UCDAVIS
COLLEGE OF ENGINEERING

Title:
Dynamic, Flexible, and Optimistic Access Control

Author:
Peisert, Sean, UC Davis
Bishop, Matt, UC Davis

Publication Date:
July 9, 2013

Series:
Computer Science

Permalink:
https://escholarship.org/uc/item/2rb5352d

Preferred Citation:
Sean Peisert and Matt Bishop, "Dynamic, Flexible, and Optimistic Access Control," UC Davis CS Technical Report CSE-2013-76, July 2013.

Keywords:
Access control, formal models, insider threat, logging, security policy

Abstract:
Traditional access controls have evolved from being static and coarse-grained to being dynamic and very fine-grained. However, a balance still must be struck: too little access inhibits usefulness, effectively creating a denial of service for people trying to do their jobs; and too much access invites breaches of security. "Break-the-glass" techniques and adaptive access control have previously been developed to address this issue. But gaps in these techniques still exist. We extend these techniques as follows: consider a system in which prohibitions fall into two classes. Core prohibitions prevent disaster, and are axiomatic to the system. Ancillary prohibitions, derived from core prohibitions, hinder the ability of an attacker to violate core prohibitions, but are not in and of themselves critical to the security of the system. We introduce *optimistic access control*, a framework in which core prohibitions are always enforced, and ancillary prohibitions are enforced only when a specific threshold is crossed. The threshold depends upon history, trust, and a variety of non-binary countermeasures. This control deals with many scenarios—including the insider threat and remote access with limited communication—that are extremely difficult to address or even characterize using current techniques. Therefore, these controls address certain gaps. Finally, we present a formal mapping to lattice models, and describe implementation ideas and issues of this method in practice.

Copyright Information:
All rights reserved unless otherwise indicated. Contact the author or original publisher for any necessary permissions. eScholarship is not the copyright owner for deposited works. Learn more at http://www.escholarship.org/help_copyright.html#reuse

eScholarship
University of California

eScholarship provides open access, scholarly publishing services to the University of California and delivers a dynamic research platform to scholars worldwide.

Dynamic, Flexible, and Optimistic Access Control

Sean Peisert[†,*] peisert@cs.ucdavis.edu
Matt Bishop[†] bishop@cs.ucdavis.edu

[†] Department of Computer Science, University of California, Davis
[*] Computational Research Division, Lawrence Berkeley National Laboratory

July 9, 2013

Abstract

Traditional access controls have evolved from being static and coarse-grained to being dynamic and very fine-grained. However, a balance still must be struck: too little access inhibits usefulness, effectively creating a denial of service for people trying to do their jobs; and too much access invites breaches of security. "Break-the-glass' techniques and adaptive access control have previously been developed to address this issue. But gaps in these techniques still exist. We extend these techniques as follows: consider a system in which prohibitions fall into two classes. Core prohibitions prevent disaster, and are axiomatic to the system. Ancillary prohibitions, derived from core prohibitions, hinder the ability of an attacker to violate core prohibitions, but are not in and of themselves critical to the security of the system. We introduce *optimistic access control*, a framework in which core prohibitions are always enforced, and ancillary prohibitions are enforced only when a specific threshold is crossed. The threshold depends upon history, trust, and a variety of non-binary countermeasures. This control deals with many scenarios—including the insider threat and remote access with limited communication—that are extremely difficult to address or even characterize using current techniques. Therefore, these controls address certain gaps. Finally, we present a formal mapping to lattice models, and describe implementation ideas and issues of this method in practice.

1 Introduction

Access control mechanisms have historically been rigid. They may be dynamic, or they may be static, but given a particular access and an environment, they either deny or grant access consistently. This is necessary to ensure that the security of the system is well-defined and satisfies given specifications.

In real life, though, access is rarely rigid. Typically, people are trusted to do things until their abilities or trustworthiness is called into question. For example, banks handle large amounts of money, so embezzlement is a serious threat. Preventing embezzlement is easy: deny all employees access to withdraw or transfer any money. But this would prevent bank tellers from doing their job. So, banks use a less rigid approach. They allow bank tellers access to money, and institute validation procedures to detect embezzlement. When a teller's station fails verification, or even if the teller's trustworthiness is called into question (e.g., due to heavily gambling debts), then the teller's authority to access money is withdrawn. Similarly, consider a very different environment: in the United States, when a voter enters a polling station during an election, poll workers check

that the voter is listed on the rolls. If so, the voter votes. If not, the voter's ability to vote is not blocked, but restricted to casting a provisional ballot. Then, after the polls close but before the results are certified, the validity of the voter casting a vote can be checked and, if successful, the vote counted.

Optimistic access control is a dynamic access control mechanism with general, non-binary countermeasures that provides the ability to allow access initially, and then reduce or monitor accesses should the level of trust decrease below a threshold. It expands on and generalizes existing work in "break the glass" techniques and adaptive access control, while also adding a general notion of *countermeasures*. Optimistic access control is a solution for uncertainty. It addresses situations in which trust is not absolute but the risk of mistakenly allowing at least some actions to take place is greater than the need to allow at least some user to do their jobs—for example, a soldier to know their orders or a banker to make financial transfers.

Many scenarios exist where risk and trust, security and access are in conflict and cannot be resolved using traditional protection matrices, for example, when simply blocking access might have catastrophic effects. For example, consider the popular notion of the process of launching nuclear missiles from a submarine. In such a situation, access to launch missiles is deterministic, and accounts for both the risk of the wrong person gaining control, while allowing the right person to always be granted access: it simply requires two keys, held only by the two most senior officers, to be turned simultaneously on opposite sides of a room. Likewise, consider a soldier carrying a portable computing device in a hostile area. How should the device know when it is being used by the legitimate owner or has been captured? Even if a legitimate owner is incapacitated, how can the device allow the remaining soldiers retrieve their orders but not an adversary? Further, what if the submarine or the portable computing device are completely disconnected and cannot react to updates in orders, such as a ceasefire? How should they respond?

There is a natural tension between security and usefulness [SS75]. An ideal security system should be precise: it should block all forbidden actions and permit all allowed actions. This is a well-known quandary: Jones and Lipton [JL75] showed that for any non-trivial system, a precise mechanism cannot be constructed. The tension is exacerbated by the layering of defensive mechanisms: such a layering means that even if an attacker can surmount one layer, the remaining layers continue to protect. However, this cuts both ways: an inner layer cannot assume that the outer layers have done their job, and so need to be strengthened to defend the system. Strengthening means restricting actions, and so strengthening often means reducing usefulness.

For example, suppose that a defense involving physical access is breached: a laptop owner walks out of her office for a few minutes, and leaves her office unlocked. If she did not lock her screen, then the failure of physical access implicitly surmounts the protection by authentication—someone can walk in and start using the laptop. One can try to protect against this by using anomalous behavior detection, but such systems are notoriously imprecise, and the only way to improve this is to allow many false positives. This is unacceptable, as it would seriously impede a legitimate user.

The tension between security and usefulness arises in other ways, particularly when attempting to defend against insider attacks ([BEF+10, §6.5:130–135] [Pei10]). For example, consider a person who is embezzling from a company. An embezzler uses the authority the company has given him—and the permissions the system has given his account—to do illegal activities. Restricting the permissions he has to ensure that he could not embezzle would cripple the usefulness of the system. Further, some threats may not involve malicious intent at all but may simply be the result of errors or users who do not know that they present a threat [Con59, Dic77].

These difficulties arise because security systems are relatively rigid—they serve as firewalls that allow permitted actions and block forbidden actions. This is not the way firewalls work in human society. For example, security in an industrial building is provided by security guards, access control cards, security cameras, and logging of door accesses. One security policy might be that only authorized people can enter a development lab, but doors can be propped open and access cards can be "borrowed." So, it is worthwhile to have a security guard notice someone acting unusually—like carrying an expensive piece of equipment—and ask them for identification, or when a laptop is disconnected from the internet, its disconnection can be noted by examining wireless access point logs, and then looking at the history recorded by nearby security cameras. Thus, by more tightly interlocking the components of system security: authentication, authorization, and auditing, and by allowing the system to be in a somewhat uncertain state, the organization runs more smoothly and efficiently without taking on undue risk.

