We Have Met the Enemy and He Is Us

Matt Bishop, Sophie Engle,
Sean Peisert, Sean Whalen
University of California, Davis
Davis, CA
{bishop,engle,peisert,whalen}@cs.ucdavis.edu

Carrie Gates
CA Labs
Islandia, NY
carrie.gates@ca.com

ABSTRACT

The insider threat has long been considered one of the most serious threats in computer security, and one of the most difficult to combat. But the problem has never been defined precisely, and that lack of precise definition inhibits solutions. This paper presents a precise definition of insider threat, and shows how the definition enables an analysis of the set of problems traditionally lumped into "the insider threat". It introduces a hierarchy of policy abstractions, and argues that the discrepancies between the different layers of abstraction expose the potential for insider threat. It also presents a methodology for analyzing the threat based upon our definitions. In the process, we introduce Attribute-Based Group Access Control, a generalization of the Role-Based Access Control model that allows any attributes to define a group. We apply this to the insider threat by defining groups based on access capabilities, and using that to identify users with a high level of threat with respect to high-risk resources.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection; D.4.6 [**Operating Systems**]: Security and Protection; H.1 [**Information Systems**]: Models and Principles

General Terms

Security,Management

Keywords

Access control, Insider threat, Security policy

1. INTRODUCTION

The *insider threat* identifies a serious threat to computer security. It describes a breach of trust by people *within* an organization or system, as contrasted to external entities for whom firewalls and other mechanisms can deny access. Insiders are different. They require access to perform their jobs, and denying this access is tantamount to a denial of service attack. But traditional methods of preventing or detecting attacks from outsiders such as intrusion detection and access control may be infeasible. Further, the inability to restrict this access to a desired level of granularity due to technical, administrative, or other reasons creates a "gap" between the least level of access needed for a task to be performed, and the least level of access that can actually be specified and maintained. This constant struggle of requirements creates the potential for insiders to abuse their privileges, a problem both of critical significance and extensive study (see Section 5).

Insider threat is ill-defined in the literature today, and is often based on a narrow or ambiguous definition of an insider. For example, "insider" connotes a binary classification—an entity either is an insider, or is not. Additionally, insider definitions assume that security policies define this distinction. We challenge both of these assumptions.

We present a definition of insider threat that does not depend on a binary definition of an "insider", but reflects a nuanced notion that there are degrees of "insiderness" and different types of insider threats. What is appropriate to detect insider threat of one type may well be inappropriate to detect threats of a different type. Our definition extends into domains that include physical, social, and cyber security influences; indeed, our definition is based upon discrepancies between policy abstractions that capture each of these domains.

From our definition of insider threat, we derive an approach to examine an organization to determine the resources that are placed at risk, and the entities that place those resources at risk. This approach combines the resources in question with the access required to be considered a threat within a particular domain. These access control domains can be mapped to the degree of potential damage and, in combination with the probability of attack, also map to the level of threat [14]. The result indicates both where detection should focus and the degree of insider threat any one person presents.

We begin by using the security policy as the basis for our definition of insider threat. We then present ABGAC (Attribute-Based Group Access Control)—a generalization of RBAC capable of identifying "insiderness" with respect to a resource. Finally, we illustrate how to use ABGAC to analyze the threat posed to high-risk resources with an example application.

2. WHAT IS THE INSIDER THREAT?

In order to understand insider threat, we need to turn first to the cornerstone of all definitions of "security", namely the *security policy*. Examining the assumptions underlying that statement of security requirements will provide the background necessary to define the insider threat in a useful way.

2.1 Security Policy Layering

Ideally, a security policy states what actions are authorized for a specific user and purpose. For example, a security policy might state that "Yasmin is authorized to read medical records for the purpose of treating patients". If Yasmin deletes the medical records, she is violating the security policy. She is also violating the security policy if she reads the medical records for the purpose of selling the information in them to insurance companies. Additionally, the security policy is violated if anyone else uses Yasmin's user account to read medical records.

This example exposes a key limitation in security policies: they may state rules that are not feasible to implement.[1] Computer systems only understand user accounts and actions, not people and purpose. Restricting the actions of an individual on a system requires that the individual be mapped to a user account and the action in the security policy be mapped to actions on the computer, and *vice versa*. Suppose that the user Yasmin is authorized to use the user account `yasmin` on the system. The system is unable to determine if someone *other* than Yasmin is using her account (or whether Yasmin is using anyone else's account). Additionally, the system is unable to discern the *intent* of a user; it can only discern *actions*. If the account `yasmin` accesses medical records, the system has no way to determine *why* the user is accessing those records—so the system cannot determine whether to allow access (because the user wants to treat patients) or to deny access (because the user wants to sell the information in them).

Of course, a security policy would indicate that Yasmin is the only person authorized to use the account `yasmin`. It might also indicate that the account `yasmin` can only access records for the purpose of treating patients. However, these are procedural requirements that the computer cannot enforce. Thus, the security policy *as implemented on the computer* is not "Yasmin is authorized to read medical records for the purpose of treating patients". It is "user `yasmin` is authorized to read medical records". Such a policy does not refer to a real person or that person's purpose. This limitation of the implemented security policy gives a user the ability to misuse her privileges.

In short, a security policy is really not a single, abstract statement of requirements. It is instead an *instantiation* of such requirements in a particular environment. This suggests viewing a security policy not as a single prescription but as a set of layers, in which the top layer (abstraction) is refined by constraints in the environment and system(s) in successive layers until the layer corresponding to the instantiated policy is reached. This idea of "layers of abstraction" is a common concept in computer science. Two obvious examples are the ISO/OSI network layers, and operating system layers such as in Dijkstra's THE system [6]. In any such layering, successive layers refine aspects of the previous layers. This refinement will create discrepancies. These discrepancies are the key to uncovering the insider threat.

Carlson's Unifying Policy Hierarchy captures this notion of layering security policies [4]. This hierarchy distinguishes what is *ideal*, *feasible*, *practical*, and *possible*. By examining when there are policy violations across levels in this hierarchy, we are able to define broad categories of insider threat problems.

