Abstract—Policies which address security and privacy are pervasive parts of both technical and social systems, and technology to enable both organizations and individuals to create and manage such policies is seen as a critical need in IT. This paper describes policy authoring as a key component to usable privacy and security systems, and advances the notions of policy templates in a policy management environment in which different roles with different skill sets are seen as important. We discuss existing guidelines and provide support for the addition of new guidelines for usable policy authoring for security and privacy systems. We describe the relationship between general policy templates and specific policies, and the skills necessary to author each of these in a way that produces high-quality policies. We also report on an experiment in which skilled programmers were asked to author policy templates using a prototype template authoring system we developed.

I. INTRODUCTION

Policies are crucial to security and privacy domains. Policy-driven systems are important because they allow a level of dynamic customization that does not require a new cycle of requirements engineering and development. Policies are especially important in security and privacy. For example, as noted by Cheswick et al., “The single most important factor of your firewall’s security is how you configure it” [1]. This idea extends to any system that enforces a policy, the system will enforce the specified policy so it is crucial that the specified policy matches the intended policy.

In IT, there is a general notion of policy-based systems as those whose behavior is guided by rules of the general form “If condition then action.” Collections of rules are considered policies, and policies can be developed for various aspects of system behavior. In social systems, organizations have policies covering proper conduct of people and effective use of resources. Information technology systems have policies which govern access to resources aimed at protecting the integrity and confidentiality of the information and resources. Individuals have policies guiding their behavior towards others formed with the intention of guiding how they live their lives. These policies might be expressed in natural language text (common in organizations), in executable code (common for IT systems), or might be implicit (common for individuals). Such policies might be seen as including high level guidance (e.g., “to insure a safe workplace”) and more specific operational rules (e.g., “don’t run with scissors”).

We see several aspects common across a wide range of policy types in technical and social systems. First, high level policies – generally expressed in human language – are refined into operational rules while attempting to keep the intent of the high level policy. This process is difficult – often subject to differences in interpretation or context. Second, the existence of multiple, possibly conflicting, policies must be accommodated. This process is also difficult, as comparison across policies requires detailed understanding of the meaning of each policy rule and is rarely straightforward. For human or technology systems there is a resulting gap – sometimes referred to as the gulf of execution [2] – between human intentions and technology capabilities. We believe developing approaches to closing the gulf of execution would be valuable. For example, most organizations store sensitive business and personal data in heterogeneous server systems. They do not have a unified way of defining or implementing security and privacy policies regarding the storage and use of that data throughout their organization. Changing legal requirements, social pressures and technologies are making these issues increasingly critical to organizations and society at large.

While our research can apply to policies in many areas, we are particularly interested in security and privacy policies. There are several reasons for this focus. First, there are a growing number of strict security and privacy audit and compliance requirements for healthcare, banking/finance, and government. This creates a practical need for improving the management of such information through policy-based systems. Second, there is considerable need for improving the management of such information through policy-based systems. Second, there is considerable need for improving the management of such information through policy-based systems. Third, privacy – when viewed as appropriate use of information – relies on the security of that information in a system, and it is difficult to talk about privacy without considering security. Our principal research objective is to create an integrated privacy and security policy management framework which builds on the commonalities between the two. This includes mechanisms and tools for supporting policy authoring which have grown out of previous
work centered on privacy policies [4].

We see the policy authoring challenge as consisting of both providing a mechanism for expression of policy intent by domain experts, and providing a mechanism for translating human statements to IT systems. To illustrate the policy authoring challenge, consider the following authoring policies:

1. Healthcare staff can forward patient medical information for the purpose of national medical research if the information is anonymized.
2. Doctors can access laboratory results.

Each of these policies authorizes an action if a condition is met. The first policy is considered a privacy policy, while the second a security access control policy. While it might be claimed that these policies can be understood by people, the task of making them enforceable by IT systems is complex. Basic elements of the policies might be extracted (e.g., the users, actions, data, conditions and purposes of the privacy policy in 1), but mapping them to attributes that a system can interpret as intended by the human authors requires IT knowledge and skills not possessed by many people responsible for establishing such policies in organizations. Context dependent definitions of attributes like “doctors”, “national medical research”, or “anonymized” need to be tied to attributes an IT system can process.