In this paper, we describe how optimism functions and how it addresses such scenarios. The model that we present is general. As we will discuss, we provide a structure for encompassing and combining several existing techniques, such as "break the glass" models and obligation-based systems. We also describe implementation ideas and issues of this method in practice. Finally, we give a formal mapping of our model to lattice models, such as the Bell-LaPadula model of security [BL73, BL75]. As such, future applications of optimistic access control can leverage this formalism, with certain guarantees in place.

2 Background and Related Work

The conflict between allowing users to do their jobs while preventing violations of security has long been a challenge. Traditionally, owners of resources have controlled access, with system administrators being able to override the settings when necessary. Regardless of who controls access, the principle of fail-safe defaults says to deny access unless access is required. Determining whether access is required is often non-trivial in practice. When the principle is applied, one often determines that access is needed because the access is denied, and the user must figure out whom to contact to gain access, or must breach the security mechanisms to gain access, in order to perform his job. In response, security policies are often permissive, in that they allow more access than is actually needed (violating the principle of least privilege, unavoidably).

Intrusion detection mechanisms use a variety of models to detect attacks. Anomaly detection uses statistical variations [Den87], misuse detection looks for malicious patterns [LJ88], and masquerade attacks [LJ88] seek to defeat both of these measures. Detection thresholds can be altered to flag more suspected attacks, but as a consequence, security administrators are quickly overwhelmed with false positives and/or legitimate users are mistakenly denied access.

Protection is often described as an access control matrix [Den71, GD71, Lam74, Lan81]. Abstractly, an access control matrix is a function that maps subjects and objects into rights, which enable specific actions. Each entry in the matrix consists of a set of rights. When a subject tries to perform an action on an object, the set of rights in the appropriate entry is checked. If the enabling right is present, the action proceeds; otherwise, it is blocked.

Conditional access control can augment traditional access control techniques by considering external factors such as application and environmental state and execution history. For example, Knorr uses workflows [Kno00] to control access rights. Abadi and Fournet [AF03] use code execution history, the Chinese Wall Model [BN89] uses access history, and Yuan and Tong [YT05]

use environmental elements. The family of models based on roles (RBAC) [FK92, SCFY96] use membership in a particular role at a particular time to control access rights. Attribute-Based Group Access Control (ABGAC) [BEP+08, BEP+09, BEF+10] generalizes the notion of "role" to arbitrary groups, and defines access rights in terms of membership in those groups.

A common example of conditional access control is for logins. Whether a user has the right to log in depends not only on the knowledge of the password (authorization) but also on the number of previous unsuccessful logins. If the user's past three attempts fail, then the continual access control blocks further logins, even if the user correctly enters the authentication information. The right to access the system depends upon the environment—that is, the number of previously unsuccessful logins.

But policy enforcement does not need to be binary. A variety of approaches have been suggested for "break-the-glass" access control techniques, such as in medicine [MCMD11, WJ11]. These techniques allow one to override limits on access, given certain exceptions, but are not actually adaptive techniques nor are they intended to be flexible enough to reduce access later. One can allow or disallow access, or take an intermediate step such as allowing access but logging all aspects of the session. For example, if a system is transaction based, then access improvidently granted can be reversed. This allows for an "optimistic" access control, in which access is granted initially and modified as appropriate. This technique has been applied to concurrency [CL95] and intrusion detection [FLK+98], and more recently to access control [Pov99, PE07]. However, that work is based on the Clark-Wilson Model [CW87], which does not take history into account. It also requires a system administrator to intervene manually to decide how to relax the access restrictions. This has the flavor of Karger's intelligent naming subsystem [Kar87], which reports suspicious events to the user and asks what to do. Other approaches to adaptive access control [Jas04, CRK+07, SW09] address the issue of risk, tradeoffs in access, and have some element of fuzzy logic control systems, much like our own system. However, our approach extends these adaptive systems with automated countermeasure responses.

Obligation based access control [GJD11, NBL08, PCW+11] is closely related to our work in that it considers both pre-obligations (which we view as "risk") and post-obligations (which we refer to as "countermeasures"). Our approach is more general than obligation-based access control because the preconditions may not obligate the user seeking access to do anything. In fact obligation-based access control is an example of the kind of policy mechanism could be instantiated using our approach.

Our model uses changes of state to condition access. It generalizes much of the earlier work. For example, Karger's subsystem provides a mechanism for raising concerns about a particular access, and our system allows that risk to be taken into account. In particular, if the access violates a particular conditional constraint, then the access can be blocked, or the access can be allowed but with increased logging and analysis of associated user activity. Most importantly, our model allows for a variety of flexible, non-binary, and automated countermeasures, the ability to both to increase and decrease trust, the means to account for risk and severity of consequences, and the ability to include external and/or global factors in access control rather than those on a single, isolated system.

Consider a file to which only two users have legitimate read access. The owner adds a third user to the list of authorized readers. The policy of "keep this file's contents confidential" is in some sense weakened, because now three users must keep the contents secret rather than just two.[1]

[1] One is reminded of Benjamin Franklin's comment that "Three can keep a secret if two of them are dead."

Thus, an appropriate amelioration for this weakening is to log accesses to the file; that way, if the contents leak, the users who had accessed the file can be determined.

Depending on the requirements, an analyst might care about different policies. If an analyst assumes that the kernel is secure (and that if it were not, it could be manipulated to report false data anyhow), the policies one might potentially care about are relatively simple: those that affect disk, network, and memory accesses. The objects in question are files, processes, the filesystem, the network, and the user environment.

But policy enforcement is limited to allowing or preventing access, or even logging the event. Real-time monitoring of all such events by a human is usually impractical. But a posteriori monitoring may still be possible, particularly when the unauthorized event is followed by a set of additional events (e.g., if the first unauthorized event is a user elevating their privileges, then additional unauthorized events could be using those elevated privileges). As a result, forensic logging and post mortem auditing are additional techniques to expand the limit of the types of threats that a system can cope with. Further, by allowing an attack to continue until it becomes unrecoverable, additional forensic data is captured, and the resulting analysis or other actions taken to stop the attack are ultimately more reliable.

In this paper, we present non-binary policy conditions and non-binary countermeasures, for which the correct conditions (end results) can be much more easily set than full attack graphs, and also potentially "discovered" via policy discovery. We discuss how optimistic access control can employ a well defined forensic model as a countermeasure. In fact on a network firewall, this policy is simple. The "log" function can trivially be applied to any firewall rule, whether or not a packet is allowed or denied.

3 Premise

In this section we describe the premise of our access control model. Before doing this, we first outline standard or traditional access control matrices, discuss how those have been augmented with execution history to create conditional access control. Finally, we discuss how we augment conditional access control with concepts of trust, countermeasures, and non-binary function output.

3.1 Standard Access Control

Abstractly, a traditional access control matrix is a mapping of a subjects $s \in S$ and objects $o \in O$ to a set of rights $r \in R$ expressed as a matrix A (whose entries are a). So:

$$a[s, o] = \{r\} \qquad \text{(Standard access control matrix)}$$

means that the subject s has r rights over the object o. When a subject tries to perform an action on an object, the set of rights in the appropriate entry is checked. If the right is present, the action proceeds; otherwise, it is blocked. In this form, the matrix is effectively a binary decision function, therefore, where the only possible responses are to allow or block.

As an example of the standard access control formalism, suppose a system disallows UNCLASSIFIED users from reading TOP SECRET documents. Let Larry, with username *larry* be an UNCLASSIFIED user and *secretplans* be a TOP SECRET document. Then $s = larry$, $o = secretplans$, and $r = \varnothing$. Thus, $a[s, o] = \varnothing$. Since no rights for Larry exist for the document, any action is denied.

3.2 Conditional Access Control

Now consider the effects of external information on rights. Larry may be able to read a document when he is on the premises of his employer, but not from home. This can be captured using two distinct access control matrices, one for each state (on/off premises). A simpler way to handle this is to have the entries in the *conditional access control matrix* be *functions* of rights rather than rights.

Let M be a set of information needed to determine whether a subject s may access an object o. For example, $m \in M$ may be the time of day, a security clearance, execution history, and/or the state of the application or system. Let $f : S \times O \times R \times M \rightarrow R$. Then:

$$a[s, o] = f(s, o, \{r\}, m) \qquad \text{(Conditional access control matrix)}$$

means that the value of the function f evaluated with the shown inputs will provide the set of rights that the subject s has over the object o.