Figure 1 shows the four layers of abstraction of Carlson's hierarchy. The highest layer is the *Ideal Oracle*, which can respond to a policy query for any subject, object, action, and environment (which includes intent) tuple. For each query, the oracle[2] will state whether the policy allows or disallows the subject to take the action on the object in the given environment. The response captures the notion of an "ideal policy". For example, given the privilege "Yasmin (subject), medical records (object), read (action)" and the environment (intent) "treating patients", the oracle will state the privilege is authorized for the given environment.

The next layer is the *Feasible Policy*, which represents what can in practice be captured on an actual system. This distinction captures the notion that a system cannot distinguish between the person Xander who logged in with Yasmin's username and password, and Yasmin herself logging in with her credentials. Additionally, the Feasible Policy discards the notion of purpose entirely. The Ideal Oracle, as an all-knowing idealized abstraction, can make such distinctions when responding to policy queries. The Feasible Policy is similar to what Schneider, et al., refer to as a *enforceable* policy [18].

From the space of all feasible policies on a system, a single policy will be selected and configured. This is the *Configured Policy*. As the size and complexity of the policy configuration increase, the number and complexity of issues related to the administration or management of that configuration also increase. A mismatch between the Feasible Policy and Configured Policy levels rises from these issues. The departure of an employee from a company without prompt removal of their privileges from the system is one example.

Finally, the *Instantiated Oracle* represents the actual conditions on a system. While a machine may be configured with a specific policy in mind, policy violations may occur due to intended or unintended exploits or attacks which may elevate the privileges of users such as exploiting buffer overflow or race condition vulnerabilities.

2.2 Defining the Insider Threat

Carlson's hierarchy allows us to characterize the notion of insider threat. The *insider threat* exists whenever someone has more authorized privileges at a lower policy level than at a higher policy level. These mismatches indicate an unauthorized increase in privilege, which exposes where there is a potential for abuse. For example, a mismatch between the Ideal Oracle and the Feasible Policy can occur due to social engineering. In our previous example, the Ideal Oracle knows that Xander is forbidden to log in as Yasmin, but the Feasible Policy cannot know that Xander found Yasmin's password scribbled on a piece of paper taped to her monitor. Thus, the set of privileges Xander has at the Feasible Policy level may be greater than he has at the Ideal Ora-

[1] Policies suffer from other limitations, too; for example, they may be incomplete or evolving. For purposes of the following analysis, we assume the policy components that define access are fixed.

[2] We use the term *oracle* similarly to how a "random oracle" is used in cryptography.

UNIFYING POLICY HIERARCHY		
Level	**Domain**	**Description**
IDEAL ORACLE	all possible (s, o, a, e) tuples	Captures notion of an ideal policy even if such a policy isn't explicitly defined.
FEASIBLE POLICY	system-definable (s, o, a) tuples	Represents what in practice can be captured on an actual system.
CONFIGURED POLICY	system-defined (s, o, a) tuples	Represents the policy as configured on an actual system.
INSTANTIATED ORACLE	all possible (s, o, a) tuples	Captures what is possible on an actual system.

s: subject o: object a: action e: environment/intent

Figure 1: Four levels of Carlson's Unifying Policy Hierarchy.

	Policy Level				
Action	**Ideal**	**Feasible**	**Configured**	**Instantiated**	**Violation***
Xander authenticates as `xander`.	allow	undefined	undefined	possible	ID $\neq$ FP
Account `xander` accesses website†...	disallow	allow	allow	possible	ID $\neq$ FP
...to check the weather.	allow	undefined	undefined	possible	ID $\neq$ FP
...to expose system to vulnerability.	disallow	undefined	undefined	possible	ID $\neq$ IN
Web browser leaks user password‡.	disallow	disallow	disallow	possible	CP $\neq$ IN
Yasmin authenticates as `xander`.	disallow	undefined	undefined	possible	ID $\neq$ IN

ID: *Ideal Oracle* FP: *Feasible Policy* CP: *Configured Policy* IN: *Instantiated Oracle*

∗ Only some of the more "severe" violation are given. † Assume the Ideal Oracle disallows web browsing for no purpose (waste of company time), or for any purpose other than checking the current weather. ‡ Assume the vulnerable web browser was exploited by Yasmin to reveal the user password.

Figure 2: Web Surfing Example: Possible Policy Violations.

cle level should he exploit this social aspect of the Feasible Policy-Ideal Oracle semantic gap. These gaps exist between any two levels of the hierarchy and define different types of insider threats.

These differing problems can be related to several real world examples. A mismatch between the Ideal Oracle and Feasible Policy, where a user[3] elevates her privilege at the feasible level by exploiting the social or intent gap, captures the notions of social engineering and covert channels. Mismatches between Feasible Policy and Configured Policy are demonstrated by administrative issues such as out-of-sync policies retaining privileges of former employees, for which a solution may be impractical on certain time scales for large companies with high turnover. The Instantiated Oracle and Configured Policy differences are shown by attacks on a system such as buffer overflows or race conditions.

In addition to describing different types of insider threat, the policy hierarchy also captures the notion of intent. Consider Xander surfing the web with a browser vulnerable to a remote exploit. He unknowingly surfs to a site that exploits this vulnerability. If the attacker, Yasmin, gains access to Xander's account, she then obtains the rights of an authorized user by inheriting the privileges of Xander. If Xander is unaware, he lacks intent to violate the policy. In this case, Yasmin would pose an insider threat because of a discrepancy between the Configured Policy (only Xander can access the system using his account) and the Instantiated Oracle (because of the exploit, Yasmin can now access the system using Xander's account). That is, the Instantiated Oracle allows the event because the Configured Policy is not sufficiently expressive. Alternately, Xander could intentionally surf to the site and allow his account to be compromised, claiming a lack of intent if his actions are detected. In this case, Xander poses an insider threat because of a discrepancy between the Ideal Oracle (which, as an all-knowing entity can make policy decisions that include the intent of the user) and the Feasible Policy (which cannot restrict access based on intent). That is, though the Ideal Oracle prohibits the event, the Feasible Policy allows it because it is insufficiently expressive. Figure 2 illustrates this in more detail.

In what follows, we focus on the insider threat related to the gap between an Ideal Oracle and a Configured Policy. For purposes of exposition, we assume that the system is configured correctly. Hence the Feasible Policy is the same as the Configured Policy. If we did not make this assumption, then errors in configuration would result in discrepancies between these two policy layers. The security policy at the Configured Policy level is represented by the access control rules employed by an organization. The insider threat can thus be defined with regard to two primitive actions:

1. violation of the Ideal Oracle abusing access granted by the Configured Policy, and
2. violation of the Configured Policy by abusing access present in the Instantiated Oracle.

