Access Control Policy Analysis and Visualization Tools for Security Professionals

Kami Vaniea
Carnegie Mellon University, USA
kami@cs.cmu.edu

Qun Ni
Purdue University, USA
ni@cs.purdue.edu

Lorrie Cranor
Carnegie Mellon University, USA
lorrie@cs.cmu.edu

Elisa Bertino
Purdue University, USA
bertino@cs.purdue.edu

ABSTRACT

Managing large sets of access-control rules is a complex task for security administrators. Each addition, deletion or modification of a rule causes many potential and unknown side effects ranging from rule conflicts to security breaches. Security researchers have attempted to alleviate this problem by proposing algorithms and tools which analyze lists of rules and provide administrators with the information that they need to better manage their rules. Unfortunately few of these analysis tools connect a policy problem to the source of the problem clearly. In this work we discuss an interface that visualizes the output of policy analysis and the source of the output in terms of rule lists and shows administrators the effect of their changes.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Access Controls*; H.1.2 [**Models and Principles**]: User / Machine systems—*Human Factors*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces

General Terms

Human Factors, Management, Security, Standardization

Keywords

Access control, Usability, Visualization, Policy Analysis

1. INTRODUCTION

The task of access-control policy management has become an increasingly hard problem for both security professionals and organizations in general. As issues such as security breaches become more public, organizations are becoming increasingly interested in effectively controlling access to their resources. In Deloitte and Touche's 2007 survey of 169 financial institutions they found that 98% of respondents were spending more money on information security than in the previous year [12].

Even with the increase in attention to information security, administrators still deal with many challenges including managing extremely large policy files for a range of different systems. Firewall rule sets, for example, can range in size from only a few rules to thousands [21]. Managing policies that are so large can require a higher cognitive load than the administrator can reasonably manage [5]. As a result administrators find ways of managing the load such as keeping technical notes about their own systems [7].

In this paper we define an access-control policy to be a collection of rules. The term rule is used as a general term to describe a tuple containing entities which can be evaluated to produce either an allow or deny decision. A firewall rule for example might contain the tuple (protocol, sourceIP, source port, destinationIP, destination port, action) while a Privacy-aware Role Based Access Control (P-RBAC) rule is actually a permission assignment of a form (role, resource, action, condition).

In addition to being large, policies can also be complicated. Understanding complex policies requires an in-depth knowledge of the policy language, policy semantics and the system on which it will be run. Even an administrator who is familiar with his system and confident in his ability to accurately make changes still makes errors [9, 16, 21]. Part of the problem is that administrators have to incorporate complex constraints communicated to them by users and higher level managers into the policy. As a result an organization's access-control policy may contain many exception cases which are often added by a variety of administrators [5, 10]. In a recent survey of over 11,000 security professionals the Ponemon Institute found that access rights to IT resources are often given out in an ad-hoc manner for specific projects (29% of respondents) or granted based on departmental demands (25% of respondents) [17].

The result is situations where policies actively running in an organization have errors which cause them to act in an unintended manner. In his work on firewall policies Wool showed that of the 37 policies he analyzed every single one was mis-configured [21]. Oppenheimer et al. conducted an analysis of web service errors and found that the largest cause of failure was administrator errors and policy errors

SOUPS Workshop on Usable IT Security Management (USM) 2008, July 23, 2008, Pittsburgh, PA USA.

were the largest category of administrator errors [16].

Security researchers would like to assist administrators in their analysis of access-control policy. To this end many tools have been created that analyze the relationships between rules and provide administrators with helpful information such as conflicts, misconfigurations and rule domination [3, 8, 20]. Unfortunately few if any of these tools are commonly used by administrators [11] for some of the reasons that are discussed in Section 2.

In this paper we will discuss some of the motivating ethnographic research that has focused on security administrators. We will look at how administrators use and interact with their tools to address security, specifically policy concerns (Section 2). Then we will consider the issues involved with policy analysis and more rigorously define our problem space (Section 3). Using the usability and policy analysis constraints we will put forward a proposed interface that would allow administrators to examine the output from policy analysis tools in a way that allows them to better understand the output and take the necessary action (Section 4). Finally we will discuss related work (Section 5) and summarize (Section 6).

2. ADMINISTRATION

Several different ethnographic and contextual studies have explored the tasks, opinions and cultural structure of security professionals. Such studies have identified several important trends in how security professionals engage in security tasks.