A common example of conditional access control is logins. Many logins provide a blocking capability, so that after n failed login attempts the account is locked, effectively denying all future logins. In the above formulation, the s would be the user, the o would be the user's account, the r would be the login right l, and m would be the number of previous incorrect login attempts. Then f is:

$$f(s, o, \{l\}, m) = \begin{cases} \{l\} & \text{if } m < n \\ \varnothing & \text{otherwise} \end{cases} \qquad \text{(Conditional access control function)}$$

Returning to the earlier example, suppose that UNCLASSIFIED users cannot ever read TOP SECRET documents, but if the time is during business hours, then UNCLASSIFIED users can append to them. In this case, let s be UNCLASSIFIED users, let o be TOP SECRET documents, let a be the append right, and let $m = (8 < time < 17)$. Then $f(s, o, \{a\}, m) = \{a\}$ if $8 < time < 17$ and $f(s, o, m) = \varnothing$ otherwise.

3.3 Optimistic Access Control

Now consider the function f. For our purposes, we define f to be *optimistic* if the following holds:

$$f(s, o, R, m) = R_1 \Rightarrow R_1 = R$$

That is, the filtering function f does not delete any rights from the matrix. Similarly, f is *pessimistic* if:

$$f(s, o, R, m) = R_1 \Rightarrow R_1 \subset R$$

Note that a pessimistic filter function always removes some rights.

In practical terms, optimistic access control begins with a permissive system. As the system state changes, information relating to its use, its environment, and other, external information may cause a subject's access permissions to be decreased or limited in some way. The key here is the *restriction* of existing rights. Were the rights to be expanded, the form of access control might be called *pessimistic* because initially, the state of the system is such that the policy indicates that the system should avoid risking the subject could compromise the system. We focus here on optimistic access control for a variety of reasons, discussed later.

The ability to increase the monitoring of the exercise of rights leads to a second function, describing the countermeasures to be applied:

$$f_c : S \times O \times R \times M \rightarrow C \qquad \text{(Countermeasure function)}$$

where S, O, and M are defined as above. C represents a set of zero or more countermeasures. The specific countermeasures appropriate for the requested access may change over time, based upon changes to the environment (including past history of accesses). Thus, like the set of rights in the conditional access control matrix, the output is not static. For example, one can imagine a system where if risk, encompassed in M, is less than a certain threshold then countermeasure A always occurs, and if risk is greater than a certain threshold, then countermeasure A and B both occur.

Note that M may change as the countermeasures are activated. An additional countermeasure may reduce the risk of compromise, and if risk is a component of M, activating that countermeasure changes M. Similarly, it may include information tangential to the protection state but necessary for the evaluation of conditions. For example, the time of day or the specific value in the program counter may not be parts of the protection state of the system, but both are certainly relevant and may well be part of conditional expressions used in evaluating f (and, *inter alia*, f_c).

Going back again to our example, some UNCLASSIFIED documents are more sensitive than others. Suppose a company considers an encrypted file to be less useful to an intruder than a plaintext file. Thus, the system might require more trust for accessing a plaintext file than an encrypted one. Denying access to the encrypted file to certain users, however, could be too restrictive, and so a low level of trust might just require the countermeasure of logging detailed information about the access and the state of the requesting process. We call this *optimistic* [FLK+98, CL95] because rather than simply blocking access, the countermeasure allows access to continue, but other, non-interfering countermeasures record and monitor the access. Other countermeasures might be *pessimistic*: they would block any communications action (writing to a file or a socket, for example) if the trust level were too low. In our notation, let $o = TopSecretEncryptedFile$. Then if o is encrypted and a *read* right is requested, then $f(s, o, \{r\}, m) = \{r\}$ and $f_c(s, o, \{r\}, m) = \{log\}$. Thus, the action is permitted and logged. If o is not encrypted, $f(s, o, \{r\}, m) = \{r\}$ and $f_c(s, o, \{r\}, m) = \varnothing$. Thus, the action is permitted but not logged. Both of these are optimistic controls.

Now consider that, in some environment m, one is not trusted to have access to the unencrypted file. Let o_c be the unencrypted file. Then $a[s, o_c] = \{r\}$. But $f(s, o_c, \{r\}, m) = \varnothing$; in this case, f is pessimistic because $\varnothing \subset \{r\}$.

3.4 Practical Assumptions

This approach requires that the relevant data from the environment be made available to the implementation of the function.

Assumption 1 A set of agents can extract needed data from the environment with a level of assurance sufficient to use the data.

An example here is identity. An agent may obtain the identity of a process trying to access a file in order to determine the correct countermeasure. But the process may be executing on behalf of a masquerader, in which case the identity will be incorrect. Thus, the decision based on the identity also includes an assessment of the assurance of that assertion of identity: if the assertion is incredible, the identity cannot be relied upon. Other data suffers from similar issues.

Assumption 2 The selection of an appropriate countermeasure depends upon the environment as well as the subject, object, and access sought.

The types of countermeasures fall into 3 categories:

1. Altering the access control matrix to create or destroy subjects, or add or remove rights or countermeasures.

2. Altering the environment, for example by increasing or reducing trust, or increasing or reducing perceived risk.

3. Triggering another system, such as a logging or alert mechanism. For example, one countermeasure could be a rollback/recovery mechanism that is implemented using a virtual machine architecture. This would provide the kind of isolation and recovery that is commonly used now by careful users who browse suspicious websites within a separate VM. The other countermeasure could be be an enhanced forensic logging system, which uses a forensic model to determine the information that is necessary to log in order to understand the user's actions.

Appropriate countermeasures might be to elevate or diminish trust, activate an intrusion detection system, change the read/write status of a disk, launch a new round of authentication, or turn on deception mechanisms that would make it harder for an attacker to determine that the system is taking other countermeasures. Alternatively, if the concern were bulk information leakage, then a user could be allowed to view information only if they are disconnected from the network and other I/O devices (e.g., removable drives) are unmounted.

Two or more countermeasures may be triggered at different times depending on particular aspects of the environment. For example, the rollback/recovery countermeasure may be limited in the types of actions that it can recover from; opening a connection to another server is an external action that cannot be rolled back. A forensic logging system might be able to allow analysis of those situations, and so it would make sense to continue to use it as a countermeasure because rollback is ineffective.

Note that post-obligations in "user obligation" systems can be inputs to the f_c countermeasure function (and multiple obligations can be encapsulated by multiple f_c functions) in the same way that pre-obligations can be encapsulated by f.

4 Mechanisms and Policies

The policies and implementation of optimistic access controls raise interesting issues. In this section we describe the properties that we seek to achieve using our access control model, and precisely and formally define the model.

A security *policy* defines what is, and is not, allowed at a particular time. A security *mechanism* enforces it. Further, traditional security mechanisms are static, based upon the way the system is configured by the system developer and/or administrator. If the system is used in environments or for purposes they do not expect, the security mechanisms may restrict the user unnecessarily, creating the tension between usefulness and security. A mechanism that minimizes both these problems has three properties:

1. The mechanism is *dynamic*; its restrictions are not based simply on configuration, but also on the current state of the machine. This allows the protection domain for each user to change as appropriate.

2. The mechanism is *optimistic*. It allows violations of parts of the security policy that are considered less important, but not parts of the security policy that enable an attacker to compromise key goals of the system. The latter must be defined explicitly, as discussed below; the former can either be defined explicitly or determined through system use, as discussed below.

3. The mechanism is *recoverable*. This means that at some point after a violation, the system can be restored to an acceptable state. We phrase this in terms of "acceptable" rather than in terms of safety properties or rollback because the acceptable state may not be identical to an earlier state, and specific safety properties may not hold. For example, an intruder may successfully access confidential data (such as credit card information) that cannot be made confidential again, but the attacker may be prosecuted. If the system managers consider this restoring the system to an acceptable state, then for our purposes this is "recovery." Note this notion is more general than the traditional recovery of rolling the system back to a known, earlier, safe state.