[3]We use the term "user" very broadly to include anyone that has cyber, social, or physical access to a resource.

In the first case, the attacker uses her legitimate access to perform some action that is contrary to the Ideal Oracle, such as leaking sensitive data to some third party. Here the attacker has legitimate access to the data or resources, but uses that access to provide the information to someone who does not have access. While the attacker has "legitimate" privileges at the Configured Policy level, she is only authorized to execute those privileges for the purpose of performing their authorized task at the Ideal Oracle level—and leaking information is not such a task.

In the second case, the attacker uses her access to extend their privileges in a manner that breaks both the access control and security policies. An example of such a breach occurs when a user might have a legitimate capability to log into a particular system, but then abuses that privilege to gain illegitimate root-level access to the system (e.g., by exploiting some system vulnerability such as a buffer overflow or race condition). The user is misusing her privileges to log on the system for an unauthorized purpose. As before, this action violates the Ideal Oracle, but not the Configured Policy, because the Ideal Oracle is more expressive. The attacker is also taking advantage of the disconnect between the Configured Policy and Instantiated Oracle levels. By exploiting a vulnerability on the system, the Instantiated Oracle grants more access than the Configured Policy.

Bishop's definition [1]—"a trusted entity that is given the power to violate one or more rules in a given security policy... the insider threat occurs when a trusted entity abuses that power"—suggests that an insider must be defined with respect to some set of rules that is part of a security policy. In our model, these policy rules are defined at different levels of the policy hierarchy. Our previous example would say that Xander is an insider *with respect to physical access to Yasmin's monitor* because he has that access, and thus is able to exploit it to gain additional access that he is not authorized to have (specifically, access to Yasmin's account).[4] More generally, in this situation an "insider" can be any individual person identified by the Ideal Oracle who has access to the accounts defined by the Feasible Policy.

Previous definitions give rules or descriptions intended to allow the reader to determine who is an "insider." This resulted in a binary distinction: an entity was either *an insider* or *not an insider*. The above analysis produces a partial ordering of degrees of "insiderness", and the access control rules for an organization can be used to develop these degrees. For example, Xander and Yasmin might both have privileges within an organization, and thus both represent "insiders" in the binary sense. However, for some resources Yasmin might have more privileges than Xander, and so therefore poses a greater insider threat than Xander with respect to those resources. At the same time, Xander may pose a greater insider threat than Bob with respect to other resources. This definition extends to physical as well as cyber security. For example, if there is a concern that printed documents might leave a building, then the rules used to define an "insider" would include access to paper printouts. A janitor is therefore an "insider", while Xander and Yasmin, both of whom work remotely, are not.

By using this definition, both researchers and security personnel can focus their efforts on detecting those attackers who are likely to cause the most damage to an organization by focusing on those resources of greatest value. We now present one such approach of identifying the potential attackers and the resources that are at risk with respect to those attackers.

[4]This is not the only rule with respect to which Xander is an insider; it is a simple one, though. Other possible rules are left to the reader's imagination.

3. HOW DO WE IDENTIFY THE INSIDER THREAT?

The above characterization of the insider threat suggests that we can determine the "insider threats" by examining the differences in access abilities of entities between successive policy abstraction layers. In our example of Xander's accessing Yasmin's account, the discrepancy of interest is the difference in access of the *individual* Yasmin (at the Ideal Oracle abstraction level[5]) and the access of the *account* `yasmin` (at all lower policy abstraction levels). The difference means that whoever has *access to* Yasmin's account has the accesses that Yasmin has. So the above characterization bases "insiderness" on access at the Instantiated Oracle level.

In order to capture this notion of insiderness as a function of access to data or resources, we propose the *Attribute-Based Group Access Control* (*ABGAC*) model. This model is a generalization of role-based access control (RBAC) [7, 17]. Unlike that model, ABGAC assigns rights based on general attributes that may or may not be included in a person's job function, rather than on the specific job functions a person has within an organization. For example, one "group" might be the set of people who come to work after 5:00PM. A second "group" might be the set of all system administrators (in which case this group is also a role). An insider attack may arise from attributes other than job function (such as being in the building after 5:00PM). ABGAC can capture entities with the same attributes. RBAC would require the attributes to be job functions to do so. Figure 3 illustrates this in more detail.

Several features of RBAC generalize naturally to ABGAC. In particular, the notion of "role containment" now becomes "group containment", and separation of duty generalizes to avoiding conflicts not arising from jobs but from other factors. For example, if Sam and Robin are married, and Sam owns a company the worth of which he knows will increase drastically, then if Sam advises Robin to invest in the company, there exists a clear conflict of interest. However, as no jobs are involved, and no separation of duty is involved, RBAC does not easily capture this situation.[6] In ABGAC, one can simply define two groups, the first containing those who know about the company's increase in value, and the second those who are related to members of the first. Then the "conflict of interest" simply says that members of the second group cannot perform an action forbidden to the members of the first, which is the straightforward generalization from RBAC's separation of duty rule.

We next construct the model from its "building blocks", and then show how to apply it. The goal of the model is to define groups of resources and, for each group, to define a set of users who have access to that group. The access, as will be noted, need *not* be authorized by the lowest level

[5]Under different (implicit) assumptions about identification, a lower layer would apply.

[6]One could make being married to Sam a job, and then use the mutual exclusion rule; but that stretches the notion of "job" quite far.

security policy (in the sense of Carlson's hierarchy); the access may be an artifact of the policy's inability to express a higher-level requirement precisely. An example of this is the restriction of access to an individual, Yasmin, that is instantiated at the lowest level by restricting access to the corresponding account `yasmin`.

3.1 Groups of Concern

For our purposes, the attributes of interest are descriptions of the protection domain of entities. Here, we mean "protection domain" in its broadest sense, not simply a technological listing of rights from capability lists (C-Lists) or access control lists (ACLs). So, the protection domain can include access rights to resources (systems, printers), documents, buildings, and generally any other object to which a user can have access. The protection domain can also include procedural access rights such as physical presence, or the ability to block access.