In this paper, we outline our directions and progress to date, along with remaining challenges in developing a policy authoring framework. We describe how we view policy authoring, and extend our notions of policy authoring to include the role of template author. This additional role captures the need to map human concepts to system attributes and requires skills beyond the business knowledge generally associated with policy formulation in organizations. The framework is presented with examples of policies involved in security access control and privacy. Our intention is to provide a framework which is comprehensive enough to enable collaboration among researchers working on policy development and implementation in different areas of security and privacy.

II. RELATED WORK

Policy authoring is important to the usable security community and has received a fair amount of research attention. To frame the discussion of usable policy authoring, we categorize the prior research into work that addresses general policy authoring and domain-specific policy authoring interfaces. We also discuss research from the computer policy community.

A. General Usable Policy Authoring

SPARCLE is a policy management workbench [5]. The initial prototypes focused on privacy policies [4]. Later work demonstrated the extensibility of the workbench to other domains [6]. Using SPARCLE, the policy author wrote policies using their choice of either guided natural language or structured entry. The ability to switch between the two representations was a feature in SPARCLE. The interface for guided natural language displayed a syntax guide above the text area where the policy author typed the policy. The guide increased their ability to write correctly structured policy statements. A natural language grammar was used to extract the policy elements from the natural language. To author a policy using structured entry the policy author selected values for policy elements from predefined lists. Empirical studies showed when policy authors used guided natural language or structured entry, they produced higher-quality policies compared to unguided natural language [5].

Reeder et al. researched common policy authoring errors [7]. A set of guidelines for designing usable security and privacy policy authoring tools were produced using the results of the study. The guidelines were based on an analysis of the errors observed and suggest ways to eliminate the errors. The guidelines included: support object grouping, enforce consistent terminology, make default rules clear, communicate and enforce rule structure, and prevent rule conflicts.

The Expandable Grid is an interactive matrix that displays a visualization of the policy. Policy elements are chosen for the x and y-axis and the intersection holds the policy decision for the two attributes. The policy decision is captured in a graphical representation that may differ based on the policy domain. The Expandable Grid is an alternative to the usual list-of-rules approach most policy systems take. An empirical study comparing Expandable Grids and the Windows XP file permissions interface suggests the Grid is more usable [8]. Expandable Grids has also been extended to P3P policies [9], an empirical study of the two suggests Expandable Grids does not improve usability beyond the level of usability achieved by P3P policies expressed in natural language [10].

B. Domain Specific Usable Policy Authoring

Role-based access control (RBAC) is a commonly used mechanism for expressing policies for access to system resources [11]. Adage is an authorization service built on RBAC in which the primary focus of the work was designing a policy-management GUI for administrators [12]. The GUI was iteratively designed with a user-centered design techniques, it allowed the user to group objects and have them represented by a single object in the interface. The direct manipulation features and the representation of object groupings for subjects and targets were shown to be beneficial to users.

ESCAPE was designed for managing file access on the web [13]. The interface allowed users to configure permissions implicitly through their actions. This was accomplished by granting read privileges to the users the content owner announced the new material to. Salmon is a policy interface designed for file access control [14]. The interface was designed to reduce the number of errors users made when authoring and reading file permissions. It was designed to provide the user with a more accurate depiction of the state of the policy.

IAM allows users to specify the effective policy rather than individual rules [15]. IAM takes the effective policy as input and suggests ways of achieving the specified policy. This increases usability by allowing the user to specify the end result without having to figure out the individual rules.
necessary. Grey is a physical access control system, a primary use of it is managing access policies for rooms on a university campus [16]. The research on the Grey system has explored the implications of allowing more fine grained policies than can be achieved with the usual lock and key.

In the effort to design a privacy agent for P3P policies, Cranor et al. discuss the issue of designing an interface to allow users to express their privacy preferences [17]. Most problems were related to finding the appropriate vocabulary to express the complex language and concepts found in privacy policies, and structuring the interface to group common elements.

Firmato was designed to address the usability of managing firewall policies [18]. The tool allows policy managers to abstract the policy from the actual mechanisms and network topology, it also allows them to use named objects to represent groups of IP addresses and port numbers. A user interface was designed to generate a visualization of the topology and services offered by the rule set.

