting exceptions to the desired DENY rule in the Grid interfaces, we expect them to perform better in the Troublemakers task compared to the Windows interface.

A.2.3 Assignment

The assignment task tests the semantics in the presence of file moves and illustrates a potentially confusing aspect of the Windows semantics.

The Assignment task statement presented to participants was:

> The Music 101 Assignment 1 had some mistakes and needs to be edited. Students should be able to read the file while it is being edited, and TAs will need to read and write the file.
>
> Move the **assignment1.pdf** file from the **Classes\Music 101\Handouts** folder to the **Classes\Music 101\Drafts** folder. Then set permissions so that:
>
> - **Music 101 Students 2008** can **read** the **assignment1.pdf** file; and
> - **Music 101 TAs 2008** can **read and write** the **assignment1.pdf** file.

In the Windows semantics, when a file is moved from one parent folder to another, it retains the rules it inherited from its old parent. When a change is made to the file's or its new parent's access control list, the file then inherits all rules from the new parent, and loses rules from the old parent. In our semantics, a moved file inherits rules from its new parent immediately.

In the initial configuration for the Assignment task, relevant rules state that:

- Music 101 Students 2008 are allowed READ access to files in the Handouts folder;
- Music 101 Students 2008 are denied READ access to files in the Drafts folder;
- Music 101 TAs 2008 are allowed READ access but denied WRITE access to files in the Handouts folder; and
- Music 101 TAs 2008 are allowed READ and WRITE access to files in the Drafts folder.

From the rules, we see that if the assignment1.pdf file is moved and inherits rules from its new parent folder Drafts, the task will be correctly completed. Thus, under our semantics, the Assignment task is complete as soon as the participant has moved the file. Under the Windows semantics, though, participants may be tricked by the rule that the file retains its old parent's rules, but may suddenly inherit all of its new parent's rules if the participant makes a change. So, under the Windows semantics, after the participant has moved the assignment1.pdf file, they may check to see that Music 101 Students 2008 can read the file. Because students are allowed to read files in the old parent folder Handouts, the participant will note that the students have the correct access. The participant may then proceed to check what access the Music 101 TAs 2008 have to the assignment1.pdf file, and note that they have can read but not write the file. However, when the participant creates a rule allowing the TAs WRITE access to the file, the file will inherit all rules from its new parent, including the rule that the students are denied READ access to files in the Drafts folder. Unless the participant goes back and checks the students' access again, they will end up with the wrong access and the task will not be completed correctly. Thus, we expect that the GS interface will perform best in the Assignment task.

A.2.4 Syllabus

The syllabus task is structured similarly to the Assignment task but with the inverse goal, so it gives the advantage to the Window semantics.

The Syllabus task statement presented to participants was:

> The Music 101 syllabus was in draft form, but is now complete and ready for students to read.
>
> Move the **syllabus.doc** file from the **Classes\Music 101\Admin** folder to the **Classes\Music 101\Handouts** folder. Then set permissions so that all members of **Music 101 Students 2008** can **read** the **syllabus.doc** file.

In the initial configuration, there is a rule stating that all students have READ access to the syllabus.doc file, but there are rules stating that four of the students are denied READ access to files in the Handouts folder. Under the Windows semantics, moving the file does not change the students' access, and since they all already have READ access to the file, the task is complete as soon as the participant moves the file into the Handouts folder. In our semantics, when the participant moves the file, it inherits its new parents' rules, so the rules denying READ access to the four students go into effect. The participant must make the extra effort to check the effective policy for those students and explicitly grant them READ access to correctly complete the task. Thus, we expect the GW and WW interfaces with the Windows semantics to perform better in the Syllabus task.

A.2.5 Clayton

The Clayton task tested whether participants could read effective permissions correctly when a user is inheriting permissions from a group.