We begin by defining the building blocks of our model.

Definition. A *resource pair* is a pair consisting of a resource (entity) and an access mode describing one way in which that entity can be accessed. For example, a pair might be $(printer, write)$, which indicates the ability to write to a printer. The "access" need not be computer-based. For example, someone physically carrying the printer out of a building does not require read, write, modify, or create access to the printer. While it might seem to require delete access, the printer in fact is not destroyed; it is moved elsewhere, and is unavailable to the owners, but the people who remove it can still read the data on its hard drive.

Definition. A *resource domain* is a set of resource pairs. This describes a domain similar to the usual notion of protection domain, but includes physical and procedural access as well as cyber access. It is oriented towards the resource (object), not the process (subject). For example, if the ability to read and write a directory system enables a covert channel, an appropriate resource domain would consist of one resource pair for reading the channel and a second for manipulating the channel.

Once defined, the resource domains need to be ordered. This enables the organization to analyze the cost of restricting access to a particular resource and the benefit of restricting access to that resource, and balance the two. The ordering might be total, such as a linear ordering, or partial, using a vector of measurements taken over different axes. The value of resource domains should not be defined solely by a systems administrator, but rather as a joint effort between the senior executives and the security administrators. Note that, once ordered, the resource domains can be combined into groups (containing a *contiguous* set of access control settings so that the order is maintained), where the group indicates the threat level a particular set of attributes represents.

Definition. An *rd-group* is a set of resource domains. Different resource domains may be related for the purposes of analysis, although singleton rd-groups (groups consisting of exactly one resource domain) will also prove useful. This definition is motivated by attacks that need access to multiple resource domains, for example an attacker who has access to one domain in order to read information, and can then exploit access to a second in order to transmit the information to an unauthorized party.

It is easy to create the ordering of resource domains and define the rd-groups based on them. The set of all resources is clearly the most valuable; following that, a partial ordering based on the subset relation defines relative values of some of the groups. For those related by the ordering, the values should reflect the ordering. For those unrelated by the ordering, the organization may choose to impose a total ordering by placing a linear value on each rd-group.

Each rd-group induces a set of subjects that have all elements of the rd-group as subsets of the subject's protection domain:

Definition. A *user group* is the set of all subjects whose protection domains are a (possibly improper) superset of the associated rd-group.

User groups are created based on the protection domains of the associated users rather than on the job functions of the associated users (as in a role-based system). The users with access to the rd-groups with the highest value then represent those users who pose the greatest risk for insider threat. There is a natural ordering of user groups based on set containment.

It is tempting to assert that all this information can be obtained by an examination of the computer system. The resources are connected to the computer or network, and the access controls resident on the system (ACLs, C-Lists, authorization and authentication controls, etc.) define resource domains. A threat analysis then gives the rd-groups of interest, from which user groups can be derived. This only works if the sole avenues of attack are cyber. Unfortunately, that is rarely true, and we suggest that it is incorrect for most insider attacks. Examples include attacks that require social engineering, physical access, procedural gaffes, and other non-cyber access to the system or resources under attack. Hence the inclusion of non-cyber accesses in the definition of resource pairs. This is central to our approach to the insider threat.

Consider an Ideal Oracle that restricts access to electronic voting machines to election officials, including poll workers. In theory, this is easy to implement: keep the machines under lock and key until they are used. In practice, this is quite difficult in many counties. For example, in San Diego County in the June 2006 election, some poll worker supervisors took electronic voting machines home in order to be able to bring them to the polling stations when those opened. Some of these "sleepovers" lasted more than a week, during which time a family member or burglar had effectively unrestricted access to the system [21].[7] Thus, the user group associated with physical access to the e-voting system would include everyone who had access to those systems at the home where the machine was kept during the sleepover. It is infeasible to deny family members and others access to the home where the machine is being kept, so the Feasible Policy (that which can be implemented) would recognize that family members, at least, would have physical access to the system. This means the family members have privileges (physical access) they should not have, and the discrepancy—the fact that the lower layer policy can-

[7]The attackers would have to bypass physical seals, but as reported in the California Top-to-Bottom review of electronic voting systems, "the testers were able to compromise the AccuVote TSx completely by bypassing the locks and other aspects of physical security using ordinary objects." They also "found numerous ways to overwrite the firmware in the AccuVote TSx." ([2], p. 10).

	Attribute		
Name	**Job Function**	**Building Access**	**Server Access**
Wilma	System Administrator	Before 5pm	Physical, Remote
Xander	Help Desk	After 5pm	Remote
Yasmin	Janitor	Before 5pm	Physical
Zane	Janitor	After 5pm	Physical

	Group Members				
ABGAC Group Attribute	**W**	**X**	**Y**	**Z**	**RBAC Role**
Job: System Administrator	●				System Administrator
Job: Help Desk		●			Help Desk
Job: Janitor			●	●	Janitor
Building: Before 5pm	●		●		*Unclear*†
Building: After 5pm		●		●	*Unclear*
Server: Physical Access	●		●	●	*Unclear*
Server: Remote Access	●	●			*Unclear*

† Using ABGAC, we are able to use attributes directly to create groups. It is unclear how some of these groups would be created in RBAC. For example, it is unclear how to create a role based on job function that includes both **W**ilma, a system administrator, and **Y**asmin, a janitor.

Figure 3: Differences between ABGAC and RBAC.

not enforce the policy with the same level of detail as the higher layer policy—makes them insiders" for the purpose of analyzing insider threats.

In this paper, we do not confine our definition of "user group" to cyber access. Suppose Xander's account allows him to read confidential data. Being a person newly introduced to security, he has moved the Post-It note that has his password written on it from his monitor (where he thinks anyone can see it) to underneath his keyboard (where he thinks no-one will ever look). Yasmin, the janitor, notices it while cleaning Xander's desk. She can now log in as him, and the system will be unable to differentiate them. This effectively makes her an insider. Similarly, an attacker, Zoros, who can install (or persuade Xander to install) a keyboard sniffer gains exactly the same type of access. Here, Yasmin and Zoros are blocked by the Ideal Oracle (because writing passwords on Post-It notes and downloading anything over the web are forbidden), but in this site, the systems cannot be configured to prevent these compromises—hence, the Configured Policy effectively grants Yasmin and Zoros access that the Ideal Oracle denies them.