C. Policy Languages

In the area of policy-driven systems, policy languages that allow maximal flexibility are a popular research direction. Ponder is a strongly typed, declarative, object-oriented language for specifying network management policies for heterogeneous systems [19]. Flexible policy languages are useful to demonstrate that a wide range of enforceable policies can be specified. In practice, their usability is limited by whether a usable policy authoring interface is available that translates the user specified policies into the language. Keynote and XACML are other examples of policy languages [20], [21].

There are also attempts at automatic policy configuration with the goal of eliminating the policy author. For example, one is a machine-learning approach to role-learning in an RBAC system [22]. It is important to note that in this area even successful attempts require an initial human authored policy. The need for usable policy authoring remains.

III. TEMPLATE-BASED POLICY AUTHORING FRAMEWORK

In our previous research, we define a template-based policy authoring framework and empirically evaluate a policy template authoring prototype [23]. The core idea of the framework is that policy authoring is a complex task which requires user participation and expertise from more than one area. The complexity and the required areas of expertise imply the need to divide policy authoring into subtasks. Identifying the policy authoring roles and associated skills necessary for the users that will complete the tasks is also needed. Our template-based framework defines three subtasks and describes the user who will accomplish each one. Each subtask in the framework is paired with a user interface to help the user complete their tasks. The roles are policy author, template author, and policy element author.

The policy author creates policies from templates, which are structured natural language representations of policy statements. The policy author has knowledge of the policy domain and is assumed to have a natural language understanding of the intended policy. Policy authors are not expected to have in-depth technical knowledge of how the policy is implemented.

The template author creates templates by combining predefined policy elements into abstract policy statements. The templates are abstractions of the policies the policy author will create. The template author is assumed to be familiar with the policies that may be written but does not need to know the exact policies that will be authored. Template authors also have some knowledge of the technical aspects of the policy domain. Templates are similar to the grammars used in natural language parsing and form the syntax for the policy statements.

The policy element author creates policy elements for the domain specific objects that will be used in policy statements. The policy element author has a thorough understanding of the technical aspect of the policy domain. They must have the knowledge to specify a policy element for the domain specific objects. The objects include users, resources, actions, system variables that can be used in conditions, and security mechanisms. Policy element authors must also specify the relationships between the objects. Policy elements form the vocabulary that will be used for the policy statements.

The framework’s authoring process is iterative. The policy author can request new templates or policy elements. Policy authors can also adjust their policies based on the available templates and policy elements. And, the template author can request new policy elements. The policy author’s ability to request new templates eliminates the expectation that the template author must predict every policy that will be written.

The following example is an access control template:

\[
\{\text{Subject}\} \text{ can } \{\text{Action}\} \{\text{Target}\} \text{ if } \{\text{Condition}\}.
\]

The following policy is an example from the medical domain. It demonstrates a policy that a policy author could write using the template.

\[
\{\text{Doctors}\} \text{ can } \{\text{read}\} \{\text{name}\} \text{ or } \{\text{current medication}\} \text{ if the } \{\text{patient}\} \text{ has been } \{\text{admitted}\}.
\]

This example demonstrates how domain specific values replace the policy elements to form policy statements from a template. An access control policy specifies when someone can perform an action on the specified resource. Subject, action, and target are required elements for an enforceable access control policy, a condition is optional.

Prior research supports the design of a template-based policy authoring framework [5]. Empirical evaluations of SPARCLE show natural language and structured-entry methods are both more usable compared to unguided natural language. A robust grammar, for each domain, is required to implement SPARCLE’s natural language features. In addition to the grammar, structured lists must be generated for each domain. At this time it is more efficient to implement a method of generating structured lists than it is to implement a sufficiently robust natural language grammar for each policy domain [5]. Since the usability of the two entry methods is comparable, and superior to unguided natural language, the template-based policy authoring framework focuses on satisfying the requirement to generate domain specific lists.
A. Empirical Evaluation of a Template Authoring Prototype

Since template-based authoring framework calls for a role not specifically considered in previous research – the template author – we conducted a user study specifically to evaluate the usability of a template authoring prototype and to observe how users understand the three roles defined. We argue that policy authoring from natural language guides has been studied previously [4], [7] and that policy element authoring is a standard programming task not unique to policy authoring. A detailed description of the user study and the primary results, those related to template authoring, are reported in another paper [23].