We define two terms:

Definition A *core prohibition (CP)* is an action or access forbidden by the security policy, or whose violation leads to a contradiction of the security policy. Such actions or accesses must be enforced: no violations are allowed. In many cases this is enforced by traditional access control mechanisms.

Definition An *ancillary prohibition (AP)* is an action or access forbidden by configuration but not by the security policy, and so may or may not be necessary to enforce the CP. There are many ways to monitor such actions. One way could even be by using a traditional host-based intrusion detection system.

We now define the two types of APs: optimistic and pessimistic. Initially, let the set of rights in the appropriate entry (cell) of the access control matrix be α.

Definition An *optimistic ancillary prohibition* allows an action to proceed with countermeasures but in such a way that the access control matrix entry contains α–that is, the set of rights is not reduced. For example, increased audit logging could be activated.

Definition A *pessimistic ancillary prohibition* allows an action to proceed, but in such a way that there is a non-empty set of rights $\beta \subseteq \alpha$ and a possibly empty set of rights $\gamma \subseteq R$ such that the access control matrix entry contains $(\alpha - \beta) \cup \gamma$–that is, some of the original rights are deleted and some rights *may* be added. For example, such a prohibition may require that a system be disconnected from the network (here, β is the right to access the network) in order to allow a user to read a file (here, γ is the right to read the file).

Let m be the state before the access is attempted, and m' the state after the access is attempted. Let f be the set of rights before the access is attempted, and f' the set of rights after the access is attempted. Symbolically, we define the relationships as follows:

- if $f(s, o, \{r\}, m) = f'(s, o, \{r\}, m')$, then the ancillary prohibition is optimistic because rights are not removed

- if $f(s, o, \{r\}, m) \subset f'(s, o, \{r\}, m')$, then the ancillary prohibition is pessimistic because rights are removed
- if $f(s, o, \{r\}, m) \cap f'(s, o, \{r\}, m') \neq \varnothing$, then the ancillary prohibition is also pessimistic because some rights are removed. Note that some rights are also added.

If the access control function determines that a violation of an AP is recoverable, then violation is allowed but countermeasures are activated to enable the recovery at any subsequent point. This type of AP is called an *optimistic ancillary prohibition (OAP)* because the system is "optimistic" that the violation can be recovered from, if needed. If the access control function determines that a violation of an AP is non-recoverable, the system disallows the violation. These APs are *pessimistic ancillary prohibitions* (PAP) because the system is "pessimistic" that the violation can be recovered from—the function calculates that cannot be recovered, hence the "pessimism."

We note that while the concepts of, ancillary, core, optimistic, and pessimistic prohibitions are precisely defined, like any access control model, their implementation in practice is dependent on the environment and certain decisions about the environment's assets made by domain experts (e.g., system administrators).

Definition An *optimistic mechanism (OM)* is a protection mechanism that allows the user to violate an AP without having any rights removed, and possibly with a countermeasure being activated.

Definition A *pessimistic mechanism (PM)* is a protection mechanism that allows the user from violating an AP providing some rights are removed, and possibly with a countermeasure being activated.

More formally, a security policy defines a set of allowed states and a set of disallowed states. View the set of disallowed states as CPs. The configuration of the actual system's access control matrix will not be precise (that is, core prohibitions alone on a non-trivial system cannot always grant access when access should be granted, while prohibiting access when it should be prohibited), and so would prevent access to many allowed states. These states, disallowed by the system's access control settings but not by an organization's policy, form the APs. A number of different approaches may be used.

This scheme balances the notion of preventative security mechanisms with that of detection. As an example, consider a firewall that has a rule set limiting access to a system. The system CP is that a particular file can be accessed only internally. When someone tries to access the system, the firewall rules—which are prohibitions, but OAP rules because the policy speaks only to who may access a file—are changed to allow the user through. The user's actions are logged. When the user tries to breach the CP, the user is blocked, and the chain of actions in the logs checked. The key action is determined to be allowing the user access through the firewall, so now the firewall rule is set to block users in the future, thereby recovering the system. Note that the access control function may continue to regard this rule as an OAP, or it may decide to make the rule a PAP; in the latter case, when the miscreant returns, that firewall rule will not allow her through again.

Two examples will demonstrate the utility of this approach. First, suppose that someone is using a system to embezzle from a company. The subject has been properly authenticated. Moreover, the subject is making several medium-sized transfers to an otherwise inactive account, something

that should require the account being reactivated (hence, an AP). In our approach, when the first transfer is attempted, the access control function reports that the subject is trusted and so the violation of this AP should be allowed (thus, it is an OAP). The OM then activates forensic logging mechanisms and allows the user to proceed. The logs can later be used to determine what happened.

Second, suppose a file in a public area is not accessible, and prohibiting access to this file is not a CP. When a user tries to read the file, the access control function allows access and activates forensic logging. The file contains a cryptographic key that the user uses to decipher and read a second file. But access to that file is a core prohibition, and the security mechanisms block the access. At this point, the log contains the data read from the first file (the key) and a record of the attempted violation of the CP. The analyst now knows the OAP, that is the prohibition against reading the first file, should be a PAP, and adjusts the access control function accordingly.

5 Ideas in Practice

We now examine the practical considerations of optimistic access controls on policies and implementation.

One can imagine a system design where access control functions in the same way that traps to the operating system kernel do. Some functions could involve interrupting a system call in process, while others queue and wait. This could vary according to any parameter used in the matrix. To implement optimistic access control, a system would need to keep track of history, measure elements considered important to risk and trust, including both general environment/alert status.

5.1 Definitions and Policies in Practice

A key question is how to establish the OAPs and PAPs. Ideally, the security policy and system configuration would be expressed in a form that allowed automatic generation of the prohibitions. For example, a formal process model defined using Little-JIL [LMW+00], a process modeling language, or a computational workflow defined using Kepler [LAB+06], could implicitly define attempted actions veering from the process (e.g., someone doing more than what they should be doing for their job) as a prohibition [SEC+10]. Such actions could then be translated from a high-level specification to a lower-level policy using the policy reverse-engineering techniques that we discussed in Section 2, or alternatively, using translation tools such as the Propel Property Elucidator [SACO02] or by using natural language processing (NLP) techniques [Che10].

Prohibitions can also be defined probabilistically by building a corpus of past actions for each user over time by monitoring past events and using machine learning to predict likely paths and effects upon the system [Dic56]. For example, the corpus may consist of system or library calls [PBKM07a], including statistical information about which system calls occur and the order in which they occur. Over time, this corpus defines a weighted graph showing the paths that a user has historically taken, with the largest weights assigned to the most common paths. This graph can be used to monitor a user's behavior in real time and predict possible aberrations. When a user takes a path that appears rare (that is, less than a certain arbitrary threshold of frequency), the system imposes optimistic countermeasures such as limiting access (e.g., granting read-only, but not write, access) until the countermeasures are countermanded or trust is increased. This also helps both with predicting the user's future actions (which we call *anticipatory forensics*) and post

mortem forensic analysis [Pei10]. That is, if a security violation is suspected, a forensic analyst can begin with the probabilistic measures to determine the most suspicious paths and likely deviations from normal. Then he can attempt to determine how likely the deviation had a non-malicious cause such as a change in job functions or a detection threshold is set too high. Now forensics becomes a matter of determining *why* the deviant path was taken, which moves it from the realm of technology to the realm of people. The advantage is that this builds on centuries of experience and interpretation, taking into account context external to the system—something an examination of the system will not provide.

A process modeling and history-based predictive approach could be combined, too, by using a static analysis of the process model [DCCN04, CACO06] to identify and eliminate single points of failure involving catastrophic attacks entirely (via redundancy or restructuring of the process), and therefore dramatically reduce the number of PAPs. Then, for non-catastrophic attacks, the history-based approach could predict failures in real-time, monitor for them, and allow them (perhaps at limited level of access) until they become unrecoverable. Only then is a pessimistic countermeasure applied.

However, actually developing such techniques is beyond the scope of this paper. Our future work includes developing a means to derive prohibitions from policy, or for the policy to designate specific resource constraints as prohibitions, as well as the evaluation of both the security and usefulness results of the prohibitions selected, and evaluation of the initial assumptions about the effectiveness of the initial assignments of prohibitions to the three categories.