3.2 Analyzing the Insider Threat

As with all defenses, determining how to defend against insider attacks involves a cost/benefit analysis. Defending against these attacks are more difficult because security and usability are so much more closely in opposition. Fundamentally, some attackers will always be able to breach security. When the attackers define the Feasible Policy (for example, controlling the decision as to which system will be procured), the discrepancy between it and the Ideal Oracle cannot be resolved and corrected. For example, if the President of the United States were to read a classified memo on television that supported his or her policy, declassifying it only for the reading, the Ideal Oracle is clearly breached because the information in classified memos is supposed to be secret. But it is infeasible to prosecute a President for this type of violation, so the Feasible Policy allows the President to perform the momentary declassification.

Our question is how to determine the cost of the insider attack succeeding. The defenders can then analyze the benefits of not defending against that attack (cost savings, etc.) taking into account the likelihood of the attack.

We begin with the rd-groups. Define a set of rd-groups that contain the domains (sets of resource pairs) that are to be protected. These induce user groups, as noted earlier. Let U be the set of all user groups and D the set of all rd-groups. Then define $c : U \times D \to \mathbb{R}^n$, where $\mathbb{R}^n$ is the real-valued vector describing the cost of a compromise.

This leads to two minimizations of the insider threat. Either approach is valid depending on the goals of the defenders.

Consider the goal to be to minimize the impact of the attack. The function c induces a partial order over the vector elements of its range (if $n = 1$, of course, this is a total ordering). In fact, this partial ordering constitutes a lattice. The goal is to minimize each entry in the vector result. In some cases, where two potential outcomes are incomparable, management must decide which is preferable.

An alternate goal is to minimize the number of users who pose an insider threat. In some sense, this is quixotic, because identifying all the potential users requires prying into the life and habits of known users (including janitors and others with physical access, etc.). This is infeasible in most realistic situations. But if we minimize the number of *known* users posing an insider threat, we may reduce the number of *unidentified* users drastically. We do this by placing a special requirement on c, specifically that $c(u, d) \leq c(u', d')$ if and only if $u \subseteq u'$. That is, given two user groups associated with a resource group, then the cost of defending against the threat is smaller for the smaller of the groups.

Next comes developing the resource groups and user groups. The first step is to determine what are the important components of the resource domain relevant to some privilege (including physical access, or lack thereof). It is not necessary to provide all components and privileges, but rather only those that are relevant to the well-being of the organization and therefore those which at risk due to insider threat. For example, access to a particular printer or computer system might not be important, but the ability to print a particular document on that system might have value. A quick initial approach to determining the relevant parts of the resource domains for a system are to ask what resources are needed, and how they would be used, to compromise the system.

Identification of all users must be done concurrently with determining the resource domains for the lattice. As noted above, the initial users include not only direct employees, but also all contractors and out-sourcers (technical, clerical, janitorial, etc.), as well as any "special case" access (such as facility visitors or guest logins). The gathering of users then proceeds as would computing a transitive closure, adding in those users with indirect access through existing users.

Two observations make this daunting task more tractable. First, one can define *proxy users*, much as is done with RBAC roles acting as proxies for users. For example, it is infeasible to identify the particular users who may use malicious code downloaded via a web site, but one could create a "download malicious user" to represent any of those real people. In some sense, this is ascribing a function (although one not recognized as a legitimate job function) to the suspect users, and then conditioning membership based on the proxy user.

The second is the existence of trust in humans to correctly implement in the procedures used to enforce policies. If these procedures are implemented correctly, they will drastically limit the users to be added to the set. But trust in humans to correctly implement the procedures requires that the procedures be appropriate, enforceable, and effective. The first two attributes are the most difficult to ensure. For example, if the resources are medical records, then should the unauthorized action be to reveal the records to an insurance company, firing the employee is not an appropriate procedure to protect the records because it does not prevent the record from being leaked (though it is entirely appropriate to do so after the leak). Similarly, a requirement that no family member touch the electronic voting system during a sleepover (as described above) is appropriate, because if it happens the system can simply not be used, but not enforceable, unless the machine is subject to round-the-clock monitoring.

Once the resource domains and users have been identified, the second stage is to map the two together based on the access the users have. This can take either (or both) of two approaches:

1. Determine what a person can do. This thought process is similar in nature to that used when creating capabilities.

2. Determine who has access to a resource. This thought process is equivalent to that used in creating an access control list.

The result from these actions is to determine the user group associated with each rd-group. From this, one can define a simple cost function c that produces a lattice as described above. This enables an initial lattice to be constructed for any system that uses an existing access control list or capability list mechanism. From that point, one can iterate, adding in other types of accesses not captured by the ACLs or C-Lists.

4. EXAMPLE APPLICATION

Consider a company that is developing a system to do electronic recordation of real estate. As the true infrastructure of many countries such as the United States lies in ownership of property, if such a system could be compromised, the results would be exceptionally destabilizing because one could no longer prove ownership of real property. Hence any such system must preserve integrity and accountability [22]. The company developing this software is a multinational company, with developers in Russia, Nicaragua, and the United States. All work from home, using a virtual private network to access the company's servers located in Iowa. Given that this software is mission-critical, the problem is to identify the potential attackers in its development who could create "trap doors" in the software to allow data to be modified after all parties believe it to be signed properly. We begin by identifying the relevant resources.

Developers create and edit software on their home systems. They download and upload the software over a virtual private network (VPN) that they use to connect to a set of servers on which the code resides. The servers are located in Iowa, and the corporate office makes daily backups. So, for this (simplified) corporation, the resources are:

1. developer systems
2. VPN
3. servers
4. backup media

A more complete enumeration would distinguish between the systems and the software on the systems, and include any dongles or other hardware devices. For simplicity's sake, we omit these details.