In the template authoring user study, twenty participants completed two template authoring tasks using a template authoring prototype. The participants were technical users with limited policy experience. Each task had a short description of a scenario and a list of policies. The participant was then asked to create templates that a policy author could use to author the policies linter. One task involved a web merchant and access to their order database. The other task was for a fictitious social networking site where users could share personal information.

Before beginning the tasks, the participants were given instructions that explained the relationship between the three roles (policy element author, template author, and policy author). Then the participants completed a policy authoring task to give them experience authoring a policy using a template, and to help them understand the difference between the template author and policy author roles. The participants were asked to think-aloud during the template-authoring tasks, completed post-task questionnaires, and answered debriefing questions. In addition, the study coordinator took notes from an observation room while the participants completed the tasks. This brief description of the user study is included here because some of the data collected provide insight specific to security and privacy policy authoring. The relevant data will be presented in the discussion of our suggested guidelines.

B. Policy Authoring Tool Extensibility

The template-based policy authoring framework was designed as a general policy authoring framework. By providing interfaces for each role that is designed and evaluated with human-centered processes, and defining the expectations of the user filling each role, the framework can be extended to any policy domain. Two policy authoring tools discussed above – SPARCLE and Expandable Grids – have been designed with the goal of being extensible to any security and privacy policy domain.

SPARCLE was developed for privacy policies, but can be extended to other policy domains if a natural language (NL) grammar is written and if the corresponding elements of the structured lists are defined. NL parsing of NL policy statements is a valuable goal but we have found development of new grammars to be more difficult than specification of policy-templates for similar policies. Some of this has to do with the usability of NL grammar tools which are complex to learn and generally tied to particular language parsers with complex recognition performance characteristics. Some of the difficulty is associated with the desirability of maintaining a structured authoring mechanism to augment the NL approach. For SPARCLE, a new policy type required a new structured entry dialog in addition to the new grammar.

Expandable Grids was also designed to be applied to many security and privacy domains. Empirical results suggest Expandable Grids is a usable policy authoring interface for file access control policies [8]. However, results from a user study of applying Expandable Grids to P3P privacy policies suggest the tool is not more usable than P3P policies expressed in natural language [10]. A description of Expandable Grid’s extensibility describes a customization process for tailoring the interface for new policy domains [24]. The customization process has several design steps: designing effective symbolic representations of policy decisions, choosing the appropriate values to represent on the two available axes, choosing effective short policy element names, and deciding how to incorporate metadata given limited space. The difference in performance between file access control and P3P policies suggests applying the customization process is not trivial, though the effectiveness of Expandable Grids depends on a successful implementation of these features.

The extensibility of a policy authoring tool should not rely on user interface design changes that must be tested with empirical evaluation for each new policy domain. This is not feasible given the continual growth of the number of policy-driven systems and the expectations on end-users to manage and understand policies. We believe that the template-based authoring approach provides a better approach through better isolation of the elements that define the specific policy context. Policy authors express natural language rules through a flexible structured entry mechanism which does not require extensive code redevelopment for different policies. Policy element authors write code specific to the policy using mechanisms standard to the implementation. Policy templates provide the appropriate interface between the two.

IV. OPTIMIZING FOR SECURITY AND PRIVACY POLICY AUTHORING

Policy-driven systems are used outside the security and privacy realm but here we focus the discussion on security and privacy policy authoring. There is a need to identify design requirements for security and policy authoring tools, but in this early stage of the research it may be best to start by specifying guidelines. We begin by presenting existing guidelines for security and privacy policy authoring. Then we discuss how the template-based policy authoring framework meets the existing guidelines. We also suggest new guidelines based on prior research, where applicable we present empirical data from an evaluation of a template authoring prototype that supports the new guidelines.