5.2 Countermeasures in Practice

Countermeasures and Increasing Trust Countermeasures can also be used to increase trust in certain cases as well. For example, if a decrease in trust reflects suspicion that the user is not who they claim to be, then successfully re-authenticating, or authenticating using a different method (e.g., biometrics), would increase trust. To demonstrate this case, define M as a function of risk $r \in R$ and trust $t \in T$, i.e., $M = R \times T$. Let $m = (high, low)$. If $read$ access is requested, then $f(s, o, \{read\}, m) = \varnothing$ and $f_c(s, o, \{read\}, m) = \{log, reauth\}$. Thus the action is denied and the countermeasures are to log and reauthenticate the user. If o is encrypted and m indicates high risk, $f(s, o, \{read\}, m) = \{read\}$ and $f_c(s, o, \{read\}, m) = \{log\}$, thus allowing the action and just logging it. If o is encrypted and m indicates medium or low risk, $f(s, o, \{read\}, m) = \varnothing$ and $f_c(s, o, \{read\}, m) = \varnothing$, thus simply denying the action, without countermeasures. Finally, if o is not encrypted, $f(s, o, \{read\}, m) = \{read\}$ and $f_c(s, o, \{read\}, m) = \varnothing$ otherwise, and so access to the file is allowed, without countermeasures.

To increase trust, suppose that the history contained in M indicates a successful re-authentication. Then, $f(s, o, \{read\}, m) = \{read\}$ and $f_c(s, o, \{read\}, m) = \{increase_trust\}$, and so trust is raised, and thus there exists a means to dynamically change future attempt at requesting access rights. Conversely, if M contains history of failure to re-authenticate then $f(s, o, \{read\}, m) = \{read\}$ and $f_c(s, o, \{read\}, m) = \{decrease_trust\}$, so access is granted, but future access may not be.

Rollback and Recovery As with all countermeasures, the checkpoint, rollback, and recovery countermeasure can be invoked when an OAP is about to be violated. This countermeasure is useful when the potential violation will be confined to the system itself and not spread outside the system. The installation of malware triggered by visiting a website that contains a steganographically

encoded exploit is such a violation. This violation may not be immediately detectable—for example, if the malware is a mail bot that remains idle for days.

An example of such a countermeasure is running a dangerous application in a virtual machine (VM). Some users run their browser this way after saving the VM's current state. If the user suspects that a page they go to installs malware, the VM and not the underlying host system is infected, and restoring the previous saved state of the VM eliminates the infection. Here, the sets of applications that can violate an OAP runs under a VM.

Forensic Logging The countermeasure of forensic logging is invoked when an OAP is violated. The system will allow the violation, but enable extensive logging (discussed below) so that all actions of the user, or involving the objects in the OAP, will be available for future analysis. The current mode of logging, unfortunately, is haphazard. Applications and operating systems log data that the developers find useful, or that they believe others will find useful. Such data tends to focus on high-level events (such as the time at which a user logged in) or flood the logs with data that might prove useful (such as capturing every system call). When a forensic analyst uses this data, she either has too little information or too much information, making analysis difficult or impossible [PBKM05]. So we impose a structure on the data [PBKM07b, Pei07] by developing a *forensic model* that describes an attack in terms of the sequence of steps that take place, and the data that is used to show that these steps took place. A forensic model can denote the set of events to be logged, aid in linking events into steps of an attack, and help bound the conditions that lead to an unusual or unexpected step in an attack.

Our approach uses optimistic, pessimistic, and core prohibitions as a basis for logging data. A key benefit of this is that analyzing the current system configuration enumerates the initial set of optimistic prohibitions, because the access control function tells us the core prohibitions. The approach builds on the existing model, but does not require pre-defining attack graphs. Indeed, by tracking the weakening of optimistic prohibitions, we can derive attack graphs.

As a specific instance, an analyst can trade off logging more information against the improvement in conducting an audit. An analyst can also strengthen the model being used when a security policy is weakened for operational purposes. For example, suppose a firewall rule is weakened to allow access to a web server (because it is seen as an OAP). We can represent the resulting weakening by augmenting the rule to enable logging while the user is allowed access through the firewall because of that rule. This will enable us to capture any attack exploiting this weakness by analyzing the resting logs.

Optimistic and pessimistic prohibitions determine *when* to log, and the forensic model determines *what* to log. The prohibitions, once weakened, form the basis of the attack graphs constructed to analyze a possible abuse of a weakened prohibition.

6 Examples

Several examples will show the utility of optimistic access control.

Example 1: Alice, Bob, Caroline and the X File

We now illustrate how the model works using several examples of how they would operate on a UNIX-like system, if implemented. For all examples, consider the following optimistic access control

matrix conditions: recall that M is a defined as set of information needed to determine whether a subject s may access an object o, where together, these define a right R. Suppose that if $M > 0.5$, access is allowed. If $M \leq 0.5$, access is denied. The interpretation of this combination of risk and the notion that the policy is pessimistic is that the system will allow any recoverable action (that is, not a "core prohibition").

Apply other countermeasures (statically, in this case) as follows: assume that for the set of countermeasures C that there exist values, representing trigger thresholds, $\delta_1 ... \delta_m \in \mathbb{R}$ where all $\delta_x \leq 1$ and $\delta_x \geq 0$. For the following examples, suppose that $m = 3$, $\delta_{log} = 0.9$, $\delta_{checkpoint} = 0.8$, and $\delta_{rollback} = 0.2$. Thus, logging and checkpointing represent optimistic mechanisms, where rollback represents a pessimistic mechanism. Further, we indicate that there is a means for raising trust, such that when trust is increased, the resulting output from the countermeasure function indicates that trust is increased by 0.2 for successful biometric re-authentication.

Suppose Alice owns the file `/usr/X`, which is readable, writable, and executable only by the owner (user). Consider the ancillary prohibition:

> *Access to file `/usr/X` is only available to Alice.*

Assume this AP is optimistic. Now, Bob wants access to `/usr/X`. The controls allow him access, but log the fact that he is getting access. For example, there may be a line in the "sudoers" file enabling Bob to execute a "cat" command as alice to the file directly. While that the administrator wants to enforce the policy, she feels that the "sudoers" configuration is necessary for usability. Here, one optimistic prohibition would allow actions to take place using sudo (rather than reconfiguring the "sudoers" file to block access), but to record them. If such an optimistic prohibition is violated, then not only is the violation itself recorded, but so are all subsequent actions that result from the violation. In this case, that would mean not just the sudo command, but the nature and targets of any actions that result from the command as well.

Scenario 1 $s =$ `alice`, $o = file($`/usr/X`$)$, *read* access is requested, and the system is such that $M = 0.4$. This may reflect the idea that the the action on the object is high risk, that the subject is not very trustworthy, or some combination of the two. Since 0.4 is less than 0.5, the *read* access would be denied. But since the issue concerns identity, suppose that the initial countermeasure is re-authentication. Thus suppose the user successfully re-authenticates using a biometric scanner. Then, because successful authentication results in an increases trust (t), which increases M by 0.2, $M = 0.6$. Now, the action is allowed since $M > 0.5$, but less than the two thresholds, so the countermeasures are taken: the event is logged and the system checkpointed. The state of m is also changed. Suppose, however, that the biometric authentication failed or timed out. Then, since $M < 0.5$, the action is denied and $C = \{log, checkpoint, rollback\}$.

Scenario 2 However, suppose a core prohibition is added as follows:

> *Access to file `/usr/X` is available only when the IP address of the machine is self-assigned (e.g., the machine is not connected to a network).*

If $s =$ `bob` and `bob` now attempts $r = read$ on $o = file($`/usr/X`$)$ where M includes a DHCP-assigned address, then access is denied). However, if the network cable were pulled out (and the IP address became self-assigned), then the core prohibition no longer inhibits `bob`'s access to the file. Now the AP comes into play, so C is based on M. If $M > 0.5$, then as usual, the event is allowed, but with countermeasures appropriate to M's actual value.

Example 2: A soldier in the field

Suppose a soldier has a handheld device containing orders and supporting information. The handheld device is constantly attempting to determine whether the device is in friendly or enemy hands. The consequence of denying access to the soldier is as grave as mistakenly giving access to the enemy. Yet neither alternative is desirable. The soldier is incapacitated in hostile territory. The device detects a new user. How can the device determine whether the new user is a surviving member of the soldier's group or enemy? One approach is for the device to give instructions to retrieve the orders in a way that would identify the user as a friend or foe.