After considerable analysis, management has concluded the insider threat that they will deal with is illicit modification of the software—a reasonable decision, given the financial constraints they operate under and given the mission of the software. The resource groups describe the types of access to the resources, which in light of the above involve some form of modification, and any rights needed in support of that goal. As an example, the backup media may be altered, removed from the premises, or destroyed, so prior versions of the software cannot be recovered. Thus, the resource group for the backup media would be:

$$\{(backups, write), (backups, remove), (backups, destroyed)\}$$

Similarly, the resource group for the servers would be

$$\{(servers, write), (servers, connect), (servers, login)\}$$

the last two resource pairs being necessary to perform the action in the first pair. Misconfiguring the VPN can enable a man-in-the-middle attack, so the resource group for the VPN would be

$$\{(VPN, configure)\}$$

Finally, accessing the developer system allows the user to modify the software directly (by writing it) or indirectly (by implanting malicious software that captures key strokes and gives them to the attacker, who can then impersonate the developer). So the relevant resource groups are

$$\{(developerstation, login), (developerstation, modify)\}$$

and

$$\{(developerstation, download)\}$$

The rd-groups of interest coincide with the resource groups, because access to only one resource is required. Again, in a full expansion of the analysis, additional rd-groups would link the resources that must be used to modify the software when multiple resources must be accessed, but for our example we confine ourselves to single resources.

Next, the user groups must be populated. Consider the user group induced by the rd-group for the developer workstations. The developers have access to those systems; they can login and modify software on the workstation. Further, they can download software, applets, and/or email with attachments. So they make up an obvious user group, one that reflects job function—just as RBAC would indicate.

But there are other relevant members of the group. Consider first the developers in Russia. They work from home, so when they work they must use some computer to log in. For our purposes, assume accessing the VPN requires special hardware (such as a dongle). Then the developer *must* access the VPN and development servers from home. Then everyone who has access to the system at home is in some sense an "insider" because, if the developer ever leaves the system connected to the VPN, anyone can access the developer servers. Similarly, those who have access to the computer could modify it to inject malicious logic to corrupt the data (programs being developed or programs used in the development). This includes friends who visit, and computer repair technicians. Finally, if the developer surfs to web sites, or receives email, or uses the computer to interact with other systems or people, those sites and people can compromise the system by exploiting vulnerabilities. This defines several classes of members of that group:

1. The developer's family, or others who live in the same home, form a group with unrestricted access to the computer.
2. Those who have access to the home (such as housekeepers, maintenance people, assistants, and so forth) have appropriate access, depending on their duties and whether they are under observation.
3. The computer system repair people have unrestricted access to the system but only during restricted periods of time.
4. Those who have electronic contact with the computer have enough access to try to compromise it by exploiting configuration or program vulnerabilities.

All these are defined by level of access, which may or may not be a product of job function. Note in particular the second and fourth groups above. These people have transient access because they are not permanently resident in the home, but their level of access while in the home ranges from very limited (if they are busy or the computer is under observation) to unrestricted for limited periods of time (if they enter the home when no-one is present).

A similar analyses shows that the user group induced by the backup media include the system administrators who make it, the senior executives who decide where they are to be stored, the people involved in the transfer, and the managers of the storage facility—as well as anyone with access to the areas where the backup media is kept during its creation, transportation, and storage. Note that the last set of people (those with physical access) are determined *not* by job function, but by their ability to reach the backup media.

The user groups induced by the other rd-groups are derived in a similar fashion.

These groups can be placed into the policy-based framework for the insider threat described above. For example, the Ideal Oracle states that developers can access the VPN to develop code, and that repair people can physically access the VPN equipment only in order to conduct repairs. This is infeasible. In particular, the VPN equipment is unable to determine the purpose of those who access it, nor is it able to detect or prevent physical access itself. Depending on the home life of the developer, the family and others in the home may, too—it is unrealistic to require a computer in a home to be kept in a locked room, accessible only to the developer. Hence the access of people at the Feasible Policy, Configured Policy, or Instantiated Oracle levels are greater than at the Ideal Oracle level.

We now consider the risk analysis. Two measurements are of interest: the value of the programs developed and the value of the particular employee. The senior executives have decided that protecting the programs requires that, above all, the backup media be protected because all versions of the code reside on them, so any prior version can be reconstructed; next, the servers containing the current version; and last, any computers used by the developers. They also believe that senior managers and system administrators are the most valuable personnel, followed by developers, followed by physical maintenance people (such as janitors). People are partitioned into groups based on the set of resources they have access to. The cost function c is therefore a vector of two elements, (effect of person attacking, value of resource).

Consider Tom, a system administrator working at the main office. He has access to the servers and the backup media. So the measurement for him is

$$(SA, \{backups, servers\})$$

Now, Kolya is a developer living in Moscow. He has access to a developer machine and the servers (to read and write code components), but not to the backups. So his measurement is

$$(D, \{developerstation, servers\})$$

Judy is the president of the company. She has access to the backup media but not to the servers nor to the developer systems. Her measurement is

$$(SE, \{backups\})$$

Finally, Angie is a janitor who sweeps out the machine room every night. Her measurement is

$$(PM, \{servers\})$$

and Natalya is Kolya's spouse, so her measurement is the same as Kolya's, or

$$(D, \{developerstation, servers\})$$

The set of resources defined here are the elements of the power set of

$$\{backups, servers, developerstation\}$$

Associated with each resource is its resource group, made up of resource pairs. As we consider only the effect of the access to the resource, we can give the resource group a single value. Because the rd-groups correspond to the resource groups in our example, the value of each rd-group is the same as the associated resource group. Identify each rd-group by the resource it refers to. Then let the values be assigned as follows:

$$\begin{aligned}
100 &\rightarrow \{backups, servers, developerstation\} \\
75 &\rightarrow \{backups, servers\} \\
60 &\rightarrow \{servers, developerstation\} \\
70 &\rightarrow \{backups\} \\
50 &\rightarrow \{servers\}
\end{aligned}$$

The values reflect, on a scale of 0 to 100, the seriousness of unauthorized modifications to the relevant rd-group. This imposes a linear ordering on the rd-groups.

In the above case, for example, the values and positions indicate that corrupted system administrators pose more of a risk to the integrity of the programs than corrupted senior management, which makes sense as senior management does not maintain, or have access to, the servers.[8]

The cost function used in this example is not realistic for several reasons. First, real cost analysis would take into account an attacker requiring multiple resources. Second, the cost would likely depend on the action taken with the resource, which we elided in the interests of simplicity. Third, the cost vector proposed here excludes intangibles such as reputation, appearance of integrity, and so forth. The cost of these is very difficult to evaluate but they are essential to a proper risk analysis, especially for a problem like the insider threat.