A. Guidelines for Security and Privacy Policy Authoring

There are unique usability challenges in authoring security and privacy policies [7]. The following guidelines have been
identified for a usable policy authoring system:

- Support object grouping
- Enforce consistent terminology
- Communicate and enforce rule structure
- Make default rules clear
- Prevent rule conflicts

A template-based policy authoring framework satisfies these guidelines, with the exception of making default rules clear and preventing rule conflicts. Policy elements address the first two guidelines. Policy elements are displayed using metadata from their definitions that show the relationships between elements. Users are able to view the groupings by clicking on policy elements for more information or hovering over elements for more detail. These features make object groupings clear [23]. Policy elements also enforce consistent terminology. The template authoring prototype was designed so the template author can easily search for and discover policy elements. The policy author is presented with a limited set of values for each policy element in the template and will not have the opportunity to introduce inconsistent terminology.

Policy templates communicate and enforce rule structure to the policy author. The template author has the knowledge and experience to know what types of policies should be written. Using this knowledge they create templates that represent the natural language policy statements that should be written. The template authoring interface also stores examples of master templates. Master templates can be modified to create new templates. Authoring policies from templates guarantees the policy author will only be presented with valid rule structures. If the policy author needs to write a policy that cannot be written from the existing templates, they communicate this to the template author. The template author has more experience with the policy domain and can help the policy author refine the rule to fit an existing template, or can create a new template using the policy author’s feedback.

The template author and the policy author need to understand whether the default rule of the system is default-allow or default-deny. In a default-deny system, the policy author must understand that only those actions explicitly allowed will be allowed and that it is unnecessary to write deny policies. For default-deny, the template authoring interface should communicate to the template author that deny templates are unnecessary and may introduce conflicts. Additionally, the interface must communicate to the policy author why templates to author deny policies are not available.

Policy conflict detection, prevention and resolution are active areas of research [25], [26], [27]. One known method of preventing conflicts is to only allow the policy author to create deny-policies or allow-policies, but to not allow both types of policies in the same policy set. This prevents instances where a specific user is denied permission, while a group the same user is a member of is granted the permission. User interface features that present conflict detection and resolution feedback should be added as research in the area continues to progress. It may be more effective to incorporate the feedback using visualization tools [28], [8].

In addition to the existing guidelines [7], we propose the following:

- Support appropriate limitation of expressivity
- Communicate risk and threats
- Provide access to metadata

We demonstrate the importance of each of the new guidelines, provide support from the literature, and present relevant data from the template authoring user study.

B. Support Appropriate Limitation of Expressivity

The guideline to support appropriate limitation of expressivity is related to limiting the set of policies that can be written. This guideline affects usability and, in turn, also affects security. In a template-based policy authoring framework, the responsibility of limiting expressivity falls on the policy element author or the template author.

To illustrate when it would be desirable to limit the policies that can be written, imagine the policy author plans to write:

Order processors can read the order fields in the database.

With this template:

\{\text{Internal Users}\} \text{can} \{\text{Action}\} \{\text{Database Record Fields}\} \text{in the database.}

In this template, the policy element \{\text{Internal Users}\} has the values: database administrators, order processors and customer service. \{\text{Action}\} has the values: read, write and edit. And the policy element \{\text{Database Record Fields}\} has the values: all fields, order fields and non-private fields.

The policy author could create the policy by choosing the following values for the policy elements:

\{\text{Internal Users}\} = \text{order processors}
\{\text{Action}\} = \text{read}
\{\text{Database Record Fields}\} = \text{order fields}

But, the policy author could also write this policy from the template:

Order processors can edit the all fields in the database.

Even without knowing the details of the policy domain or the threat model, one can assume based on the principle of separation of duties (or separation of privileges) that the second policy is undesirable [29]. Users in the order processors group should be able to view the order, but they should not be able to modify the order. From a security standpoint, it’s likely that only an administrator should have direct access to modifying the database. Or, if it is acceptable for other roles to modify the database, the template should force a policy where logging or another security mechanism is enabled.

In the template authoring user study described earlier, nine of the twenty participants expressed concerns about template flexibility while authoring templates. For example, one participant commented, “But a policy author could say the order processors can access all fields in the database, I want it more specific.” Another participant spent a few extra minutes checking their work and looking over their templates for the
task. Finally, they expressed their concern, “I think I’m done but I don’t like what I have ... they can write what’s on the paper but also a lot more than that. Is that right? They can write a lot of policies that probably shouldn’t be allowed.”