In this case, a core prohibition might be that the machine should never be disassembled; tamperproof housing that would self destruct if opened would implement this. An ancillary prohibition would prevent the device from revealing orders unless it detects the unique electromagnetic signature given off by the owner's heart. This is pessimistic because it blocks access until a specific condition is met.

Contrariwise, an ancillary prohibition that increases logging when it detects the owner's heartbeat and the sound of gunfire is optimistic, because it continues to allow access while adding appropriate countermeasures (here, logging). Another optimistic ancillary prohibition is one that requires visual authentication of the user if it is picked up within two minutes of the failure to detect the original user's heartbeat, and that otherwise directs the user to a location where the user can be identified in person.

The core prohibition in this example is:

> *Disallow disassembly of this machine.*

The first pessimistic ancillary prohibition is:

> *Unless the signature of the owner's heart is detected, do not reveal orders.*

Note there is no associated countermeasure. The optimistic ancillary prohibition is:

> *Allow access when the owner's heartbeat and the sound of gunfire are detected.*

The associated countermeasure is to increase logging. The second pessimistic ancillary prohibition is:

> *If the user's heartbeat was not detected within the last two minutes, block access.*

Here, the associated countermeasure is to re-authenticate the user, either visually (if within two minutes of the blocking) or in person (otherwise).

Example 3: Stock market circuit breakers

In the United States, many stock exchanges contain "trading curbs" or "circuit breakers" to prevent crashes [U.S11]. These circuit breakers are metrics that monitor whether the stock market has declined by a particular percentage relative to a quarterly base threshold, and if so, takes actions to temporarily halt trading or close the market for the day. The precise actions depends on the exchange in question, the time of day, and the degree to which the market has declined.

We can express the current New York Stock Exchange (NYSE) implementation of circuit breakers [New11] using our access control formalisms as follows:

Define M as a function of time $t \in T$ and level $l \in L$ such that $M = R \times T$.

Normally, for $930 \leq t \leq 1600$, $f(s, o, r, m) = allow$ and for all other values of t, *deny*.

For a base threshold B, defined as the quarterly circuit breaker level for the Dow Jones Industrial Average (DJIA):

if $B * 0.8 < t \leq B * 0.9$ and $1000 \leq t < 1400$ then $f =$ halt for 1 hour
if $B * 0.8 < t \leq B * 0.9$ and $1400 \leq t < 1430$ then $f =$ halt for 0.5 hours
if $B * 0.7 < t \leq B * 0.8$ and $1000 \leq t < 1300$ then $f =$ halt for 2 hours
if $B * 0.7 < t \leq B * 0.8$ and $1300 \leq t < 1400$ then $f =$ halt for 1 hour
if $B * 0.7 < t \leq B * 0.8$ and $t \geq 1400$ then $f =$ market closes
if $t \leq B * 0.7$ then $f =$ market closes

To define this using our terminology, we define two states: good and bad. In the "good" state, a pessimistic ancillary prohibition—"disallow trading"—applies. In the "bad" state, a optimistic ancillary prohibition—"allow trading"—applies. The ancillary prohibitions are:

> *Pessimistic ancillary prohibition: if the state is "good" and the circuit breaker level is met, halt trading as indicated above.*

> *Optimistic ancillary prohibition: if the state is "bad" and the time period for halted trading is exceeded, resume trading.*

So the right added or removed is "trade." The pessimistic prohibition removes the right, whereas the optimistic prohibition adds it back. The countermeasures in both cases are to toggle the state: "good" to "bad" for the pessimistic, "bad" to "good" for the optimistic.

Example 4: Probabilistic Doors

At a high security institution, there is a desire to make sure that an attacker cannot "game" the system by determining the minimum credentials needed in order to gain access to a particular facility. One way to do this is by using optimism—that is, access to the facility is optimistic unless a particular security level is not currently in effect (code red, for example), in which case a core prohibition would take precedence. In other words, sometimes we give access to that facility even when credentials simply adequate, and not "perfect". In such an event, we also trigger additional countermeasures to scrutinize the activities of the person who was granted access. Note that we speak generally about access in this case: it could equally be a door or it could be access to a computer system.

We can express the using our access control formalisms as follows: define M as a function of a security threat level ranging from 1 (high threat) to 5 (low threat) correlating roughly with the security level added to a pseudorandom number of either 1 or 0. Thus, the pseudorandom number can sometimes increase the security level by incrementing the number, but will never decrease it. Define C to be a security clearance level from the set (NONE, SECRET, TOP_SECRET, SCI).

if $M \geq 5$ and $C =$ SECRET then $f =$ allow but turn on logging (e.g., security cameras)
else if $M \geq 5$ and $C \geq$ TOP_SECRET then $f =$ allow
else if $M \geq 4$ and $C =$ TOP_SECRET then $f =$ allow but turn on logging (e.g., security cameras)
else if $M \geq 4$ and $C =$ SCI then $f =$ allow
else if $M \leq 3$ and $C \geq$ SCI then $f =$ allow but turn on logging (e.g., security cameras)
else $f =$ deny.

Example 5: Role-Based Access Control

Role-based access control (RBAC) [SCFY96] bases the ability to access information on one's job functions. We now present an example of how optimistic access control can implement RBAC with additional optimistic policies.

Suppose that Ann is a registered nurse who works at the Very Big Hospital. She is assigned to the Oncology Ward and participates in the treatment of patients there. She is also an in-hospital quality control analyst, who reports to the hospital's Quality Control Committee.

On the oncology ward, Ann regularly accesses medical records of those patients she is helping to treat. As this is a small and well-defined set of patients, she should not have to access records of other patients. So, in her job as an oncological nurse, she has access to those patients' records and no others.

As an in-hospital quality control analyst, Ann needs to check the records of randomly selected patients for sampling purposes, and the records of specific patients when a lapse in quality of treatment is suspected. Hence, in that job, Ann routinely accesses a very large number of records from all different wards in the hospital. Ann is positioned to be the archetypal insider: someone with a legitimate need to access many records. In her job for the oncology ward, she needs access to records on the oncology ward, so there are no prohibitions or restrictions for her access to those records.

However, when accessing records outside of oncology ward, she is doing so in her role as as a quality control committee member. In such a case, a clear concern is the abuse of those records. So, an optimistic policy is put in place: she is expected to select no more than 5% of the set of all patient records at random. Then, if the number of records accessed is above 5%, additional logging is enacted. If she access more than 10% of records, all access removed except for access to patients in oncology ward, for which she has a legitimate and possibly urgent need to access.

7 Relationship to Other Models

This section places the optimistic model of access control in the context of other, well-studied models for the purpose of demonstrating the computability, flexibility, and versatility of the model.

7.1 Integrating Prohibitions/Policies into the Standard Access Control Model

Integrating prohibitions into the standard access control matrix model requires changing the way conditionals are handled. The change lies in the set of rights that the conditional tests against. Consider the command for subject p to give subject q the right to read file o:

command $add \bullet read \bullet file(p, q, o)$
  if own **in** $A[p, o]$
  then
    enter r **into** $A[q, o]$;
  end

Let f be the optimistic access control function that prevents both q and a third subject x from having read rights over o simultaneously. Then $f(x, o, a, m)$ becomes:

$$f(x, o, a, m) = \begin{cases} \varnothing & \text{if } q \text{ has rights over } o \\ \{r\} & \text{otherwise} \end{cases}$$

and $f(q, o, a, m)$ becomes:

$$f(q, o, a, m) = \begin{cases} \varnothing & \text{if } x \text{ has rights over } o \\ \{r\} & \text{otherwise} \end{cases}$$

and the commands are:
command $add \bullet read \bullet file(p, s, q, o)$
    if *own* **in** $A[p, o]$ **and** r **in** $f(s, o, a, m)$
    then
        enter r **into** $A[q, o]$;
    end
These commands affect only the access control matrix. Countermeasures can be triggered during the evaluation of the function f as needed. The environment m provides any ancillary information for this triggering.