2.1 Tool Usage

Administrators use many different tools in their daily activities. These tools vary by tasks and administrators often switch between them as their task changes [5, 7]. We know of no research specifically addressing the types of tools commonly used to modify and maintain policies but there are several studies examining the types of tools used in general security administration.

Botta et. al. found that security professionals appreciate tools that exhibit easy to understand correlations between the data and the tool's output. Administrators in their study tended to avoid tools where it was difficult to understand how the output related to the data [7]. In their work Barrett et al. described an instance where an administrator mis-interpreted how a tool worked and spent several hours attempting to remove a non-existant bug [5].

As a result many administrators write their own scripts or use basic command line tools such as *ps* (lists all running processes). Botta et al. observed one administrator who had accumulated 2,000 scripts over 25 years [7]. These tools are less powerful than specially designed tools with beautiful GUIs but the administrator can be sure that the output he is getting is correct. He can also feel assured that his mental model of how the tool is working matches what is actually happening.

2.2 Cognitive Load

Cognitive overload is another major issue for administrators. The policies they deal with can be extremely large and virtually impossible to maintain in working memory. As a result administrators have to use methods and tools to reduce policies into manageable pieces that can be worked with.

Altmann discusses the impact of the limitations of working memory in his work on expert computer programmers. He found that while it is virtually impossible for an expert computer programmer working on a large program to keep all relevant details in working memory the programmer is able to store the location of potentially needed details in near-term memory. When the programmer has the need for such a detail he is able to retrieve the location (approximate line number) of the needed detail which can then be referenced [4]. We anticipate that a security professional who is manipulating a large policy file may follow a similar mental process. As he works with the file he builds an understanding of the location of relevant information. When a specific detail is needed he is able to retrieve its location (provided it has been recently seen) and can scroll back to view it in detail.

Observation of security administrators reveals that they actively attempt to deal with extra cognitive load by progressively eliminating irrelevant data. In their work on log visualization Abdullah et al. observed that administrators would use tools in order to remove all the irrelevant data first then try and sort through what remained [1]. Botta et al. also observed administrators using tools in order to do filtering of data. An administrator would take the output from one tool and feed it into another tool which could then be used to find the important information.

2.3 Motivating Example

Bob is an administrator at the corporate offices for an international banking company. The corporate offices contains many office buildings which house several thousand employees. Access to offices, meeting rooms and labs within the buildings is controlled by a swipe card system maintained by the IT department of which Bob is a member.

Whenever employees need access to a new location they submit a request to an online form which sends a notification to Bob or one of his co-admins who then make the necessary changes to the central policy. Since several different people are all making changes to the building's access-control policy there are cases where one administrator adds a rule not knowing that another administrator has previously created a rule concerning the same user, action, resource tuple. When this happens the system can enter an unintended state. In the best case some rules may never be used, in the worst case the system may enter a conflicted state and all members of a group may be unable to access their resources.

To help address policy issues Bob would like to use a policy analysis program which could help him identify and fix potential issues before they become problems.

3. POLICY ANALYSIS

We ground work reported here on Privacy-aware Role Based Access Control (P-RBAC) because of following reasons:

- P-RBAC is a natural extension of well-known RBAC models [15, 13] to support fine-grained access control policies. Customized conditional languages [15, 14] in P-RBAC can express commonly used conditions in policies. Policy authors can directly specify flexible relation, AND/OR, between permission assignments [14]. These functionalities of P-RBAC make it an appropriate tool for complicated policy deployment en-

vironments.

- Policy ratification [2] is directly supported by P-RBAC [13, 14, 15]. Policy ratification provides a formal process to policy managers/authors to certify the appropriateness of a policy before the policy is activated. Three primitive operations are identified for Policy ratification: conflict detection, dominance and coverage [2]. Policies are in conflict if they cannot be enforced simultaneously. A policy is dominated or redundant if the policy when added to a system does not modify the behavior of the system. Coverage refers to the determination of whether a policy set covers all the cases of interest, e.g., all users in the system have rights to at least one resource in the system.

3.1 A Brief Introduction of P-RBAC

P-RBAC [13, 14, 15] is a family of Privacy-aware Role Based Access Control (RBAC) models that extend RBAC with support for privacy policies. P-RBAC supports the notion of "privacy-aware" permissions, that is, authorizations taking into account privacy-relevant information, such as the purpose of data usage, as well as "regular permissions", that is, which role can perform which action on which resource under what conditions. P-RBAC, illustrated in Figure 1, includes several sets of entities: Users, Roles, Resources, Actions, Purposes, Obligations, and Conditions expressed by using customized conditional languages [15, 14].