5. PRIOR WORK

Many researchers have investigated the problem of insider threat. However, most papers have either not precisely defined an insider, instead assuming that the user inherently understands the term, or have provided a definition too narrow, too broad, or too domain-specific to be useful. Without a consistent definition of an insider, each researcher develops her own definition that is particular to her own data set, situation, biases and assumptions.

As a result, research into the detection of insider threats can not necessarily be applied from one domain to another as the underlying model does not necessarily translate between the domains. Several definitions exist, and these can even be contradictory. For example, a RAND Corp. report defines an insider as "an already trusted person with access to sensitive information and information systems" ([3], p. xi). Elsewhere it defines an insider as "someone with access, privilege, or knowledge of information systems and services" ([3], p. 10), omitting the need for that person to be trusted. A paper on database security defines it as "a subject of the database [who] thereby has personal knowledge of information in the confidential field" [8]. A different report implicitly defines the insider as anyone operating inside the security perimeter ([12], p. 3), again ignoring trust and also knowledge of the systems. This reflects many non-computer situations. Even in insider trading, something that the U.S. Securities and Exchange Commission, the courts, and other government organizations have spent a significant amount of time investigating and prosecuting, there is a large grey area making it difficult to be certain whether a particular trade by an insider is legal or illegal without seeing a large, heavy suitcase full of $100 bills.

These (sometimes implicit) definitions are based on a notion of "trust", encapsulated in rules to identify those who are "trusted". Those who were not trusted are by definition not insiders. In some cases, a mechanism such as seniority in an organization may distinguish outsiders from insiders, but ultimately those mechanisms classify people based on rules. Thus, in any case, a set of rules distinguishes "insiders" from "outsiders". This results in a binary distinction: an entity is either *an insider* or *not an insider*. Our work eschews this, focusing instead on a graduated notion of "insiderness".

The problem of defining an insider is further complicated by the assumption of a perimeter that can be defined, such that someone inside the perimeter is an insider. However, the concept of distinct borders around an organization are blurring with the increased usage of mobile computing, outsourcing and contracting. Even in those cases where a distinct border can be defined, many definitions focus on technology borders and fail to consider physical borders and the ability to circumvent borders (for example, by social engineering). Again, our work in identifying insiderness ignores the concept of "perimeter" entirely, focusing instead on the concept underlying the notion of perimeter: access.

Many papers focus on handling the insiders rather than trying to define the problem. Two examples will suffice to make this point. Hu et al. [10] discuss using algorithms based on the role-based access control (RBAC) model to define insider threats and construct rules for intrusion detection systems to detect such attacks. However, they tacitly assume that an insider is defined by job function, and do not take into account physical or social insiders. Hasan et al. [9] discuss a threat model for storage systems that includes a brief analysis of insider attacks. They do not define insiders explicitly, beyond the phrase "insiders with privileged access", but they do observe that "a masquerader can launch insider attacks", suggesting that some insider attacks are social (masquerade) rather than cyber (using authorized privileges illicitly). Chinchani et al. [5] present a theory of assessment for the insider, defining insiders as "legitimate users who abuse their privileges, and given their familiarity and proximity to the computational environment, can easily cause significant damage or losses." This theory then models the user's view of the organization and of key information and capabilities. The focus is on the cyber insiders, and does not take into account those who are insiders because of physical or social reasons.

Others focus on the profile of the insider or the magnitude of the problem. Schultz [20] surveys many of the papers in this area, all of which focus on the classic triad of means, motive, and opportunity. Randazzo et al. [15] discuss the

[8]Note that the metric equates the system administrators and senior management. In practice, a more nuanced measure would need to take into account the ability of senior management to hire and fire system administrators, and a host of other issues that are ignored here to keep the example as simple as possible.

problem in the banking community. The 2007 Computer Security Institute (CSI) Computer Crime and Security Survey reported that "[i]nsider abuse of network access or email ... edged out virus incidents as the most prevalent security problem" ([16], p. 2) and that "37 percent of respondents attribute a percentage of their organization's losses greater than 20 percent to insiders" (*ibid.*, p. 12). Our approach captures "opportunity", namely who has access, and "means", namely what kind of access do they have, in rd-groups and user groups. Motive being a human subject, we must leave that to non-technical investigation.

6. FUTURE WORK

In practical terms, what can we do with this method? Returning to our original statement: our purpose is to enable legitimate users to do legitimate work, while seeking a way to defend against any user from taking illegitimate actions. For example, if an exploit is known, there are two possible defenses: prevent against the exploit, or allow it to happen but log it because knowledge of the action occurring (and the cause and effect of the action) is sufficient to undo (in whatever way) the present damage and attempt to ameliorate future damage [14]. When deciding which of these two options to pursue, it is clear that sometimes preventing the exploit is possible. However, at other times, the threat is low enough (or the opportunity cost of lost productivity is high enough) that actions must simply be allowed to occur, and they must be logged, instead of being prevented.

Any method of logging, of sufficient granularity [13], can be used to address this issue. However a systematic approach, using a model tailored to the levels of abstraction that we have indicated, would have the greatest possibility of success at the lowest cost. For example, one method of implementing such a solution is to identify the *goals* of attackers and compare those goals to the actual capabilities of those attackers. Those reference points, and knowledge of the deterministic operation of the system could be used to design "attack trees" of sorts [19], which can then be used to guide the forensic data to be logged [14]. That data, when overlaid on top of the attack graphs, can help guide the understanding of the actions of the attackers without necessarily interfering in legitimate actions of users, and thus generating false positives where they can be least afforded (this technique has also been used successfully to correlate network intrusion detection alerts [23]).

The notion of a hierarchy of policy abstractions suggests many avenues for future work. For example, we have defined the insider threat in terms of discrepancies between the levels (specifically, where a lower level gives a user more access than does the higher level). Can we broaden the use of these layers to capture notions of vulnerabilities? For example, a configuration vulnerability arises when the Configured Policy disagrees with the Feasible Policy. This suggests that some systems may have *inherent* vulnerabilities for specific types of policies, such as the Linux system without ACLs described above. This area needs more exploration, because the study of vulnerabilities lacks a firm theoretical foundation.