When participants expressed feelings related to this theme they were reminded that the task was to generate policy templates that a policy author could use to write the sample policies on their task sheet. After the participant completed the tasks, their concerns were discussed in the debriefing session. One participant described their perspective in terms of the user roles when they talked about how they chose the policy elements to use, “The template author should limit the choices for the policy author. Depending on the policy author, I would choose a different level of attributes. Like for a new person, I’d want to limit the choices.” It was unexpected that the participants would embrace their role as the template author in this way and consider the experience of the policy author as they created the templates. It was most certainly welcome though, as their concerns provided additional insights on the relationships between the roles especially for security and privacy domains.

It might seem that specifying the additional relationships between policy element values is an unacceptable amount of work to add to the policy authoring process. Within a template-based policy authoring framework, however, the relationships between values is specified once then in the future is uniformly enforced by the interface such that the template author is held to the restriction and the policy author is as well. If such a limitation is necessary, it is better to specify it once rather than rely on the policy author to implicitly enforce it each time they author new policies or modify existing policies.

The responsibility of specifying these restrictions could be assigned to the policy element author, the template author, or both. If the policy element author must specify the additional relationships their user interface must support such features and make it clear how the changes affect the template author and policy author. It may be best for the policy element author to assume the task because they are familiar with the technical details of the system and have the best understanding of the security mechanisms. However, the template authoring interface should also provide a way to restrict the values that can be combined to create a policy. As participants suggested, when the policy elements are put together to form a template there may be combinations that should not be allowed.

We have discussed how building restrictions into the templates may reduce the policy author’s ability to author insecure or undesirable policies; we have not addressed how this will affect the policy author’s user experience. The end result of this guideline influences the policies the policy author can create from a template. This will change how they are able to interact with some templates. Prior work discusses the user frustration that results when their intended policy cannot be authored, and their willingness to work around policies when this happens [30]. This must be addressed in the policy author’s user interface by making it clear why certain policies cannot be written considering when users do not understand the security reasons behind a mechanism they are more likely to subvert it [31].

C. Communicate Risk and Threats

This guideline represents the need to capture and communicate information about risk and threats. Computer security can be thought of in terms of requirements, security policies, and mechanisms [32]. The goal of computer security is to prevent undesirable events from occurring. The requirements specify the undesirable events and the policies represent decisions of how to satisfy the requirements by stating what is and is not allowed. The mechanisms enforce the policy. In this context, policies can be technical or procedural policies, security mechanisms enforce the technical policies and users in the system may need to follow-up on items that get flagged by the automated policies.

Presumably the requirements are defined based on risk analysis [33]. A quantitative risk analysis is best [34]. But even the results of a rudimentary or qualitative evaluation can benefit the policy author. The policy author may be better equipped to author high-quality policies if they understand the relevant threats and risks. The inclusion of this additional information could also be of use to an audit.

Applying this guideline to the template-based policy authoring framework means communicating risk and threat information between the three roles. There should be a mechanism such that the policy element author can indicate to the template author that specific elements should be used with certain considerations in mind.

For example, if the policy author was writing access control policies for internal and external users, there should be a way for the policy element author to indicate that the templates involving external users should make careful use of specific security mechanisms when access is allowed. Similarly, the templates could include information to make the policy author aware of the security implications of different policy element values. In most policy authoring interfaces all settings are treated the same by visually presenting the options uniformly. It may benefit users if policy decisions with a higher impact were presented in a different manner.

For a simple way to implement this guideline, consider the Windows file permissions interface as an example. The security implication of allowing full privileges to a group with many members may be quite different than giving one user read access. But, the user experience of clicking the checkbox to delegate the permissions is the same. The optimal way to manage the user interactions in this case is a subject in need of further research but the first step raising the question of whether the interaction should be different.

This guideline would be useful for firewall policy management interfaces. Correct firewall policy management is crucial since an incorrect policy could deny important traffic or allow malicious traffic that should be blocked. In a quantitative evaluation of firewall configurations, 37 rule sets were evaluated to measure configuration quality [35]. The evaluators looked
for general configuration errors based on industry standards to assess the quality of the rule set.