7.2 Lattice Models and Optimistic Controls

Consider a lattice model such as the Bell-LaPadula model of security [BL73, BL75]. These associate a security compartment with each subject and object. Each compartment is described by a label, typically a security level and a set of categories. The notion of *tranquility* describes the nature of the association. *Strong tranquility* means that the associations are static. *Weak tranquility* means that the associations are dynamic, and the labels may change provided the requirements of the security policy are met. We focus on a lattice model with weak tranquility.

For our example, we have the following security policy drawn from the world of security classifications. A distinguished subject s has a set of objects R_Y that it must be able to read, and R_N that it must not be able to read. Thus, these are the *core prohibitions*. The rest of the objects fall into 2 classes:

1. Those objects that the policy makes no statements about ($R_?$)

2. Those objects that s may read if a sufficient business reason exists (R_B), but not otherwise

The objects in the second group start with s being unable to read them. Then, as s makes a case for seeing each object in that set, the controls are relaxed to grant s read permission. Thus the access control mechanism is pessimistic, and the *ancillary prohibition* is the restriction that s cannot read elements of R_B initially.

Let R_{BY} be the set of objects in R_B for which s has made a business case to read. Using our notation, let f be the access control function. Then $f(s, o, a,)$ becomes:

$$f(s, o, a, m) = \begin{cases} \varnothing & \text{if } o \in R_N \cup (R_B - R_{BY}) \\ \{r\} & \text{otherwise} \end{cases}$$

This explicitly prevents s from reading anything that it is not authorized to read, and allows s to read everything else.

The ancillary mechanisms extend trust to s reading some elements of R_B as s makes a business case to the controllers that reading an object in that set is necessary. Then that object is added to the set R_{BY}, which is initially empty. When necessary, objects may be withdrawn from that set, effectively revoking read permissions.

A second example involves writing. A subject s is trusted to append accurate information to certain files but not to others (perhaps because s's sources are questionable, and the second set of files requires higher integrity than s's sources are believed to have). As before, we have the sets W_Y, W_N, and $W_?$. We also have a set of files that s is initially allowed to append to, W_X. As time passes, s's trustworthiness evolves, and as it changes so do the elements of W_X.

In terms of our model, the *core prohibitions* prevent s from writing to objects in W_N. The *ancillary prohibitions* allow s to append to objects in W_X. As objects can be withdrawn from that set, this is an optimistic access control because it starts with permissions being granted and retracts them as needed. Using the notation above, the function $f(s, o, a, m)$ becomes:

$$f(s, o, a, m) = \begin{cases} \{w\} & \text{if } o \in W_Y \cup W_X) \\ \varnothing & \text{otherwise} \end{cases}$$

Thus, we have shown how our optimistic model can be mapped to lattice models, thereby demonstrating its versatility for instantiating formal models, and the guarantees that those models provide.

8 Conclusion

Optimism provides several key benefits over previous approaches, unified in a single model: a means for applying gradations of security enforcement, rather than simply binary enforcement; a means of applying security dynamically, rather than statically; and a means of providing more precise control by gathering more information while delaying strong enforcement. The essence of optimistic access control is to supply a more precise way to enforce "least privilege." Normally, systems implement least privilege with a static set of permissions that constrain the user. Optimistic access control allows the user to begin with a set of privileges reflecting the believed "least privileges" that she needs, and augment that set as necessary to perform her job. The user's use of these additions can be monitored to ensure that they are really necessary, and can also be removed when no longer needed.

A variation on optimistic access control is *pessimistic access control*. Optimistic access control allows computation to proceed until a system is unrecoverable or trust falls too low. Pessimistic access control disallows actions until otherwise notified. This particular variation is in line with the *Principle of Least Privilege* [SS75]. With pessimistic access control, the mechanism could be implemented as simply as: deny permission until that permission is explicitly requested. When permission is requested, the mechanism evaluates the request. If the request does not violate a core prohibition, then the mechanism directs the system to log (or take some other countermeasure) but grant access.

In this paper, we have provided several examples of how optimistic and pessimistic access control can aid in implementing access control in a variety of situations. We also provide a structure for encompassing and combining several existing techniques, such as "break the glass" models. We have also demonstrated formally how optimistic and pessimistic access control have decidability results closely related to those of lattice models such that future applications of optimistic access control can leverage this formalism, with certain guarantees in place.

Measuring the impact of optimistic and pessimistic access control on a computer system is difficult because of the interaction of the workload and the environment (e.g., as a UNIX kernel extension vs. a social media application). However, experiments on UNIX systems [PBKM07a]

show that even collecting and analyzing every system call is tractable in both disk space and processor overhead.

As with many systems involving interactions with humans, evaluating the effectiveness of core prohibitions based on traditional metrics of ratios of attacks prevented and attacks allowed may not be feasible in a traditional scientific sense. Evaluating the effectiveness of optimistic prohibitions based on a metric of attacks that are logged and analyzed, and result in enough information to analyze the results of the attack, requires a detailed set of real attacks that could be deployed in a double-blind manner, unknown to the system designers. That said, important metrics include the nature of prohibitions that start "optimistic" and are changed to "pessimistic," because they leave the system too vulnerable. Also important are prohibitions discovered to be necessary, even if not initially realized, and prohibitions that are discarded entirely because they are not useful.

In order to deploy these techniques in a production environment, measures of risk and trust would be needed. To that end, we are evaluating the use of social networks to inform trust, access, and knowledge, such as using Davis Social Links (DSL) [BYHW07]. We are also developing a model to generate a risk measure for each individual, producing an ordered list of threats based on group attributes. The result will be a framework to provide flexible and dynamic detection and countermeasures for enforcement based on these varying levels of risk, where levels of *access* and *knowledge*—our measures of *insiderness*—serve as proxies for the level of risk. This model is built upon our earlier Attribute-Based Group Access Control (ABGAC) model [BEP+08, BEP+09, BEF+10], which is a generalization of role-based access control that groups users based on similar attributes or access rights, rather than by roles (which often include exceptions). The new model defines groups of resources, and for each group, a set of users who have access to that group. The resource groups are ordered by "business value," thus creating a corresponding ordered list of user groups. User groups with access to the resource groups of greatest value pose the greatest threat to the organization. A longer-term goal of this work includes user studies to tune security restrictions to minimize their impact on user productivity. Also, while our approach here will be developed for a operating system, we believe our model will be applicable to other platforms that use a security model based on access control.

Acknowledgements

This research was supported in part by the Director, Office of Computational and Technology Research, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy, under contract number DE-AC02-05CH11231. It is also supported in part by the National Science Foundation and the GENI Project Office under Grant Number CNS-0940805, and by the National Science Foundation under Grant Numbers CCF-1018871, CCF-0905503, CCF-1049738, and CNS-1258577.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of any of the sponsors of this work.

The authors of this work are grateful to Schloss Dagstuhl-Leibniz Center for Informatics and in paticular to the organizers and participants of Dagstuhl Seminar #08302, "Countering Insider Threats" (`http://www.dagstuhl.de/08302`) and #10341, "Insider Threats: Strategies for Prevention, Mitigation, and Response" (`http://www.dagstuhl.de/10341`), whose insightful comments and lively discussions greatly helped advance the ideas in this paper.

References

[AF03] Martín Abadi and Cédric Fournet. Access Control Based on Execution History. In *Proceedings of the 10th Annual Network and Distributed System Security Symposium (NDSS)*, 2003.

[BEF+10] Matt Bishop, Sophie Engle, Deborah A. Frincke, Carrie Gates, Frank L. Greitzer, Sean Peisert, and Sean Whalen. A Risk Management Approach to the 'Insider Threat'. In Christian W. Probst, Jeffrey Hunker, and Matt Bishop, editors, *Insider Threats in Cybersecurity*, Advances in Information Security Series, pages 115–138. Springer, Berlin, September 2010.

[BEP+08] Matt Bishop, Sophie Engle, Sean Peisert, Sean Whalen, and Carrie Gates. We Have Met the Enemy and He is Us. In *Proceedings of the 2008 New Security Paradigms Workshop (NSPW)*, Lake Tahoe, CA, September 22–25, 2008.

