Detecting and Countering Insider Threats: Can Policy-Based Access Control Help?

Jason Crampton[1]

Information Security Group
Royal Holloway, University of London
Egham, United Kingdom

and

Michael Huth[2]

Department of Computing
Imperial College London
London, United Kingdom

Abstract

As insider threats pose very significant security risks to IT systems, we ask what policy-based approaches to access control can do for the detection, mitigation or countering of insider threats and insider attacks. Answering this question is difficult since little public data about insider-threat cases is available, since there is not much consensus about what the insider problem actually is, and since research in access control has by-and-large not dealt with this issue in the past. We explore existing notions of insiderness in order to identify the relevant research issues. We then formulate a set of requirements for next-generation access-control systems whose realization might form part of an overall strategy at addressing the insider problem.

Keywords: Trustworthiness. Trust. Insider Problem. Insiderness. Access Control Policies.

1 Introduction

Most people have an intuitive, albeit informal, understanding of the terms "insiders" and "insider attacks". We all have seen spy movies in which agents and double agents exploit inside knowledge or privileged access to inflict damage on a hostile regime. The consequences of insider attacks can be extremely damaging. In January 2008, for example, Jerome Kerviel circumvented internal security mechanisms to place more than $70 billion in secret, unauthorized derivatives trades which, according to his employer Société Générale, resulted in a net loss of $7.2 billion to the bank [15].

[1] Email: jason.crampton@rhul.ac.uk
[2] Email: M.Huth@imperial.ac.uk

This paper is under consideration to be electronically published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

It is less clear, though, how to formally define what insiders and insider attacks are, or what effective measures one could or should take to discover, prevent, mitigate or counter threats from insiders. For a small family-run business, for example, insider threats may pose a negligible risk. For a national intelligence agency, on the other hand, insider threats may be by far the biggest source of risk and potential damage.

For these reasons, which we will discuss in more detail in Section 2, we argue that one should not seek to provide a single universally applicable definition of "insider" or "insiderness". In this paper, we mean to explore the insider problem from the perspective of *access control within IT systems*. We realize that this confines the insider problem to a single aspect of an organization's security mechanisms. But we feel it is useful to impose such focus in our investigations for the following reasons:

- most of today's IT systems implement some form of access control – for example, to identify authorized users of the system, to limit the actions that each authorized user may perform – and understanding an organization's access control should help in discovering insiders, their degree of insiderness, and the level of risk they would represent
- an understanding of insiderness and insider threats in terms of access-control privileges and access history may allow IT systems to adapt existing access-control mechanisms in order to mitigate, counter or even prevent insider attacks
- the recent advances in policy-based access control may be leveraged to develop access-control frameworks in which policies can evolve dynamically or be retrofitted to deal with insider threats
- such policy-based frameworks may then be able to incorporate quantitative methods of risk evaluation, such as anomaly detection, in order to formulate and enforce effective insider control.

It is also our hope that the understanding of the insider problem gained from studying the well-defined and structured context of policy-based access control may be transferrable to policies that are merely descriptive and not enforced within IT systems. Jerome Kerviel, for example, managed to launch his attack because, over time, he was able to take on two roles that should not have been held by any single individual (even at different points in time). But this role-based separation of duty [21] was not implemented in any part of the IT system. Support for role-based access control (RBAC) [21] within that IT system, in combination with proper identity management, may have prevented or at least detected this insider attack.

We recognize that there are various factors that culminate in insider attacks, as described in [14], some of which cannot be addressed or phrased in terms of access control. But some certainly can, as highlighted in Observations 5-7 in [14].

Outline of paper.

In Section 2 we motivate this work and discuss relevant related work. In Section 3 we introduce an alternative trust-based perspective on the insider problem. In Section 4 we identify requirements that arise from this new definition and sketch how we can realize these requirements by leveraging recent work in access-control

policy languages. We conclude by discussing additional related work and our plans for future research in Section 5.

2 Motivation and related work

We first introduce three examples that illustrate the diversity of the insider problem. We then discuss some existing work on defining insiders and the insider problem, concluding the section with a discussion of previous work on access control and the insider problem.