The Clayton task statement presented to participants was:

> Clayton, a Theory 101 TA, is going away on a trip and cannot grade the Simple Solo assignment. Because of this he will be put in charge of grading the Simple Harmony assignment. Clayton will need read and write access to the Simple Harmony assignment. He also may wish to refer to the Simple

Solo assignment, so he should be allowed to read the Simple Solo submissions folder. However, you don't want him to accidentally overwrite the submissions to Simple Solo when viewing them.

Set permissions so that **Clayton** can **read and write** the **Simple Harmony** subfolder in the Theory 101\Submissions folder.

Set permissions so that **Clayton** can **read, but not write**, the **Simple Solo** subfolder in the Theory 101\Submissions folder.

A.2.6 Quincy

The Quincy task tested whether participants could preserve a desired exception to a group rule when changing that group rule.

The Quincy task statement presented to participants was:

> Students in the Choir class are going to sing the War Requiem, so the Choir instructor, Savanna, has asked you to give them access to the War Requiem lyric sheets. However, you remember that last week she asked you to prevent Quincy from accessing any tenor parts, because, although he obnoxiously insists he is a tenor, Savanna wants him to sing baritone. You asked, and she confirmed that Quincy should not have access to the tenor part. Other students should have access to all the parts.
>
> Set permissions so that **Choir 1 Students 2008** can **read** the files in the **War Requiem** subfolder of the Choir 1\Lyrics folder. But remember that Quincy should not be allowed to read the tenor part.

We designed the Quincy task to test a weakness of Reeder et al.'s recency conflict-resolution method [24]. In a recency conflict-resolution method, a rule at the group and folder level can override an exception at the user and file level that was intended to be kept. In the initial configuration for the Quincy task, there is rule denying Quincy READ access to the Tenor.pdf file in the War Requiem folder. Under a recency method, when a rule is created to allow all students READ access to all files in the War Requiem folder, the DENY rule applying to Quincy will be overridden. However, under both the Windows method and our specificity method, the exception will stay in place because it is more specific than the new rule applying to students and the War Requiem folder. Thus, we expect similar performance amongst the interfaces for the Quincy task.

A.3 Results for other tasks

Results for the six tasks not reported on in the main body of the paper are shown in Figure 4. They suggest that the Expandable Grid may have additional advantages over the Windows file-permissions interface besides its conflict-resolution method. In particular, the Grid (both GS and GW) led to higher accuracy in the Troublemakers task, which tests the ability of the interface to make policy anomalies stand out. GS also has considerably higher accuracy for the Assignment task, which tested our rule for when a moved file inherits its new parent folder's permissions. However, the Assignment and Syllabus tasks were designed as a task pair, in which GS has the advantage relative to the Assignment task's goals and GW and WW have the advantage for relative to the Syllabus task's goals, and GS's superior performance in the Assignment task is tempered by its inferior performance in the Syllabus task.

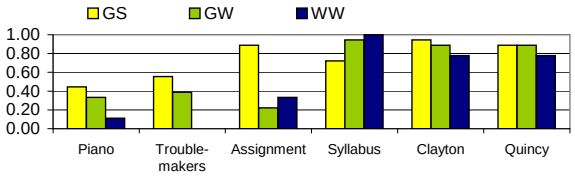

Figure 4: Accuracy results for the six tasks not reported on in the main body of the paper, showing proportion of participants correctly completing each task with GS, GW, and WW interfaces.

A.4 Training task

A.4.1 Carolyn

The Carolyn task is a simple task to give participants practice with searching for a user, searching for a file, adding a user to a group, reading effective permissions, and creating a new policy rule. It did not involve a rule conflict, so did not provide any practice to participants in how to handle conflicts. We used the same training task for participants in all three experimental conditions.

The Carolyn task statement presented to participants was:

> Carolyn is auditing the class Music 101 this semester. She won't be submitting any assignments for the class and should not be allowed to do anything to the Submissions folder.
>
> Add **Carolyn** to the **Music 101 Students 2008** group. Set permissions so that she has **no read or write access** to the **Classes\Music 101\Submissions** folder.