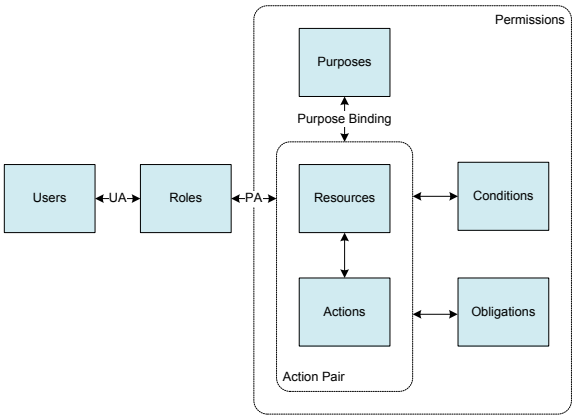

Figure 1: Components in P-RBAC

A user in P-RBAC is human being, and a role represents a job function or job title within the organization with some associated semantics regarding the authority and responsibility conferred on a member of the role. Resources refer to any objects or information that should be protected. An action is an executable image of a program, which upon invocation executes some function for the user. The types of action and resources that P-RBAC controls depend on the type of system in which they are implemented. Conditions further limits the scope of the permission to realize fine-grained access control. The motivations for introducing purposes and obligations in P-RBAC are discussed in [13, 15]. A permission consists of an action, a resource, a condition, a purpose, and an obligation set. Access control policies are specified by user assignments (assigning users to roles) and permission assignments (assigning permissions to roles). In this paper, we leave the analysis related to obligations [13] and purposes as future work and only focus on permission assignments due to space reasons. It should be noted that we may use the terms *permission assignments* and *rules* interchangeably depending on the context.

3.2 Relation of Permission Assignments

In our example Bob, the administrator, manages access control policies for a large system in conjunction with several other administrators and departmental policy authors. Now we investigate how these administrators author policies together with departmental authors in terms of P-RBAC assignments.

Bob and his co-workers are part of the central IT department for their company but there are many other IT departments spread all over the world which are responsible for maintaining the policies that relate to their own departments. Since Bob's department is the primary IT department headed by its Chief Security Officer, they write some special permission assignments which are enforced across all offices. Usually these special permission assignment are high level directives that are not sufficiently fine-grained and cannot be directly enforced. Meanwhile, Bob, his co-workers, and department authors also write some permission assignments for departments, in other words, these permission assignments are only enforced within some departments.

Two different permission assignments are identified in this scenario. One is "if" permission assignment, also known as *terminate* permission assignment, that will allow an access request by itself within a department. Another one is "only if" permission assignment, also know as *mandatory* permission assignment, that will be required by all access requests related to the permission assignment [6, 18]. P-RBAC directly supports both mandatory and terminate permission assignments by allowing policy administrators to specify relations like "AND" and "OR" among permission assignments.

$$p_{m1} \wedge p_{m2} \wedge ... \wedge p_{mn} \wedge ((p_{td11} * p_{td12}...) \vee (p_{td21} * p_{td22}...) \vee ...)$$

where $\wedge$ represents "AND", $\vee$ represents "OR", and $*$ represents either "AND" or "OR", p_{mi} represents mandatory permission assignments, and p_{tdij} represents the j-th terminal permission assignments in the i-th department. The expression specifies that in order to allow an access request all mandatory permission assignments must allow the request and at least one terminate permission assignment allow the request as well.

3.3 Conflicts and Dominance

Conditions in permission assignments and the complexity of the relations between permissions assignments can easily cause administrators to think they had just made a good change when in truth their new policy conflicts with some existing one or is simply subsumed by another policy.

Since all policies have to be enforced simultaneously, and at same time conditions are required for fine-grained access control, it is possible that conflicting permission assignments arise due to unsatisfiable conditions. For instance, given two permission assignments:

PA_1 = Employee, Access, Conference Room 1101, 9am < Time < 12pm
PA_2 = Employee, Access, Conference Room 1101, 1pm < Time < 5pm