2.1 Illustrative scenarios

Inevitably, any organization of even moderate size and complexity must trust (some of) its employees with sensitive information and resources. For this reason, most organizations have security policies that specify who is authorized to access what resources. To ensure that these policies are enforced, all modern operating systems and many applications provide authentication and authorization services. In using these services, an organization allows a number of individuals access to privileged knowledge of mission-critical information, and special access rights to mission-critical resources or both. To us, the *insider problem* refers to the inherent threats that organizations face when granting individuals such special privileges. Here are some examples of what many people would consider to be insider problems, the first one corresponding loosely to a case reported in [14]:

- a system administrator may be given the right to manipulate folders that contain a substantial portion of her organization's intellectual property;
- urgent building work at a company's headquarters may require non-employees to be given permission to enter security-sensitive parts of that building; or
- the personal assistant of a chief financial officer (CFO) would have access to the CFO's diary and contents or patterns of communication.

Part of the problem is that these individuals enjoy some particular trust relationship with the organization:

- the system administrator is trusted not to just encrypt all files with a key only she knows, e.g. in order to blackmail her organization
- the building workers are trusted not to be IT security specialists from competitors who want to infiltrate the company's premises and intranet
- the personal assistant of a senior officer is expected to keep the existence and status of merger talks confidential.

Organizations have to trust individuals, otherwise we cannot distinguish between authorized and unauthorized users. However, it may not be possible to articulate a trust relationship with any precision because of various reasons: lack of software support (e.g. for the enforcement of role-based seperation of duty in the attack by Jerome Kerviel), excessive cost, staff constraints, etc. In the context of the above scenarios:

- A small start-up company may simply not be able to afford several system ad-

ministrators, thus leaving the entire management of stored intellectual property (including backup procedures) to that sole administrator.

- It may be impractical to guard the movements of each individual builder in a confined work space, creating an opportunity to access the intranet through a wall socket.
- A CFO naturally insists on a convenient single point of contact for arranging meetings, taking minutes, etc. But such convenience increases the risk and threat level of an insider attack by the personal assistant.

The above examples already indicate the difficulty of obtaining a working definition of "insider" that will fit all instances of the "insider problem". This has already been emphasized by Bishop in [3]. To illustrate this difficulty, it is too simplistic to say that every and only employees of an organization are insiders for that organization, considering that the above builders might be employees of an "outside" contractor but do have physical access to the "inside", the company's headquarters.

Even with a working definition of "insider" in hand, it is not clear how to put that definition into use so that it may aid with the detection, mitigation or countering of insider threats. Probst and Hunker [20] point out that many organizations take basic steps toward preventing insider attacks but rarely engage in addressing serious insider attacks, and the authors argue that this appears to be rational economic behavior. Under that assumption, there is therefore economic pressure that research deliver solutions that can be deployed and managed in a manner that is perceived to be economically viable.

2.2 Definitions of insiders

Matt Bishop organized a panel at NSPW'05 on the insider problem and noted the diversity of existing definitions of insiders [6], which we cite here: Brackney and Anderson define an insider to be *"someone with access, privilege, or knowledge of information systems and services"* and a definition of the insider problem as *"malevolent (or possibly inadvertent) actions by an already trusted person with access to sensitive information and information systems"* [4]; in contrast, Patzakis defines an insider (implicitly) as *"anyone operating inside the security perimeter"* [16].

The summary of the recent Dagstuhl Seminar on countering insider threats [2] proposes several definitions of an insider, e.g.,

- as someone "defined with respect to a resource, leading to *degrees of insiderness*[3]"
- as "somebody with legitimate", past or present, "access to resources"
- as a "wholly or partially trusted subject"
- as "a system user who can misuse privileges"
- as someone with "authorized access who might attempt unauthorized removal or sabotage of critical assets or who could aid outsiders in doing so".

It should be clear that the protagonists in the scenarios described earlier fit some

[3] Our italics.

of these definitions better than others. Any attempt at a comprehensive definition of insiders will have shortcomings. For example, if an "outsider" manages to break into an IT system by masquerading as a legitimate inside user, does this constitute an insider attack? In Section 3 we offer our own definition, which is based on and recalls definitions of trust and trustworthiness from the 1970ies.