One error was allowing connections to “Any” service to enter the network. This was considered a mistake because there are many high-risk services that should not be allowed and allowing all connections leaves the network open to known vulnerabilities. Almost 80% of the rule sets evaluated in the study had this error. A template-based authoring system could prevent a rule like this from being authored. A general template for firewall policy rules is:

\{Action\} \{Source IP addresses\} to \{Destination IP addresses\} \{Destination Port\}

In the template-based framework policy elements could be defined to prevent this error. The policy element \{Destination Port\} could be defined such that the value “Any” is not an allowable choice in the list of values. This would strongly encourage policy authoring to write more restrictive policies for incoming services. Another option is to communicate the risk of this policy to the policy author if they select “Any” for the policy element \{Destination Port\}.

In a study of SSL warnings, Sunshine et al. found that warnings that described the potential risks involved with continuing to visit the website in question increased awareness of risk among study participants [34]. But policy authoring research has not considered how to incorporate this type of information with the user experience. A reference is made to a loosely similar idea in a case study of designing a privacy preference interface [37]. Since specifying privacy preferences is a daunting task for the end-user, the work suggested organizing the display of the preferences with the more critical items near the top. This assists the class of user who will only spend a limited amount of time modifying their preferences – if they only make a few decisions, at least they’ve made the important ones.

The policy authoring interface could also provide a space to enter comments about the policy. This space could hold information about why the policy was written, what threat it is intended to protect against, or conditions under which the policy can be deleted. These features may help the policy author manage their policies more efficiently.

D. Provide Access to Metadata

It should be straightforward for the policy author to find the relevant information so they can understand what terms mean and understand the relationships between policy elements and values.

Policy domains tend to use jargon in their policies or use concepts that are unfamiliar to the policy author. For this reason, all available metadata for policies and policy elements should be easy to access. In the template-based framework, the policy elements can be associated with a large amount of metadata to provide as much context information to the policy author as possible and to support the implementation of the features mentioned above. Ontologies have been proposed in the semantic web domain as a way to structure and manage policy metadata [38]. In the semantic web domain, web services are being researched with the end-goal that they will eventually negotiate privacy and security decisions on behalf of the user. Representing the nuances of the user’s intended policy is especially important when the system is negotiating policy decisions for the user.

Prior work highlights the necessity of providing clarification for terminology not understood by the user. It’s been noted that there is a lack of standardized language even for the most commonly used security concepts [39]. Cranor et al. note the difficulty of finding the right terminology for privacy policies [17]. A user study of P3P policy authoring also supports this guideline, participants were confused by the terminology used in the user study tasks, and the user interface did not have room to provide sufficient metadata [24].

Our template authoring prototype was designed such that additional information about a policy template or policy element is accessible by clicking on the element in the user interface. The participants from the template authoring user study found the ability to view the details of policy elements made it easier to understand what the policy elements were and how they could be used to form templates.

V. CONCLUSIONS AND FUTURE WORK

In this paper we discuss remaining research challenges for security and privacy policy authoring. We focus our work on the importance of guidelines for usable policy authoring. We propose new guidelines to add to existing guidelines [7], and discuss their benefits. We demonstrate the guidelines a template-based policy authoring framework intrinsically meets and discuss features that can be added to meet the remaining guidelines. Also, where applicable, we present empirical evidence from a user study on template authoring that indicates the user demand, and utility, of these features [23]. Throughout the paper we outline the direction of the template-based policy authoring framework and discuss the progress to date.

The additional guidelines we suggest for security and privacy policy authoring are:

- Support appropriate limitation of expressivity
- Communicate risk and threats
- Provide access to metadata

The next step for this research is to design and evaluate a policy element authoring prototype that meets the new guidelines. It is also necessary to do the same with a policy authoring prototype. Empirical evaluation will show whether the guidelines are sufficient for achieving a policy authoring tool that is more usable than existing tools.

It is important to continue researching better tools and mechanisms for security and privacy policy-authoring, and to establish guidelines for better interfaces as we learn more. To achieve security goals, it is crucial that policy authors are able to author high-quality policies and to ensure the specified policy matches their intended policy.

ACKNOWLEDGMENT

This research was sponsored by the U.S. Army Research Laboratory (ARL) and the U.K. Ministry of Defence and was
accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. ARL, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES