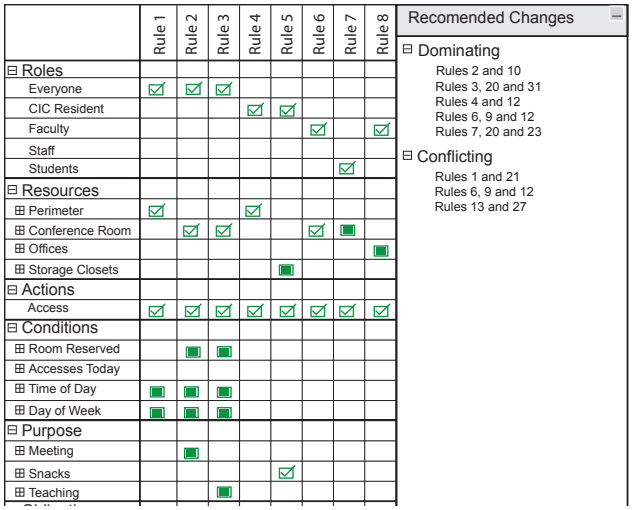

Figure 2: Base interface for Prisimos . On the left is the policy shown in a grid interface. On the right is the output of a policy analysis algorithm. Clicking on any of the recommendations causes the interface to shift to the one pictured in Figure 3.

If both PA_1 and PA_2 are mandatory permission assignments, then no one can really access the conference room because it is impossible to access the room at a time of day that is both between 9am-12pm and at the same time between 1pm-5pm. If one of the permission assignments is mandatory and the other one is a terminal permission assignment, then the terminal permission assignment is useless.

Similarly, it is possible that one permission assignment dominates another permission assignment. For instance, given two permission assignments:

PA_3 = Employee, Access, Conference Room 1101, 9am $<$ Time $<$ 5pm
PA_4 = Employee, Access, Conference Room 1101, 1pm $<$ Time $<$ 5pm

If PA_3 and PA_4 are mandatory permission assignments, PA_4 dominates PA_3 because the existence of PA_3 has no effect on the final decision based on these policies. By contrast, if PA_3 and PA_4 are terminal permission assignments with OR relation, PA_3 dominates PA_4 for the same reason.

Theoretically the detection of conflicts and dominance relation in policies is a satisfiability problem of conditions that can be NP-hard in its worst cases. However, it has been shown that for conditions most commonly used in policies, such as range on numbers, string comparison, and relation on a finite partial order set, tractable algorithms exist [2]. These algorithms are applied in this paper to detect possible conflicts and dominance relations in permission assignments.

4. Prisimos

The Prisimos system is a proposed (not yet implemented) design based on the authors' prior experience with system administration and policy analysis. Prisimos is intended to help administrators visualize and address issues with conflicts and dominance by communicating the output of policy analysis algorithms in a meaningful and actionable way.

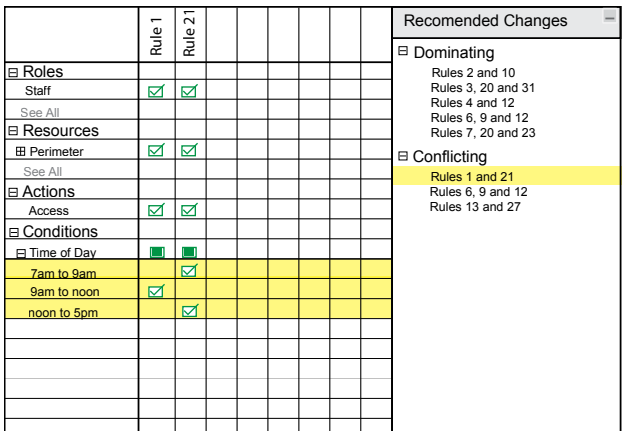

Figure 3: The Prisimos interface after the user has selected to view conflicting rules 1 and 21. The interface limits the display to only include relevant policy elements.

Prisimos is designed to present both the output of a policy analysis algorithm and the policy itself to the user in a way which visualizes the results of the analysis in terms of the individual rules in the policy.

The primary interface for Prisimos is shown in Figures 2 and 3. The interface is divided into two primary components. On the left is a grid interface which displays the policy as a list of rules. On the right is a sidebar which shows the output of a policy analysis algorithm. Used together the two components allow the user to step through the algorithm output and for each potential issue examine the relevant parts of the policy itself before deciding upon the appropriate action.