2.3 The insider problem and access control

For sake of completeness, we now provide a high-level description of what we understand by access control. In an attempt to implement enterprise security policies, every attempted interaction between an authorized user and a protected resource (which could be as specific as a particular entry in a database table or as general as a computer system) is "trapped" by the access-control system. This interaction is modeled as an access request. The access-control system determines whether the access request is authorized (according to some policy and relevant security configuration data) and then either allows the interaction to proceed (if the request is authorized) or prevents the interaction otherwise. Hence, an access-control system typically comprises: a *policy enforcement point* (PEP), which traps attempted interactions, generates access requests and enforces authorization decisions; a *policy decision point* (PDP), which accepts access requests and returns authorization decisions; and a *policy repository*, which stores authorization policies. In order to specify an access-control model, therefore, we need to define a language for articulating authorization policies and an algorithm that is used to evaluate access requests with respect to such policies.

It can be seen that many problems that arise when discussing the insider problem are related to abuse of privileges granted by authentication or authorization services, both of which may be regarded as providing some form of access control. So how can an understanding of access control enable us to better understand and address the insider problem? There is very little existing work in this area, although Park and Giordano suggest the following issues need to be addressed in order to counter the insider threat in large-scale, dynamic IT systems [18]:

- The management of privileges should not be based on identities, as this won't scale to distributed, highly dynamic and heterogeneous systems. The authors propose RBAC as a potential solution.
- Access control needs to also support finer levels of granularity. For example, most systems grant access to an entire structured file and not, say, to specific fields in that file.
- Most approaches to access control decide an access request by a static and predefined set of rules. There is a need for "*active* access control", which makes decisions dependent on context.

We believe that these requirements are rather generic and do little to address the insider problem. Indeed, it is hard to believe that RBAC, which is by design unconcerned with individual users, is an appropriate underlying model for an access-control system that provides some protection against insider threats. Moreover, there is little evidence to suggest either that existing access-control systems

are insufficiently fine-grained (access-control systems for relational databases are generally rather fine-grained, for example) or that having more fine-grained access-control systems would help counter the insider threat in general. We would agree that access-control systems will need to be more flexible and responsive to environmental conditions, not just pre-defined policy rules, although it has to be noted that any adaptive access-control system will still need to be governed by some pre-defined (policy) rules.

In the next two sections, we formulate an alternative definition for the insider problem that is more directly related to access control. We then discuss how recent advances in trust and reputation systems, and access-control policy languages can be used to address this revised insider problem.

3 Trust, trustworthiness, and the insider problem

In the context of access control, there is an assumption that authorized users will behave in a certain way: that is, authorized users are assumed – that is to say, *trusted* – not to abuse the privileges for which they are authorized. The simplest example is that it is assumed (must be assumed, in fact) that a user will not divulge his username and password to another user. Clearly, however, some authorized users are not *trustworthy*. The insider problem, then, arises because the set of trusted users (with respect to some particular context) is not equal to the set of trustworthy users (for the same context).

Our scenarios and the proposed definitions above share a concern for resources and the interactions between users and those resources, where these interactions could modify, create or destroy resources. Given that insiders are somehow trusted with managing and transforming resources responsibly, it may be fruitful to focus on access requests and the potential threats that granting such requests may pose to the tactical or strategic aims implicit in the underlying IT system of an organization. For example, instead of asking whether an outsider is masquerading as an insider or not, we may want to focus on the risk and probability that a specific inside user account has been compromised, and let our access-control systems adapt appropriately should such a compromise be detected.

This consideration of resources and of access rights to resources has led to the notion of degrees of insiderness [1], where the more privileges and skills a person has, the more risk he or she represents. However, we would argue that it is the abuse of an authorization that poses a risk and that this risk is some function of the authorized action, the trustworthiness of the user and relevant contextual information (which might include, for example, previously authorized access requests). There is no need for different levels of insiderness, just different levels of trustworthiness. The degree of insiderness of a user has also been regarded as the threat level posed by that user [1]. Again, we do not regard this notion as particularly helpful. It seems intuitively reasonable to regard a user that poses a significant threat as simply being less trustworthy.

We now briefly discuss connections between our formulation of the insider problem, trust-management systems, and risk-assessment frameworks. In particular, there is a natural correspondence between trustworthiness and the probability of

misuse of authorizations. This suggests that trustworthiness and existing risk-assessment methodologies could be combined in a risk-aware access-control system. We elaborate this point in the next section. It is interesting to note that trust-management and reputation-management systems could be regarded as tools to compute trustworthiness (not *trust*).

This connection is illustrated by similarities with problems and known attacks in reputation systems [9]: A rogue insider may over time build up trust through loyal and compliant behavior, leading to more seniority and more privileged access rights, only to then inflict fatal damage to his organization on behalf of a competitor.

We note that there has been some preliminary work on combining risk and access control [8] and on using trust to assign users to roles [7,13]. However, these efforts have a rather narrow focus, respectively on Multi-Level Security and Role-Based Access Control.

4 Access control and trustworthiness

As we have said, we believe that the central problem is that trusted users may not be trustworthy. (That is, we need to be able to deal with a user for whom an erroneous, unjustified or obsolete assumption has been made about his reliability.) The question, then, is how to define access-control policies in such a way that the privileges for which users are authorized can be modified automatically (i.e, without human intervention) when those users are revealed, or suspected, to be untrustworthy. Of course, a preliminary problem is how to reliably identify and report to the access-control system those users whose trustworthiness is questionable. We briefly mention this topic in Section 3.

We therefore list additional requirements for addressing the insider problem within policy-based, access-control systems. To understand these requirements, it is useful to think of access requests as tuples of the form (s, o, a, c) where s ranges over subjects (i.e. user proxies), o ranges over objects (i.e. protected resources), a ranges over actions (such as *read*, *write*, etc.), and c ranges over contextual information (such as whether the user is accessing the system remotely). Sets of such tuples are thus subsets X of a space $S \times O \times A \times C$ of access requests. Such subsets can therefore be interpreted as predicates, which we refer to as *request predicates*. Such predicates may depend only on context, so they could be seen as describing subsets of C. We then call such predicates *context predicates*.

Informally, given X there is a natural policy p_X^+ that authorizes all requests in X and denies all others. [4] Dually, a policy p_X^- might deny all requests in X, and authorize all others.

Our requirements are as follows:

- The ability to declare context predicates and request predicates, where the latter express sets of access requests of interest and the former capture contextual state; and the ability to transform such declarations into access-control policies, for example by turning predicates X into policies p_X^+ or p_X^-.

[4] Strictly speaking, it is the policy-decision point that decides whether or not a request is authorized by a policy or not. However, we don't make this distinction explicit henceforth.

- The ability to transform an access-control policy p into a set of access requests X that captures some aspect of the behavior of policy p. For example, we might want to identify all requests authorized by policy p. However, this is by no means the only possibility. We may wish, e.g., to compute those requests for which the policy p is overdefined (conflicts) or underdefined (gaps).
- Support for typed, modular programming of access-control policies, with a rich set of declarative coordination patterns. This would facilitate the creation of robust, composed authorization policies and would exploit the features supported by the previous two requirements.

Given these requirements, we propose that there is value in transferring the principles of declarative programming into the domain of active access control. Declarative programming (by which we loosely mean extensions of functional programming and logic programming; or modeling languages based on logic such as Alloy [11]) focuses on the "what" and not on the "how" of computation. So declarative access-control policies are concerned with what requests will be granted under which contextual circumstances, but they don't have to specify how such policies are then enforced. [5]

This separation of concerns between policy specification and implementation brings additional benefits. One can design declarative languages that have typed modules and so support robust policy composition that facilitates reuse and offers a policy authoring tool that is hopefully descriptive and intuitive enough for policy writers. [6] Indeed, we believe the use of modular, declarative programming techniques will be able to support active access control that has a flexible range of granularity and is not necessarily identity-based.

We now discuss each of these requirements in more detail and illustrate what their realizations would and will provide. The first requirement suggests the use of abstract request predicates and propositional logic connectives so that one can form expressions such as

$$\texttt{Manager} \wedge \texttt{OnDuty} \wedge \neg\texttt{Weekend} \tag{1}$$

The access request is left implicit. Some of these abstract request predicates may even not depend on the access request. For example, if a subject is requesting to edit a certain PDF document, the evaluation of the above expression will not depend on the type of action or document, but will only depend on whether said subject presently has role `Manager`, is currently on duty, and whether the request occurs on a week-day. So `Weekend` would better be seen as a context predicate.

