Usability Challenges in Security and Privacy Policy-Authoring Interfaces

Robert W. Reeder[1], Clare-Marie Karat[2], John Karat[2], and Carolyn Brodie[2]

[1] Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh PA 15213, USA
reeder@cs.cmu.edu,
[2] IBM T.J. Watson Research Center, 19 Skyline Dr., Hawthorne NY 10532, USA
{ckarat, jkarat, brodiec}@us.ibm.com

Abstract. Policies, sets of rules that govern permission to access resources, have long been used in computer security and online privacy management; however, the usability of authoring methods has received limited treatment from usability experts. With the rise in networked applications, distributed data storage, and pervasive computing, authoring comprehensive and accurate policies is increasingly important, and is increasingly performed by relatively novice and occasional users. Thus, the need for highly usable policy-authoring interfaces across a variety of policy domains is growing. This paper presents a definition of the security and privacy policy-authoring task in general and presents the results of a user study intended to discover some usability challenges that policy authoring presents. The user study employed SPARCLE, an enterprise privacy policy-authoring application. The usability challenges found include supporting object grouping, enforcing consistent terminology, making default policy rules clear, communicating and enforcing rule structure, and preventing rule conflicts. Implications for the design of SPARCLE and of user interfaces in other policy-authoring domains are discussed.

Keywords: Policy, policy-authoring, privacy, security, usability.

1 Introduction

Policies are fundamental to providing security and privacy in applications such as file sharing, Web browsing, Web publishing, networking, and mobile computing. Such applications demand highly accurate policies to ensure that resources remain available to authorized access but not prone to compromise. Thus, one aspect of usability, very low error rates, is of the highest importance for user interfaces for authoring these policies. Security and privacy management tasks were previously left to expert system administrators who could invest the time to learn and use complex user interfaces, but now these tasks are increasingly left to end-users. Two non-expert groups of policy authors are on the rise. First are non-technical enterprise policy authors, typically lawyers or business executives, who have the responsibility to write policies governing an enterprise's handling of personal information [1]. Second are end-users, such as those who wish to set

C. Baranauskas et al. (Eds.): INTERACT 2007, LNCS 4663, Part II, pp. 141–155, 2007.

up their own spam filters, share files with friends but protect them from unwanted access [2,3,4], or share shipping information with Web merchants while maintaining privacy [5]. These two groups of non-expert users need to complete their tasks accurately, yet cannot be counted on to gain the expertise to tolerate a poorly designed or unnecessarily complex user interface.

Despite the need for usable policy-authoring interfaces, numerous studies and incidents have shown that several widely-used policy-authoring interfaces are prone to serious errors. The "Memogate" scandal, in which staffers from one political party on the United States Senate Judiciary Committee stole confidential memos from an opposing party, was caused in part by an inexperienced system administrator's error using the Windows NT interface for setting file permissions [6]. Maxion and Reeder showed cases in which users of the Windows XP file permissions interface made errors that exposed files to unauthorized access [4]. Good and Krekelberg showed that users unwittingly shared confidential personal files due to usability problems with the KaZaA peer-to-peer file-sharing application's interface for specifying shared files [3]. This evidence suggests that designing a usable policy-authoring interface is not trivial, and that designers could benefit from a list of potential vulnerabilities of which to be aware.

This paper reports the results of a user study which had the goal of identifying common usability challenges that all policy-authoring interface designers must address. The study employed SPARCLE [7], an application designed to support enterprise privacy policy authoring. However, the study was not intended to evaluate SPARCLE itself, but rather to reveal challenges that must be addressed in policy authoring in general. SPARCLE-specific usability issues aside, the study revealed five general usability challenges that any policy-authoring system must confront if it is to be usable:

1. **Supporting object grouping;**
2. **Enforcing consistent terminology;**
3. **Making default rules clear;**
4. **Communicating and enforcing rule structure;**
5. **Preventing rule conflicts.**

These challenges are explained in detail in the discussion in Sect. 6. Although these challenges have been identified from a user study in just one policy-authoring domain, namely enterprise privacy policies, a review of related work, presented in Sect. 7, confirms that the challenges identified here have been encountered in a variety of other policy-authoring domains, including file access control, firewalls, website privacy, and pervasive computing. This work, however, is the first we are aware of to describe policy-authoring applications as a general class and present usability challenges that are common to all.

2 Policy Authoring Defined

"Policy" can mean many things in different contexts, so it is important to give a definition that is germane to the present work on security and privacy policies.

For the purposes of this work, a policy is defined as a function that maps sets of elements (tuples) onto a discrete set of results, typically the set {ALLOW, DENY} (however, other result sets are possible; for example, Lederer et al. describe a policy-based privacy system in which tuples representing requests for a person's location are mapped to the set {PRECISE, APPROXIMATE, VAGUE, UNDISCLOSED} [8]). Elements are defined as the attributes relevant to a specific policy domain; the values those attributes can take on are referred to as element values. Element values may be literal values, such as the username "jsmith", or they may be expressions, such as "if the customer has opted in." Policies are expressed through rules, which are statements of specific mappings of tuples to results. Policy authoring is the task of specifying the element values in a domain, specifying rules involving those elements, and verifying that the policy comprised by those rules matches the policy that is intended.

In the privacy policy domain, for example, a policy maps tuples of the form (<user category>, <action>, <data category>, <purpose>, <condition>) to the set {ALLOW, DENY} [9]. Here, user category, action, data category, purpose, and condition are elements. ALLOW and DENY are results. An example privacy policy rule would map the tuple ("Marketing Reps", "use", "customer address", "mailing advertisements", "the customer has opted in") to the value ALLOW, indicating that marketing representatives can use the customer address data field for the purpose of mailing advertisements if the customer has opted in. Here, "Marketing Reps", "use", "customer address", "mailing advertisements", and "the customer has opted in" are element values.

To take another example, in the file permissions domain, a policy maps tuples of the form (<principal>, <action>, <file>) to the set ALLOW, DENY. An example rule might map ("jsmith", "execute", "calculator.exe") to the value DENY, indicating that the user jsmith cannot execute the program calculator.exe.

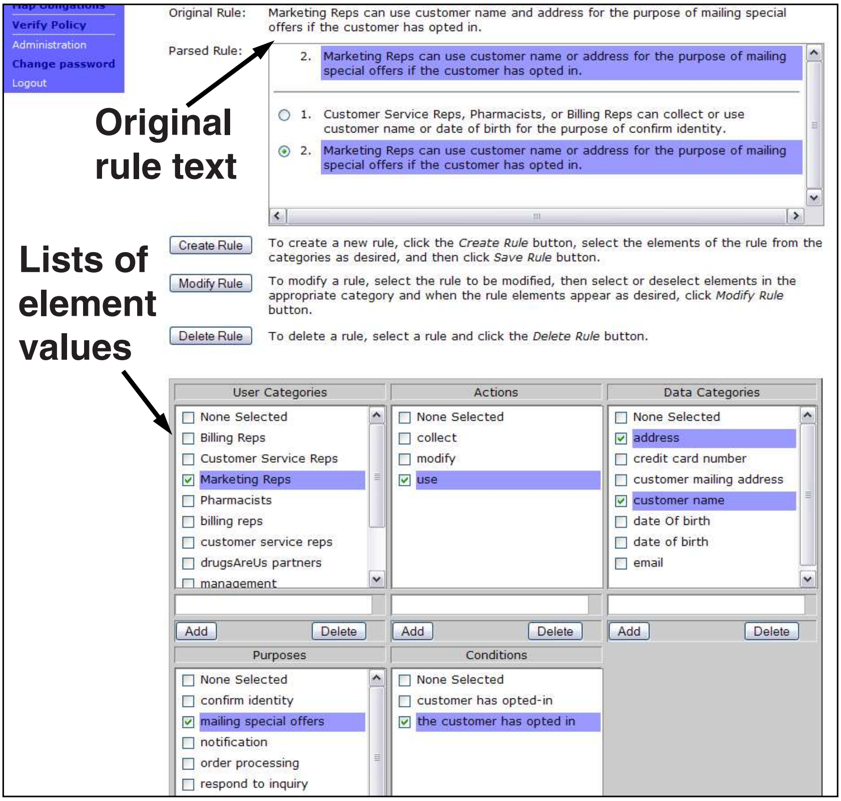
Since the rules in a policy may not cover all possible tuples, policy-based security and privacy systems typically have a default rule. For example, the SPARCLE system has the default rule that all 5-tuples of (<user category>, <action>, <data category>, <purpose>, <condition>) map to DENY. Additional rules are specified by policy authors and all author-specified rules map a 5-tuple to ALLOW. The default rule need not necessarily be a default DENY; a default ALLOW is also possible (policies with a default ALLOW rule are often called "blacklists", because any element value listed explicitly in the policy is denied access), or the default can vary according to some values in the tuples (for instance, a default rule might state that all accesses to shared files on a computer are allowed by default to local users but denied by default to remote users). Similarly, user-specified rules need not necessarily map exclusively to ALLOW or exclusively to DENY; it is possible to allow users to specify rules that map to either ALLOW or DENY. However, policy systems that allow users to specify both types of rules introduce the potential for rule conflicts, which can be a significant source of user difficulty [10,2,4].

3 The SPARCLE Policy Workbench

In the present work, policy authoring usability was investigated through a user study in which participants used the SPARCLE Policy Workbench application. SPARCLE is a Web-based application for enterprise privacy policy management. The application includes a front-end user interface for authoring privacy policies using a combination of natural language and structured lists. While the current embodiment of SPARCLE is tailored for the privacy policy domain, the user interface was designed with an eye toward supporting policy authoring in other domains such as file access control. From a research perspective, the two interaction paradigms supported by SPARCLE, natural language and structured list entry, can as easily be applied to privacy policy management as to file, network, system, email, or other policy management. Thus SPARCLE is a suitable tool for studying policy authoring in the general sense. Portions of the SPARCLE user interface relevant to the present study are described below; a more complete description of the SPARCLE system can be found elsewhere [1].

One way to write policy rules in SPARCLE is to use the natural language interface on the Natural Language Authoring page. The Natural Language Authoring page contains a rule guide, which is a template that reads, "[User Category(ies)] can [Action(s)] [Data Category(ies)] for the purpose(s) of [Purpose(s)] if [(optional) Condition(s)]." This template indicates what elements are expected in a valid rule. The Natural Language Authoring page further contains a large textbox for entering rule text. Policy authors can type rules, in natural language, directly into the textbox. For example, a rule might read, "Customer Service Reps, Pharmacists, and Billing Reps can collect and use customer name and date of birth to confirm identity." SPARCLE has functionality for parsing an author's rules so that it can extract rule element values automatically from the author's text. When parsing has completed, the user proceeds to the Structured Authoring page to see the structured format of the policy.

The Structured Authoring page, shown in Fig. 1, shows the results of parsing each policy rule. When SPARCLE parses each policy rule, it saves the elements (i.e., user categories, actions, data categories, purposes, and conditions) found in that rule. The elements are reconstructed into sentences and shown next to radio buttons in a list of rules. The lower half of the Structured Authoring page contains lists of element values, one list for each of the five elements of a privacy policy rule. These lists contain some pre-defined, common element values (e.g., "Billing Reps" as a user category, "collect" as an action, or "address" as a data category) as well as all element values defined by the policy author and found by the parser in rules written on the Natural Language Authoring page. Policy authors can also alter these element value lists directly by adding or deleting elements. When a rule is selected from the list of rules at the top of the Structured Authoring page, all of the element values in that rule are highlighted in the lists of element values. Policy authors can edit rules by selecting different element values from the lists. It is also possible to create rules from scratch on the Structured Authoring page.



Original Rule: Marketing Reps can use customer name and address for the purpose of mailing special offers if the customer has opted in.

Parsed Rule: 2. Marketing Reps can use customer name or address for the purpose of mailing special offers if the customer has opted in.

○ 1. Customer Service Reps, Pharmacists, or Billing Reps can collect or use customer name or date of birth for the purpose of confirm identity.

● 2. Marketing Reps can use customer name or address for the purpose of mailing special offers if the customer has opted in.

To create a new rule, click the *Create Rule* button, select the elements of the rule from the categories as desired, and then click *Save Rule* button.

To modify a rule, select the rule to be modified, then select or deselect elements in the appropriate category and when the rule elements appear as desired, click *Modify Rule* button.

To delete a rule, select a rule and click the *Delete Rule* button.

User Categories

- □ None Selected
- □ Billing Reps
- □ Customer Service Reps
- ☑ Marketing Reps
- □ Pharmacists
- □ billing reps
- □ customer service reps
- □ drugsAreUs partners
- □ management

Actions

- □ None Selected
- □ collect
- □ modify
- ☑ use

Data Categories

- □ None Selected
- ☑ address
- □ credit card number
- □ customer mailing address
- ☑ customer name
- □ date Of birth
- □ date of birth
- □ email

Purposes

- □ None Selected
- □ confirm identity
- ☑ mailing special offers
- □ notification
- □ order processing
- □ respond to inquiry

Conditions

- □ None Selected
- □ customer has opted-in
- ☑ the customer has opted in

Fig. 1. SPARCLE's Structured Authoring page. Policy authors can create or edit rules on this page by selecting from the lists of element values in the lower half of the page.

4 Policy Authoring Usability Evaluation

We conducted a laboratory user study using the SPARCLE application to identify usability problems experienced by users in policy-authoring tasks.

4.1 User Study Method

Participants. We recruited twelve participants, consisting of research staff and summer interns at a corporate research facility, for the user study. Participants varied in age from 20 to 49; four were female. Since participants did not have experience authoring privacy policies or using SPARCLE, we considered them novice users for our purposes. Participants were compensated for their participation with a certificate for a free lunch.

Apparatus. Participants accessed SPARCLE through the Internet Explorer web browser on a computer running Windows XP. SPARCLE ran on a server on a local intranet, so users experienced no network delays. We set up a camera and voice recorder in the laboratory to record participants' actions and words.

Training Materials. We presented participants with a 4-page paper tutorial on how to use SPARCLE to give them a basic introduction to the SPARCLE system. The tutorial walked participants through the SPARCLE Natural Language Authoring and Structured Authoring pages as it had participants write and edit a two-rule policy. The tutorial took about 15 minutes to complete.

Tasks. We wrote three task scenarios: the "DrugsAreUs" task, the "Bureau of the Public Debt" task, and the "First Finance" task. The three scenarios describe medical, government, and finance enterprises, respectively, and thus cover a broad range of the types of enterprises that require privacy policies in the real world. Each task scenario described an enterprise and its privacy policy requirements in language that did not explicitly state rules to be written into SPARCLE, but suggested content that might go into explicit rules. The intent of the scenarios was to give participants an idea of what to do, but to have them come up with their own language for their rules. An example of one of the three task scenarios, the "DrugsAreUs" task, is listed in Table 1.

Table 1. The task statement given to participants for the DrugsAreUs task, one of three tasks used in the user study

The Privacy Policy for DrugsAreUs

Our business goals are to answer customer questions when they call in (Customer Service), fulfill orders for prescriptions while protecting against drug interactions (Pharmacists), and to provide customers valuable information about special offers (Marketing). In order to make sure our customers' privacy is protected, we make the following promises concerning the privacy of information we collect at DrugsAreUs. We will only collect information necessary to provide quality service. We will ask the customers to provide us with full name, permanent address and contact information such as telephone numbers and email addresses, and a variety of demographic and personal information such as date of birth, gender, marital status, social security number, and current medications taken. On occasions where we need to verify a customer's identity, Customer Service Reps will only use the social security number to do so. Our pharmacists will use the current medication information when processing new orders to check for drug interactions.

We will make reports for our internal use that include age and gender breakdowns for specific drug prescriptions, but will not include other identifying information in the reports and will delete them after five years. For example, our research department might access customer data to produce reports of particular drug use by various demographic groups.

Procedure. We asked participants to complete a demographic survey before beginning the user study. We then gave participants the SPARCLE tutorial and asked them to complete it. We provided help as needed as the participants worked through the tutorial. After they had completed the tutorial, we instructed participants to think aloud during the study [11]. We then presented participants with tasks. Each participant was presented with two of the three task scenarios and asked to complete them one at a time. We instructed participants to imagine they were the Chief Privacy Officer of the enterprise described in each scenario and to use SPARCLE to author the rules they thought were necessary to protect personal information held by the enterprise while still allowing the enterprise to carry out its business. We counter-balanced the selection and presentation of the scenarios across participants so that each scenario was presented to eight participants and each scenario was the first scenario presented to four participants and the second scenario presented to four other participants.

Data Collection. The data we collected included text of rules written, video of participants and the computer screen on which they worked, think-aloud audio, and results of the demographic survey.

4.2 Data Analysis Method

We performed two data analyses. In the first analysis, we looked at the rules participants wrote to find errors in rules. In the second analysis, we reviewed videos and think-aloud audio data to find any additional incidents of errors and usability problems not found in the first analysis.

First Analysis: Errors in Rules In the first analysis, we read through participants' final rules and considered their implementability. An implementable privacy rule was defined as a rule that can be unambiguously interpreted by an implementer, which is a human or machine that produces the actionable code to carry out automated enforcement of a policy's intentions. With respect to implementability, we identified seven errors. Two of these errors, undetected parser errors and unsaved rule changes, are system-specific issues to SPARCLE, and are not further discussed here, because the objective of this work was to identify errors that might occur in any interface for policy-authoring. The five non-system-specific errors we found were group ambiguities, terminology mismatches, negative rules, missing elements, and rule conflicts. We identified and counted occurrences of each type of error. The five errors, criteria used to identify each type of error, and examples of each error are below:

1. **Group ambiguity:** Composite terms, i.e., terms that represented a set of other terms, were used in the same policy as the terms they apparently represented. This was considered an error for implementation because it was often not clear exactly which terms were represented by the composite term. For example, one rule said, "DrugsAreUs can collect necessary information...," in which "necessary information" is a composite term presumably representing data referred to in other rules such as "customer mailing address," and "current medications taken." However, it is not immediately clear just what data is referred to by "necessary information." As another example, one rule contained both the terms "contact information" and "permanent address." This would imply that, contrary to common usage, "permanent address" is not part of "contact information." An implementer could easily be confused as to whether the term "contact information" used in a different rule included permanent address data or not.

2. **Terminology mismatch:** Multiple terms were used to refer to the same object within the same policy. Examples of terminology mismatches included "email address" and "email addres"; "Financial control" and "Finacial control"; "gender" and "gender information"; "properly reporting information to the IRS" and "providing required reports to the IRS."
3. **Negative rule:** A rule's action contained the word "not". Although SPARCLE is a default-deny policy system, implying that it is only necessary to specify what is allowed, some participants attempted to write negative rules, i.e., rules that prohibited access. These rules are unnecessary, and can lead to confusion on the part of an implementer who is expecting only positive rules. An example of a negative rule was, "Bureau of the Public Debt can not use persistent cookies...."
4. **Missing element:** A rule was missing a required element. The missing element was usually purpose. An example of a rule with no purpose was "Customer Service Reps can ask full name, permanent address, and medication taken."
5. **Rule conflict:** Two different rules applied to the same situation. Only one rule conflict was observed in our study. SPARCLE's policy semantics avoid most potential for rule conflict by taking the union of all access allowed by the rules except in the case of conditions, for which the intersection of all applicable conditions is taken. The one observed example of a rule conflict included the rules, "Customer Service Reps can access customer name for the purpose of contacting a customer **if the customer has submitted a request**," and "Customer Service Reps can access customer name for the purpose of contacting a customer **if the customer has expressed a concern**." Since the user category ("Customer Service Reps"), action ("access"), data category ("customer name"), and purpose ("contacting a customer") all match in these rules, taking the intersection of conditions would imply that "Customer Service Reps can access customer name for the purpose of contacting a customer" only when both "the customer has submitted a request" and "the customer has expressed a concern" are true. Thus, in the case that the customer has submitted a request but not expressed a concern, the first rule would seem to apply but is in conflict with the latter rule.

Second Analysis: Review of Video and Think-Aloud Data. In the second analysis, we reviewed video of user sessions and transcripts of user think-aloud data for critical incidents which indicated errors or other usability problems. We defined critical incidents in the video as incidents in which users created a rule with one of the errors indicated in the first analysis but subsequently corrected it. We defined critical incidents in the think-aloud data as incidents in which users expressed confusion, concern, or an interface-relevant suggestion. Once we had identified critical incidents, we classified them according to the error or usability problem that they indicated. While critical incidents were caused by a variety of SPARCLE-specific usability problems, only those incidents relevant to the five general policy-authoring rule errors identified in the first data analysis are reported here. There were no critical incidents that indicated general rule errors not already identified in the first data analysis; thus, the second data analysis simply served to confirm the results of the first through an independent data stream.

Below are some typical examples of user statements classified as critical incidents, followed by the error under which we classified them in parentheses:

- "I don't want to have to write out a long list of types of information without being able to find a variable that represents that information to be able to label that information. In this case the label might be personal information defined to include customer name, address, and phone number." (Group ambiguity)
- "I'm not sure how to do negations in this template." (Negative rule)
- "It says I must specify at least one purpose, and I say, 'why do I have to specify at least one purpose?' " (Missing element)

5 Results

Results from the two data analyses, combined by adding the unique error instances found in the second analysis to those already found in the first analysis, are shown in Fig. 2. The errors in Fig. 2 are listed according to total frequency of occurrence, and within each error, are broken down by the task scenario in which they occurred. Since 2 of the 3 task scenarios were presented to each of 12 participants, there were 24 total task-sessions, 8 of each of the three scenarios. Thus, for example, the "group ambiguity" bar in Fig. 2 indicates that group ambiguity errors occurred in 11 of 24 task-sessions, including 5 of 8 "DrugsAreUs" sessions, 1 of 8 "Bureau of the Public Debt" sessions, and 5 of 8 "First Finance" sessions.

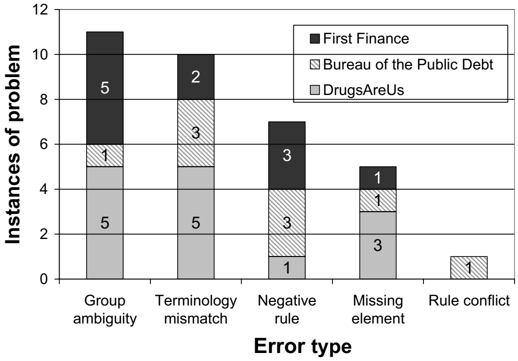


Fig. 2. Results of first and second data analyses, showing instances of five types of errors, broken down by task scenario in which they were observed. There were 24 total task-sessions, 8 sessions for each of the three tasks.

Of the five errors, group ambiguity errors were observed most frequently; a total of 11 instances of group ambiguity errors were found. The other errors, in order of frequency of occurrence, were terminology mismatches, negative rules, missing elements, and rule conflicts.

6 Discussion

The errors that participants made in this study suggest the five general policy-authoring usability challenges listed in the introduction to this paper. The challenges arise from inherent difficulties in the task of articulating policies; however, good interface design can help users overcome these difficulties. Group ambiguities suggest that users have a need for composite terms, but also need support to define these terms unambiguously. Terminology mismatches suggest the need for the interface to enforce, or at least provide some support for, consistent terminology. Negative rules are not necessary in SPARCLE's default-deny policy-based system, so users' attempts to write negative rules suggest that they did not know or did not understand the default rule. Missing elements are caused by users' failure to understand or remember the requirement for certain rule elements. Finally, rule conflicts are a known problem that a good interface can help address.

The identification of these five policy-authoring usability challenges is the primary result of this study. One of these challenges, communicating and enforcing rule structure, had already been anticipated, and SPARCLE's rule guide on the Natural Language Authoring page was designed to guide policy authors to write rules with correct structure. Rule conflicts, a well-known problem in policy-authoring domains, were also anticipated. SPARCLE, in fact, was designed to prevent rule conflicts by using a default-deny system and requiring that policy authors only write rules that mapped exclusively to ALLOW. It is only in the optional condition element that a conflict is possible in SPARCLE rules, so it was not surprising that only one rule conflict was observed in the present study. The remaining three usability challenges observed were largely unanticipated when SPARCLE was designed. Because of the fairly common failure to anticipate some of the five challenges, in SPARCLE and in other designs discussed above in the Introduction and below in the Related Work, the identification of these challenges is a significant contribution.

Before discussing the usability challenges observed, it is worth considering one methodological concern. Some of the errors revealed, particularly group ambiguities and negative rules, and the frequency with which they were observed in the present study, may have been influenced by the specific task scenarios presented to users. However, errors, except for the one instance of a rule conflict, are distributed fairly evenly across the three tasks, so it does not appear that any one task was responsible for eliciting a particular error type. Furthermore, the task scenarios were written based on experience with real enterprise policy authors and real enterprise scenarios, so the errors revealed are very likely to come up in the real world. However, the frequency values reported here should not be taken as necessarily indicative of the relative frequency of occurrence of these errors in real-world policy work.

Having identified five usability challenges for policy-authoring domains, it is worth discussing how these challenges might be addressed. Each challenge is discussed in turn below.

6.1 Supporting Object Grouping

Group ambiguities were caused by users not understanding what terms covered what other terms. Many solutions already exist to help users with tasks that involve grouped elements. Perhaps the most prominent grouping solution is the file system hierarchy browser. A hierarchy browser allows users to create groups, name groups, add objects to and remove objects from groups, and view group memberships. Hierarchy browsers may often be appropriate for policy-authoring tasks. However, hierarchical grouping may not always be sufficient. In file permissions, for instance, system users often belong to multiple, partially-overlapping groups. Any of a variety of means for visualizing sets or graphs may be useful here; also needed is a means for interacting with such a visualization to choose terms to go into policy rules. What visualizations and interaction techniques would best support grouping for policy authoring is an open problem.

6.2 Enforcing Consistent Terminology

Ambiguous terminology is nearly inevitable, but there are certainly ways to mitigate the most common causes, which, in this study, included typos and users' forgetting what term they had previously used to refer to a concept. A spell-checker could go a long way toward eliminating many typos, like “socail security number”, in which “social security number” was obviously intended. A domain-specific thesaurus could help resolve abbreviations, aliases, and cases in which the same object necessarily has multiple names. For example, a thesaurus could indicate that “SSN” expands to “social security number”, and that “e-mail”, “email”, and “email address” all represent the same thing. A display of previously used terms for an object might help users remember to reuse the same term when referring to that object again; SPARCLE's structured lists are an example of such a display. In some policy-authoring domains, the terminology problem may be resolved for the policy author by pre-defining terms. For example, in file permissions, the actions that can be performed on a file are typically pre-defined by the file system (e.g., read, write, and execute).

6.3 Making Default Rules Clear

Showing default rules may be a trivial matter of including the default rule in interface documentation or in the interface display itself. However, the concept of a default rule and why it exists may itself be confusing. One method of illustrating the default rule to users is to present a visualization showing what happens in unspecified cases [4]. SPARCLE includes such a visualization, although it is not onscreen at the same time a user is authoring rules [12].

6.4 Communicating and Enforcing Rule Structure

SPARCLE already does a fairly good job of enforcing rule structure. Participants in the present study recovered from forgotten purpose elements in 2 out of 5 cases due to SPARCLE's prominent display of the phrase "None Selected" as the purpose element when no purpose was specified; a corresponding "Missing Purpose" error dialog also helped. Other interaction techniques like wizards, in which users are prompted for each element in turn, would likely get even higher rates of correct structure. Which of these techniques or combination of techniques leads to the fewest missed elements will be the subject of future work.

6.5 Preventing Rule Conflicts

Rule conflicts were rare in this study; the one observed conflict can be attributed to a lack of awareness about the semantics of the condition element. However, rule conflicts have been shown to be a serious usability problem in past work in other policy-authoring domains [10,2,4]. Clearly, interfaces need to make users aware of conflicts, and if possible, show them how conflicts can be resolved.

Rule conflicts have been the focus of some non-interface-related theoretical work [13,14]; algorithms exist for detecting conflicts in a variety of policy contexts. This work could be leveraged by interface designers. However, the means for presenting policy conflicts to authors have yet to be evaluated. A few visualizations and interfaces have attempted to do this, such as those discussed below in the Related Work [10,2,4], but it is not clear whether they succeed at conveying conflicts, the need to resolve them, and the means to resolve them to users.

7 Related Work

Although the present study looked for usability challenges in just one policy-authoring domain, related work in other domains confirms that the usability challenges identified here are general problems encountered in a variety of policy-authoring domains. However, past work has only identified the challenges as unique to specific domains, rather than as part of the more general policy-authoring problem.

A need for supporting groups of element values has been found in several domains. Lederer et al. found a need for supporting groups of people in a user interface for setting location-disclosure policies in a pervasive computing environment so that policies could be scaled to environments with many people [15]. They present an interface for setting location-disclosure policies, but mention support for grouping as future work. The IBM P3P Policy Editor is a policy-authoring interface that uses hierarchical browsers to show users how Platform for Privacy Preferences (P3P) element values are grouped [16]. Zurko et al.'s Visual Policy Builder, an application for authoring access-control policies, allowed authors to create and label groups and to set constraints on groups to prevent conflicts due to group overlaps [17].

Finding good terminology and using it consistently has long been recognized as a problem for virtually every user interface [18]. In the policy-authoring area, Cranor et al. acknowledged the difficulty of finding comprehensible terminology in developing an interface for allowing users to specify what privacy practices they prefer in websites with which they interact [5]. Good and Krekelberg found that the use of four different terms for the same folder confused users in an interface for specifying shared folders in the KaZaA peer-to-peer file-sharing application [3].

Communicating default rules has been shown to be a problem in setting file permissions by both Maxion and Reeder [4] and Cao and Iverson[2]. Cranor et al. also discuss their efforts to come up with an appropriate default rule and communicate that rule to users [5]. Good and Krekelberg found that that KaZaA did not adequately communicate default shared files and folders to users [3].

The above-referenced independent studies of file-permissions-setting interfaces, Maxion and Reeder [4] and Cao and Iverson [2], also found that users have difficulty detecting, understanding, and correcting rule conflicts in access control systems. Zurko et al. considered the problem of conveying rule conflicts to users in their design of the Visual Policy Builder [17]. Al-Shaer and Hamed acknowledge the difficulties that rule conflicts cause for authors of firewall policies [10]. Besides human-computer interaction work, some theoretical work has acknowledged the problem of rule conflicts and found algorithms for detecting conflicts in policies [13,14].

Lederer et al. report five design pitfalls of personal privacy policy-authoring applications that do not include the same usability challenges listed here, but do raise the important question of whether configuration-like policy-authoring interfaces are needed at all [19]. They argue that in personal privacy in pervasive computing environments, desired policies are so dependent on context, that users cannot or will not specify them in advance. While their argument is undoubtedly correct for some domains, there remains a need for up-front policy authoring in many situations: the system administrator setting up a default policy for users, the policy maker in an enterprise writing a privacy policy to govern how data will be handled within the organization, and even the end-user who does not want to be bothered with constant requests for access, but prefers to specify up-front what access is allowed.

8 Conclusion

In order to be usable, policy-authoring interfaces, which are needed for a wide variety of security and privacy applications, must address the five usability challenges identified in the user study described in this paper: supporting object grouping, enforcing consistent terminology, making default policy rules clear, communicating and enforcing rule structure, and preventing rule conflicts. Some of these issues have been addressed before in domain-specific policy-authoring interfaces and elsewhere, but all might benefit from novel, general interaction techniques. As more policy authoring interfaces for users are created to fit the variety of applications that depend on accurate policies, researchers and designers would benefit from considering the five usability challenges discussed in this paper and creating innovative interaction techniques to address them.