The grid interface on the left side of the interface displays the policy itself, the design was inspired by the Expandable Grid presented by Reeder et al. [19]. Across the top of the grid is a list of all the currently displayed rules which in this case represent permission assignments. On the left are all the entities of the rules. Each rule in the policy is broken into its component entities, in this case roles, resources, actions and conditions. These component entities are displayed on the left as top level nodes for their sections. Below each entity name are all the potential values that entity could have. For example, in Figure 3 under the conditions entity we see the group *Time of day* and below that the list of different time ranges. In this figure the number of values shown for each entity of potential day times has been reduced to only show those values relevant to conflicting rules 1 and 21. Clicking the "See All" link would open up the entity so all values would be shown.

In the interior of the grid are a series of green boxes. The presence of a green check box means that the specified rule uses that value. In Figure 3 we can see that Rule 1 concerns *Staff*, *Perimeter*, *access* and *9am to noon*. In natural language this might read *Staff are permitted to access any perimeter door between 9am and noon*. A green box next to a group is used to indicate that a member of that group is mentioned in the current rule. This allows an administrator to easily scan for relevant groups without having to search through them. In Figure 2 we can use the green boxes to see that *Time of Day* is used in Rules 1 but *Room Reserved*

is not used as a condition.

The sidebar on the right displays the output of the algorithm analysis. As was mentioned in Section 3 the Prisimos visualization is designed to accommodate a wide range of different types of algorithms. In our example the algorithm looks for Conflicting and Dominating rules which are displayed in their own sections in the Recommended Changes bar. A different type of analysis algorithm might look for potential administrator error or more domain specific concerns. For each type of issue we list the sets of rules which are having that issue. In Figure 2 we can see that rules 1 and 21 are in conflict. We can also see that rules 6, 9 and 12 are in conflict.

The user can click on any of the identified conflicting sets of rules (Figure 2) which will cause the grid interface to the left to only display those rules (Figure 3). Additionally the interface will automatically hide all values which are not used by the currently displayed rules and highlight the values that are in conflict. This reduces the cognitive load the administrator has to deal with by eliminating unnecessary information and highlighting important information.

We believe that a visualization like Prisimos will assist administrators in viewing the output from policy analysis tools for several reasons. First the tool assists the user by minimizing the parts of the policy that need to be viewed at one time. This reduces the amount of working memory the administrator needs to view the policy. It also allows the user to ignore all unimportant parts of the policy and look only at the relevant rules in an uncluttered environment instead of scrolling around trying to find and mentally parse the rules.

Secondly, it is easy to see the relationship between what is shown in the policy section of Prisimos and the actual policy. The tool engages in minimal alterations of the rules themselves and instead attempts to display the rules in a format that is as close to the actual rules as possible. This format makes it easier for an administrator to form an accurate mental model of the policy itself. Additionally if an administrator wants to double check a specific rule with the original file it should be easy for them to compare the visualization of the rule with the text version.

5. RELATED WORK

There have been several systems proposed which assist policy authors in managing their policies through automatic analysis of the policy rules.

Alshaer et al. presented a Policy Advisor for firewall policies which analyzes a set of firewall rules and provides the user with the output in the form of a series of recommendations. Their approach also shows the list of rules in an easily comparable table format [3]. However, unlike our proposed solution their interface makes no attempt to limit the number of rules a user is viewing. Additionally the Policy Advisor is designed to specifically work with firewall rules and is not well suited for access-control policies in general.

There are several interfaces that have been developed that assist the user in modifying and creating system actionable security policy. A *system actionable* policy is a policy which is created and edited by a human but enforced and acted upon by a computer system. One example is the SPARCLE Policy Workbench which allows policy authors to write policy in natural language and parses the policy so it can be enforced by a computer [8, 20]. The Expandable Grid is another example of a tool which allows users to manipulate the system actionable policy for File System rules [19]. Both of these tools are designed for creating and viewing policies but they address the issue of viewing the output of analysis tools in a simple and understandable way.

6. SUMMARY

We looked at how security administrators regard and use their tools to manage large systems. We also looked at how managing such systems is challenging for administrators due to the system's size and complexity. Analysis tools that analyze large rule sets for issues exist but are difficult to use and don't conform to how administrators typically use tools.

We then looked at policy analysis in terms of P-RBAC. We examined some of the potential issues an administrator might face in a system using P-RBAC and how the policy can have conflicts and domination issues.

Finally we proposed a system called Prisimos which provides a visualization for both the policy and the analysis. Using information hiding techniques Prisimos allows users to step through each issue identified in the analysis viewing only details relevant for that specific issue.

In future work we hope to build the Prisimos system and connect it with existing policy analysis tools. We would like to explore how the visualization would work in different policy domains. We would also like to test the effectiveness of the interface in laboratory studies with real users followed up by studies on administrators in real settings.