The expression in (1) is declarative since it does not elaborate on how the connections between, say, `Manager` and subjects are being implemented. In particular, it does not necessarily commit to an underlying RBAC model nor does it explicitly depend on the handling of authenticated attributes, etc. The provision of abstract

[5] We assume that declarations will have intuitive operational or denotational semantics and will therefore be implementable.

[6] It is of interest to note that the financial trading community has now recognized the need to be able to program as close to the user domain (e.g. financial traders) as possible. Declarative programming, mostly in the form of functional programming, is becoming an important tool in that sector, even for back-office aspects such as specifying and analyzing contracts [12].

interfaces for specifying access-control policies has previously been suggested and developed in the work of Bruns and Huth [6,5].

One can now lift expressions such as the one in (1) to genuine policies, as in

$$\texttt{Grant if Manager} \wedge \texttt{OnDuty} \wedge \neg\texttt{Weekend} \tag{2}$$

which is the declarative way of encoding policies of form p_X^+ discussed above. We adopt here the interpretation of [6,5], where the policy in (2) grants all requests that satisfy the composed predicate in (1), and where said policy has a gap (meaning it is undefined) at all other requests.

The second requirement allows us to identify sets of access requests that stem from policies themselves. For sake of illustration, consider two primitives p@`Denies` and p@`Grants` that, given an access-control policy p, declare request predicates that capture that set of access requests that policy p denies, respectively grants. It should be noted that policy p may well not be equal to policy `Grant if` p@`Grants`. For example, policy p may report a conflict for a particular request (due perhaps to the composition of inconsistent sub-policies), or it may report an error value indicating data incompatibilities, etc. But policy `Grant if` p@`Grants` can only grant or report a definitional gap.

Meeting the third requirement means that we can wrap policy composition patterns into parameterized modules, and then instantiate or reuse such modules in policy composition or orchestration. For example,

```
pol insiderThreat(abnormalBehavior: reqs, insider: reqs) {
  (grant if !abnormalBehavior & insider) >
  (deny  if abnormalBehavior | !insider)
}
```

declares such a module with two types, `pol` for policies and `reqs` for request predicates. This module takes as input two request predicates and outputs a policy, where `>` declares a priority composition. The output policy therefore grants a request if it comes from an insider and if no abnormal behavior has been flagged; and it denies a request if either abnormal behavior has been flagged or the request does not come from an insider. The module itself does not state how insiders and abnormal behavior are defined. But this separation of concern creates flexibility, since different circumstances may require different notions of insiders or abnormality.

For example, `abnormalBehavior` could indicate whether a request wants to copy unusually large amounts of data from one medium or location to another. Or `abnormalBehavior` could encapsulate a risk-based or statistical analysis of the requestor's behavior, and so abstract quantitative results into a binary "decision". In another context, `insider` may declare those sets of requests for which either a manager assigns pay increments to non-managers or for which non-managers rate the performance of their managers, as in

$$\begin{aligned}&(\texttt{manager} \wedge \texttt{assign} \wedge \texttt{payIncrease}) \vee \\ &(\neg\texttt{manager} \wedge \texttt{rate} \wedge \texttt{managerPerformance})\end{aligned} \tag{3}$$

We point out that insiders defined as in (3) are not defined in mere terms of roles,

but in terms of the combination of role, object, and action. In particular, it views managers as "outsiders" when it comes to rating the performance of managers. Of course, such combinations could be enriched with additional context predicates.

Such request predicates also leverage policy analysis. Since p@`Grants` and `insider` are both request predicates, we can examine whether the implication

$$p@\texttt{Grants} \rightarrow \texttt{insider} \tag{4}$$

is valid. If so, then policy p will not grant requests unless they stem from those who are declared to be insiders. This would show that policy p cannot be persuaded to grant access to outsiders, stipulating that outsiders are defined by $\neg$`insider`.

This approach does not impose the exclusive use of static analysis methods though. A dynamic version of (4) would test its validity at run-time, and adapt policy behavior accordingly, if needed. We can express this in a parameterized module:

```
pol grantOnlyInsiders(P : pol, insider : reqs) {
  (grant if P@grants & insider) > deny
}
```

The idea here is that the module takes a policy `P` and a request predicate `insider` as input, and retrofits policy `P` such that the retrofitted policy `grantOnlyInsiders(P,insider)` will grant only if the request is in the "set" `insider` of interest and the policy `P` would grant that request. The retrofitted policy `grantOnlyInsiders(P,insider)` denies all other requests, including those that don't satisfy `insider` but would be granted by policy `P`. This ensures that (4) holds in all executions of the access-control system, but now for the invocation `grantOnlyInsiders(P,insider)` instead of for policy `P`.