Another interesting question arises from the issue of "insiders by omission", in which a user is *denied* privileges at one level that a higher level authorizes her to have. First, how serious is this problem? All the studies and surveys of the insider threat focus on insiders who have too much access. What problems arise when a responsible party has too little access? This can also be seen as an insider attack by the developers of the Feasible Policy, Configured Policy, and the Instantiated Oracle that deny the defender adequate access to defend. The nature of this problem is unclear, and merits study.

7. CONCLUSION

We have presented an initial approach to defining the insider threat problem. While the majority of existing research implicitly defines an insider as a binary condition (one is either an insider or not), this paper takes the approach of defining insiderness based on access attributes. More specifically, we have defined a lattice consisting of rd-groups on one axis and users (not roles) on the other axis. By ordering resources based on their value, we can then group them by their value. By then grouping users according to their ability to access resources, we can provide a continuum of insiderness. This allows researchers and security personnel to focus on those who pose the greatest threat to an organization.

Finally, the ABGAC model introduces a notion of generalized groupings, mimicking the idea of computer "group". The focus on ABGAC using access as a defining mechanism for groups, including non-cyber types of access such as physical or social (*i.e.*, social engineering) raises issues that other models using access *control* mechanisms do not deal with. The difference is between what is authorized (by the access control mechanisms) and what is possible (for example, by circumventing those mechanisms). The introduction of the possible, rather than the allowed, seems useful as security deals with what "is", rather than with what is "meant to be".

Acknowledgements. Sophie Engle was supported by grant H98230-07-1-0234 from the Department of Defense to the University of California at Davis. Matt Bishop was supported by grant CND-0716827 from the National Science Foundation to the University of California at Davis. The views and conclusions expressed in this paper are those of the author, and not necessarily those of any funding agency.

Sean Peisert was supported by grant 2006-CS-001-000001 from the U. S. Department of Homeland Security, under the auspices of the Institute for Information Infrastructure Protection & I3P research program. The I3P is managed by Dartmouth College. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security, the I3P, or Dartmouth College.

The title of this paper is from a comic character named Pogo, drawn by Walt Kelly [11].

8. REFERENCES

[1] Matt Bishop. Position: "Insider" is Relative. In *Proceedings of the 2005 New Security Paradigms Workshop (NSPW)*, pages 77–78, Lake Arrowhead, CA, October 20–23, 2005.

[2] Matt Bishop. Overview of red team reports. Technical report, Office of the California Secretary of State, Sacramento, CA, July 2008.

[3] R. Brackney and R. Anderson. Understanding the Insider Threat: Proceedings of a March 2004 Workshop. Technical report, RAND Corporation, Santa Monica, CA, March 2004.

[4] Adam Carlson. The Unifying Policy Hierarchy Model. Master's thesis, Dept. of Computer Science, University of California at Davis, June 2006.

[5] R. Chinchani, A. Iyer, H.Q. Ngo, and S. Upadhyaya. Towards a Theory of Insider Threat Assessment. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, pages 108–117, June 28–July 1, 2005.

[6] E. W. Dijkstra. The Structure of the THE Multiprogramming System. *Communications of the ACM (CACM)*, 11(5):341–346, May 1968.

[7] D. F. Ferraiolo and D. R. Kuhn. Role Based Access Control. In *Proceedings of the Fifteenth National Computer Security Conference*, pages 554–563, October 1992.

[8] Robert Garfinkel, Ram Gopal, and Paulo Goes. Privacy Protection of Binary Confidential Data Against Deterministic, Stochastic, and Insider Threat. *Management Science*, 48(6):749–764, Jun 2002.

[9] Ragib Hasan, Suvda Myagmar, Adam J. Lee, and William Yurcik. Toward a Threat Model for Storage Systems. In *Proceedings of the 2005 ACM Workshop on Storage Security and Survivability (StorageSS)*, pages 94–102, New York, NY, USA, 2005. ACM.

[10] Ning Hu, Phillip G. Bradford, and Jun Liu. Applying Role Based Access Control and Genetic Algorithms to Insider Threat Detection. In *Proceedings of the 44th Annual ACM Southeast Regional Conference (ACM-SE)*, pages 790–791, New York, NY, USA, 2006. ACM.

[11] Walt Kelly. Zeroing In On Those Polluters: We Have Met the Enemy and He Is Us. *The Best of Pogo*, 1982.

[12] J. Patzakis. New Incident Response Best Practices: Patch and Proceed is No Longer Acceptable Incident Response. Technical report, Guidance Software, Pasadena, CA, September 2003.

[13] Sean Peisert, Matt Bishop, Sidney Karin, and Keith Marzullo. Analysis of Computer Intrusions Using Sequences of Function Calls. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 4(2):137–150, April–June 2007.

[14] Sean Philip Peisert. *A Model of Forensic Analysis Using Goal-Oriented Logging*. PhD thesis, Department of Computer Science and Engineering, University of California, San Diego, March 2007.

[15] M.R. Randazzo, M. Keeney, E. Kowalski, D. Cappelli, and A. Moore. *Insider Threat Study: Illicit Cyber Activity in the Banking and Finance Sector*. US Secret Service and CERT Coordination Center, 2004.

[16] Robert Richardson. *2007 Computer Crime and Security Survey*. Computer Security Institute, 2007.

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

[18] Fred B. Schneider. Enforceable Security Policies. *ACM Transactions on Information and System Security (TISSEC)*, 3(1):30–50, February 2000.

[19] Bruce Schneier. Attack Trees: Modeling Security Threats. *Dr. Dobb's Journal*, 24(12):21–29, December 1999.

[20] E. E. Schultz. A Framework for Understanding and Predicting Insider Attacks. *Computers and Security*, 21(6):526–531, 2002.

[21] Marc Songini. E-voting security under fire in San Diego lawsuit. *Computerworld*, August 4, 2006.

[22] Thomas Walcott and Matt Bishop. Traducement: A Model for Record Security. *ACM Transactions on Information and System Security*, 7(4):576–590, Nov. 2004.

[23] Jingmin Zhou, Mark Heckman, Brennan Reynolds, Adam Carlson, and Matt Bishop. Modelling Network Intrusion Detection Alerts for Correlation. *ACM Transactions on Information and System Security (TISSEC)*, 10(1), February 2007.