7. ACKNOWLEDGEMENTS

The work reported here has been supported by the IBM OCR project "Privacy and Security Policy Management."

8. REFERENCES

[1] K. Abdullah, C. Lee, G. Conti, J. A. Copeland, and J. Stasko. IDS rainstorm: Visualizing IDS alarms. In *Visualization for Computer Security, 2005. (VizSEC 05). IEEE Workshop on*, 2005.

[2] D. Agrawal, J. Giles, K.-W. Lee, and J. Lobo. Policy ratification. In *POLICY '05: Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05)*, pages 223–232, Stockholm Sweden, 2005. IEEE Computer Society.

[3] E. S. Al-Shaer and H. H. Hamed. Firewall policy advisor for anomaly detection and rule editing. In *Proceedings of IEEE/IFIP IM*, 2003.

[4] E. M. Altmann. Near-term memory in programming: a simulation-based analysis. *International Journal of Human-Computer Studies*, 54:189–210, 2001.

[5] R. Barrett, E. Kandogan, P. P. Maglio, E. Haber, L. A. Takayama, and M. Prabaker. Field studies of computer system administrators analysis of system management tools and practices. In *CSCW*, 2004.

[6] A. Barth, A. Datta, J. C. Mitchell, and H. Nissenbaum. Privacy and contextual integrity: Framework and applications. In *SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P'06)*, pages 184–198, Washington, DC, USA, 2006. IEEE Computer Society.

[7] D. Botta, R. Werlinger, A. Gagné, K. Beznosov, L. Iverson, S. Fels, and B. Fisher. Towards understanding it security professionals and their tools. In *SOUPS '07: Proceedings of the 3rd symposium on Usable privacy and security*, pages 100–111, New York, NY, USA, 2007. ACM.

[8] C. A. Brodie, C.-M. Karat, and J. Karat. An empirical study of natural language parsing of privacy policy rules using the sparcle policy workbench. In *SOUPS '06: Proceedings of the second symposium on Usable privacy and security*, pages 8–19, New York, NY, USA, 2006. ACM Press.

[9] X. Cao and L. Iverson. Intentional access management: Making access control usable for end-users. In *Sumposium On Usable Privacy and Security (SOUPS)*, 2006.

[10] S. Kaemer and P. Carayon. Human errors and violations in computer and information security: The viewpoint of network administrators and security specialists. In *Applied Ergonomics*, volume 38, pages 143–154, 2007.

[11] E. Kandogan and E. M. Harber. *Security Administration Tools and Practices*, chapter 18, pages 357–375. O'Reilly, Sebastopol, CA, 2005.

[12] R. Mears and L. Ponemon. 2007 privacy & data protection survey, Febuary 2008.

[13] Q. Ni, E. Bertino, and J. Lobo. An obligation model bridging access control policies and privacy policies. In *SACMAT '08: Proceedings of the 13th ACM symposium on Access control models and technologies*, New York, NY, USA, 2008. ACM Press.

[14] Q. Ni, D. Lin, E. Bertino, and J. Lobo. Conditional privacy-aware role based access control. In J. Biskup and J. Lopez, editors, *ESORICS*, volume 4734 of *Lecture Notes in Computer Science*, pages 72–89. Springer, 2007.

[15] Q. Ni, A. Trombetta, E. Bertino, and J. Lobo. Privacy-aware role based access control. In V. Lotz and B. M. Thuraisingham, editors, *SACMAT*, pages 41–50. ACM, 2007.

[16] D. Oppenheimer, A. Ganapathi, and D. A. Patterson. Why do internet services fail, and what can be done about it? In *Proceedings of the USITS 03 4th USENIX Symposium on Internet Technologies and Systems*, 2003.

[17] L. Ponemon. 2008 national survey on access governance, us study of it practitioners. Private and Confidential Document, Febuary 2008.

[18] D. Raub and R. Steinwandt. An algebra for enterprise privacy policies closed under composition and conjunction. In *ETRICS*, pages 130–144, 2006.

[19] 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 *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1473–1482, New York, NY, USA, 2008. ACM.

[20] K. Vaniea, C.-M. Karat, J. B. Gross, J. Karat, and C. Brodie. Evaluating assistance of natural language policy authoring. In *SOUPS*, 2008.

[21] A. Wool. A quantitative study of firewall configuration errors. *IEEE Computer*, 37(6):62–67, 2004.