Finally, we illustrate how policy languages that meet our requirements could realize the approach we had suggested in Section 3, which uses trustworthiness to inform a risk-evaluation strategy for access control. The parameterized module

```
bool tooRisky1(R: req; riskThreshold : double) {
  return (cost(R) > riskThreshold)
}
```

returns a Boolean decision and takes as input a single request `R` and a threshold `riskThreshold` for the risk the access-control system is willing to accept when granting this request. The declaration of module `tooRisky1` is agnostic about how the real cost of granting request `R` is computed within expression `cost(R)`. Cheng et al. [8] provide a numer of explicit formulae for computing such costs in the context of multi-level security.

An alternative implementation is module

```
bool tooRisky2(R: req; riskThreshold : double) {
  return (cost(R.object,R.action) / trustworthiness(R.subject)
         > riskThreshold)
}
```

Here risk is perceived as being too high if the estimated cost (which is based on

the requested object and the requested action) divided by the requestor's trustworthiness is strictly above a specified risk threshold. Although the module `trustworthiness(S : subject)` is agnostic about how the trustworthiness of a subject is being computed, it does stipulate that this trustworthiness is a function of the subject alone, and not of the nature of a specific request. The advantage of this global approach to trustworthiness is the low overhead in managing trust state (including dymamic updates of such state). Some applications, though, may require more local notions of trustworthiness (as in `tooRisky1`) that are, for example, a function of objects and actions.

In any event, the evaluation of modules such as `tooRisky2` makes the access-control system well aware of the degree of trustworthiness of subjects, and can make its access-control decisions depend on such varying degrees.

We illustrate how these functions could be used to make a policy risk-aware. For example

```
pol riskFilter(P : pol) {
  (deny if tooRisky1(this)) > P
}
```

is a module that takes policy `P` as input and then produces a policy `riskFilter(P)` that denies if the presented request `this` is judged to be too risky; otherwise, it will behave as policy `P`.

Of course, functions like `tooRisky2` are defined in terms of functions such as `cost` and `trustworthiness`, so we will need to extend the access-control architecture to include components that can evaluate such functions. The Chinese Wall policy is probably the earliest example of an access-control policy that requires the evaluation of a "helper" function to make an authorization decision. For the Chinese Wall policy this function is particularly simple, but it illustrates that the integration of historical information with access-control mechanisms can be achieved. We believe that the harnessing and co-ordination of tools such as host-based intrusion detection systems and reputation systems within a policy-driven access-control framework offers a powerful and principled way of tackling the threat posed by insiders.

5 Concluding remarks

In [5] it has been shown that primitives such as p@`Grants` and p@`Denies` are definable as request predicates if policies are constructed by combining request predicates with operators similar to those discussed in this paper. In that work, request predicates were atomic, without structure. But it is easy to extend policy composition and the above primitives to request predicates that support composition operators from propositional logic, as demonstrated in the talk [10] at the Dagstuhl Seminar [2].

The ideas put forward in this paper are perhaps closest to the work by Probst et al. in [19]. The authors point out that the predominant approach to addressing the insider problem is to maintain audit logs for "post mortem" analysis. They suggest a formalism for high-level access-control models, where models can then be mapped into terms of a process algebra. Those terms are then subjected to static

analyses that yield safe overapproximations of users capabilities and restrictions. Our proposal extracts expressions that state such capabilities and restrictions, but we can not only use them for static analysis but also as wrappers that can extend or restrict given policies at run-time.

In future work, we intend to more formally design a language in which access-control policies could be made risk-aware in the manner described in this paper. We also mean to examine existing case studies of insider attacks mediated through access-control mechanisms to determine whether our approach could have detected, prevented or otherwise mitigated such attacks if policies has been made risk-aware. A further avenue for future work would be the specification of an access-control architecture incorporating the usual decision and enforcement points as well as components for computing and maintaining trustworthiness and other contextual quantitative data.