References

1. Karat, J., Karat, C.-M., Brodie, C., Feng, J.: Privacy in information technology: Designing to enable privacy policy management in organizations. International Journal of Human-Computer Studies 63(1-2), 153–174 (2005)
2. Cao, X., Iverson, L.: Intentional access management: Making access control usable for end-users. In: Proceedings of the Second Symposium on Usable Privacy and Security (SOUPS 2006), New York, NY, pp. 20–31. ACM Press, New York (2006)
3. Good, N.S., Krekelberg, A.: 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), New York, NY, April 2003, pp. 137–144. ACM Press, New York (2003)
4. Maxion, R.A., Reeder, R.W.: Improving user-interface dependability through mitigation of human error. International Journal of Human-Computer Studies 63(1-2), 25–50 (2005)
5. Cranor, L.F., Guduru, P., Arjula, M.: User interfaces for privacy agents. ACM Transactions on Computer-Human Interaction 13(2), 135–178 (2006)
6. U.S. Senate Sergeant at Arms: Report on the investigation into improper access to the Senate Judiciary Committee's computer system (2004), available at `http://judiciary.senate.gov/testimony.cfm?id=1085\&wit_id=2514`
7. Karat, C.-M., Karat, J., Brodie, C., Feng, J.: Evaluating interfaces for privacy policy rule authoring. In: Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems(CHI 2006), New York, NY, pp. 83–92. ACM Press, New York (2006)
8. Lederer, S., Mankoff, J., Dey, A.K., Beckmann, C.P.: Managing personal information disclosure in ubiquitous computing environments. Technical Report UCB-CSD-03-1257, University of California, Berkeley, Berkeley, CA (2003), available at `http://www.eecs.berkeley.edu/Pubs/TechRpts/2003/CSD-03-1257.pdf`
9. Ashley, P., Hada, S., Karjoth, G., Powers, C., Schunter, M.: Enterprise Privacy Architecture Language (EPAL 1.2). W3C Member Submission 10-Nov-2003 (2003), available at `http:www.w3.org/Submission/EPAL`
10. Al-Shaer, E.S., Hamed, H.H.: Firewall Policy Advisor for anomaly discovery and rule editing. In: Marshall, A., Agoulmine, N. (eds.) MMNS 2003. LNCS, vol. 2839, pp. 17–30. Springer, Heidelberg (2003)
11. Ericsson, K.A., Simon, H.A.: Protocol Analysis: Verbal Reports as Data. Revised edn., MIT Press, Cambridge, MA (1993)
12. Brodie, C., Karat, C.M., Karat, J.: An empirical study of natural language parsing of privacy policy rules using the SPARCLE policy workbench. In: Proceedings of the 2006 Symposium on Usable Privacy and Security (SOUPS 2006), New York, NY, July 2006, pp. 8–19. ACM Press, New York (2006)
13. Agrawal, D., Giles, J., Lee, K.-W., Lobo, J.: Policy ratification. In: Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2005), Los Alamitos, CA, June 2005, pp. 223–232. IEEE Computer Society Press, Los Alamitos (2005)

14. Fisler, K., Krishnamurthi, S., Meyerovich, L.A., Tschantz, M.C.: Verification and change-impact analysis of access-control policies. In: ICSE 2005, pp. 196–205. IEEE Computer Society Press, Los Alamitos (2005)
15. Lederer, S., Hong, J.I., Jiang, X., Dey, A.K., Landay, J.A., Mankoff, J.: Towards everyday privacy for ubiquitous computing. Technical Report UCB-CSD-03-1283, University of California, Berkeley, Berkeley, CA (2003), available at http://www.eecs.berkeley.edu/Pubs/TechRpts/2003/CSD-03-1283.pdf
16. Cranor, L.F.: Web Privacy with P3P. O'Reilly, Sebastopol, CA (2002)
17. Zurko, M.E., Simon, R., Sanfilippo, T.: A user-centered, modular authorization service built on an RBAC foundation. In: Proceedings 1999 IEEE Symposium on Security and Privacy, Los Alamitos, CA, May 1999, pp. 57–71. IEEE Computer Society Press, Los Alamitos (1999)
18. Molich, R., Nielsen, J.: Improving a human-computer dialogue. Communications of the ACM 33(3), 338–348 (1990)
19. Lederer, S., Hong, J., Dey, A.K., Landay, J.: Personal privacy through understanding and action: Five pitfalls for designers. Personal and Ubiquitous Computing 8(6), 440–454 (2004)