References

[1] Bishop, M., S. Engle, S. Peisert, S. Whalen, and C. Gates, *Case Studies of an Insider Framework*, Proc. of Hawaii International Conference on System Sciences, pp. 1–10, IEEE Computer Society Press, 2009.

[2] Bishop, M., D. Gollmann, J. Hunker, and C. W. Probst, *Countering Insider Threats*, Dagstuhl Seminar 08302, Leibnitz Center for Informatics, 18 pp., Dagstuhl Seminar Proceedings, ISSN 1862 - 4405, July 2008.

[3] Bishop, M., *Panel: The Insider Problem Revisited*, Proc. of NSPW 2005, ACM Press, 2006.

[4] Brackney, R., and R. Anderson, *Understanding the Insider Threat*, Proc. of a March 2004 Workshop, RAND Corp., Santa Monica, California, March 2004.

[5] Bruns, G., and M. Huth, *Access-Control Policies via Belnap Logic: Effective and Efficient Composition and Analysis*, Proc. of CSF 2008, pp. 163–178, IEEE Computer Society Press, 2008.

[6] Bruns, G., D. S. Dantas, and M. Huth, *A simple and expressive semantic framework for policy composition in access control*, Proc. of FMSE 2007, pp. 12–21, ACM Press, 2007.

[7] Chakraborty, S., and I. Ray, *TrustBAC: integrating trust relationships into the RBAC model for access control in open systems*, Proc. of SACMAT '06, pp. 49–58, ACM Press, 2006.

[8] Cheng, P.-C., P. Rohatgi, C. Keser, P. A. Karger, and G. M. Wagner, *Fuzzy Multi-Level Security: An Experiment on Quantified Risk-Adaptive Access Control*, IBM Research Report, RC24190 (W0702-085), Computer Science, February 2007.

[9] Hoffman, K., D. Zage, and C. Nita-Rotaru, *A Survey of Attack and Defense Techniques for Reputation Systems*, To appear in ACM Computing Surveys, Volume 41, Issue 4, December 2009.

[10] Huth, M., *A Simple Language for Policy Composition and Analysis*, Talk given at [3]. `www.doc.ic.ac.uk/~mrh/talks/Dagstuhl08.pdf`

[11] D. Jackson, *Software Abstractions: Logic, Language, and Analysis*, MIT Press, 2006.

[12] Jones, S. P., J.-M. Eber, and J. Seward, *Composing contracts: an adventure in financial engineering (functional pearl)*, ACM SIGPLAN Notices 35(9): 280–292, ACM Press, 2000.

[13] Lee, A., and T. Yu, *Towards a dynamic and composable model of trust*, Proc. of SACMAT'09, pp. 217–226, ACM Press.

[14] Moore, A. P., D. M. Cappelli, and R. F. Trzeciak, *The "Big Picture" of Insider IT Sabotage Across U.S. Critical Infrastructures*, Technical Report CMU/SEI-2008-TR-009, ESC-TR-2008-009, Carnegie Mellon University, May 2008.

[15] The New York Times, *French Bank Says Rogue Trader Lost $7 Billion* , 25 January, 2008.

[16] Patzakis, J., *New Incident Response Best Practice: Patch and Proceed is No Longer Acceptable Incident Response Procedure*, Guidance Software, Pasadena, California, September 2003.

[17] Park, J. S., and J. Giordano, *Role-Based Profile Analysis for Scalable and Accurate Insider-Anomaly Detection*, Proc. IPCCC'06, 2006.

[18] Park, J. S., and J. Giordano, *Access Control Requirements for Preventing Insider Threats* , Proc. ISI'06 LNCS 3975, pp. 529–534, Springer, 2006.

[19] Ch. W. Probst, R. R. Hansen, and F. Nielson, *Where Can an Insider Attack?*, Proc. of FAST'06, LNCS 4691, pp. 127–142, Springer, 2006.

[20] Ch. W. Probst, and J. Hunker, *The Risk of Risk Analysis-And its Relation to the Economics of Insider Threats*, Proc. of the Eighth Workshop on the Economics of Information Security (WEIS 2009), June 2009.

[21] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, *Role-Based Access Control Models*, IEEE Computer 29(2): 38–47, 1996.