[BEP+09] Matt Bishop, Sophie Engle, Sean Peisert, Sean Whalen, and Carrie Gates. Case Studies of an Insider Framework. In *Proceedings of the 42nd Hawaii International Conference on System Sciences (HICSS), Cyber Security and Information Intelligence Research Minitrack*, Waikoloa, HI, Jan. 5–8, 2009.

[BL73] David Elliott Bell and Leonard J. LaPadula. Secure Computer System: A Mathematical Model. Technical Report 2547, Volume II, MITRE, 1973.

[BL75] David Elliott Bell and Leonard J. LaPadula. Secure Computer System: Unified Exposition and Multics Interpretation. Technical Report EST-TR-75-306, Electronic Systems Division, Air Force Systems Command, Hanscom AFB, Bedford, MA, 1975.

[BN89] David F.C. Brewer and Michael J. Nash. The Chinese Wall Security Policy. In *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, pages 206–214, Oakland, CA, May 1989.

[BYHW07] Lerone Banks, Shaozhi Ye, Yue Huang, and S. Felix Wu. Davis Social Links: Integrating Social Networks with Internet Routing. In *Proceedings of the ACM SIGCOMM 2007 Workshop on Large Scale Attack Defense (LSAD)*, pages 121–128, Kyoto, Japan, 2007.

[CACO06] Bin Chen, George S. Avrunin, Lori A. Clarke, and Leon J. Osterweil. Automatic Fault Tree Derivation from Little-JIL Process Definitions. In *Proceedings of the 2006 Software Process Workshop (SPW 2006) and 2006 Process Simulation Workshop (PROSIM 2006)*, volume Vol. 3966, pages 150–158, Shanghai, China, May 2006. Springer-Verlag LNCS.

[Che10] Bin Chen. *Improving Processes Using Static Analysis Techniques*. PhD thesis, Department of Computer Science, University of Massachussetts, Amherst, Sept. 2010.

[CL95] Crispin Cowan and H. Lutfiyya. Formal Semantics for Expressing Optimism: The Meaning of HOPE. In *Proceedings of the 4th ACM Symposium on Principles of Distributed Computing*, pages 164—173, 1995.

[Con59] Richard Condon. *The Manchurian Candidate*. McGraw-Hill, 1959.

[CRK+07] P.C. Cheng, P. Rohatgi, C. Keser, P.A. Karger, G.M. Wagner, and A.S. Reninger. Fuzzy Multi-Level Security: An Experiment on Quantified Risk-Adaptive Access Control. In *IEEE Symposium on Security and Privacy*, pages 222–230, 2007.

[CW87] David D. Clark and David R. Wilson. A Comparison of Commercial and Military Security Policies. In *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, pages 184–194, Oakland, CA, April 1987.

[DCCN04] Matthew B. Dwyer, Lori A. Clarke, Jamieson M. Cobleigh, and Gleb Naumovich. Flow Analysis for Verifying Properties of Concurrent Software Systems. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 13(4):359–430, 2004.

[Den71] Peter J. Denning. Third Generation Computer Systems. *ACM Computing Surveys*, 3(4):175–216, 1971.

[Den87] Dorothy E. Denning. An Intrusion-Detection Model. *IEEE Transactions on Software Engineering*, SE-13(2):222–232, February 1987.

[Dic56] Philip K. Dick. The Minority Report. *Fantastic Universe*, 4(6):4–36, January 1956.

[Dic77] Philip K. Dick. *A Scanner Darkly*. Doubleday, 1977.

[FK92] David F. Ferraiolo and D. Richard Kuhn. Role Based Access Control. In *Proceedings of the Fifteenth National Computer Security Conference*, pages 554–563, October 1992.

[FLK+98] Leslie Franklin, Meng-Jang Lin, Ramanathan Krishnamurthy, Keith Marzullo, Chanathip Namprempre, Aleta Ricciardi, and Jeremy Sussman. Combining Optimism and Intrusion Detection. Unpublished, May 1998.

[GD71] G. Scott Graham and Peter J. Denning. Protection: Principles and Practice. In *Proceedings of the AFIPS Fall Joint Computer Conference (FJCC)*, pages 417–429. ACM, November 16-18 1971.

[GJD11] Deepak Garg, Limin Jia, and Anupam Datta. Policy Auditing over Incomplete Logs: Theory, Implementation and Applications. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, pages 151–162, 2011.

[Jas04] Jason Program Office. Horizontal Integration: Broader Access Models for Realizing Information Dominance. Technical Report JSR-04-132, MITRE Corporation, 2004.

[JL75] Anita K. Jones and Richard J. Lipton. The Enforcement of Security Policies for Computation. In *Proceedings of the Fifth Symposium on Operating System Principles (SOSP)*, pages 197–206, Nov. 1975.

[Kar87] Paul A. Karger. Limiting the Damage Potential of Discretionary Trojan Horses. In *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, pages 32–37, Oakland, CA, April 27–29, 1987.

[Kno00] Konstantin Knorr. Dynamic Access Control through Petri Net Workflows. In *Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC)*, pages 159–167, Dec. 2000.

[LAB+06] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific Workflow Management and the Kepler System. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.

[Lam74] Butler W. Lampson. Protection. *ACM Operating Systems Review*, 8(1):18–24, January 1974.

[Lan81] Carl E. Landwehr. Formal Models for Computer Security. *Computing Surveys*, 13(3), September 1981.

[LJ88] Teresa F. Lunt and R. Jagannathan. A Prototype Real-Time Intrusion-Detection Expert System (IDES). In *Proceedings of the 1988 IEEE Symposium on Security and Privacy*, pages 59–66, Oakland, CA, April 18–21, 1988.

[LMW+00] Barbara Staudt Lerner, Eric K. McCall, Alexander Wise, Aaron G. Cass, Leon J. Osterweil, and Stanley M. Sutton, Jr. Using Little-JIL to Coordinate Agents in Software Engineering. In *Proceedings of the 15th IEEE International Conference on Automated Software Engineering (ASE)*, pages 155–163, Grenoble, France, Sept. 2000.

[MCMD11] S. Marinovic, R. Craven, J. Ma, and N. Dulay. Rumpole: A Flexible Break-Glass Access Control Model. In *Proceedings of the 16th ACM Symposium on Access Control Models and Technologies*, pages 73–82. ACM, 2011.

[NBL08] Q. Ni, E. Bertino, and J. Lobo. An Obligation Model Bridging Access Control Policies and Privacy Policies. In *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 133–142. ACM, 2008.

[New11] New York Stock Exchange. NYSE Circuit Breakers. `http://usequities.nyx.com/markets/nyse-equities/circuit-breakers`, Current as of Oct. 4, 2011.

[PBKM05] Sean Peisert, Matt Bishop, Sidney Karin, and Keith Marzullo. Principles-Driven Forensic Analysis. In *Proceedings of the 2005 New Security Paradigms Workshop (NSPW)*, pages 85–93, Lake Arrowhead, CA, October 2005.

[PBKM07a] 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.

[PBKM07b] Sean Peisert, Matt Bishop, Sidney Karin, and Keith Marzullo. Toward Models for Forensic Analysis. In *Proceedings of the Second International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE)*, pages 3–15, Seattle, WA, April 2007.

[PCW+11] M. Pontual, O. Chowdhury, W. Winsborough, T. Yu, and K. Irwin. On the Management of User Obligations. In *Proceedings of the 16th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 175–184, 2011.

[PE07] Keshnee Padayachee and J. H. P. Eloff. Enhancing Optimistic Access Controls with Usage Control. In Costas Lambrinoudakis, Günther Pernul, and A Min Tjoa, editors, *Trust, Privacy and Security in Digital Business*, number 4657 in Lecture Notes in Computer Science. Springer Berlin/Heidelberg, 2007.

[Pei07] 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.

[Pei10] Sean Peisert. Optimistic Access Control and Anticipatory Forensic Analysis of Insider Threats. In M. Bishop, L. Coles-Kemp, D. Gollmann, J. Hunker, and C. W. Probst, editors, *Insider Threats: Strategies for Prevention, Mitigation, and Response*, number 10341 in Dagstuhl Seminar Proceedings. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2010.

[Pov99] Dean Povey. Optimistic Security: A New Access Control Paradigm. In *Proceedings of the 1999 Workshop on New Security Paradigms (NSPW)*, pages 40–45, Caledon Hills, Ontario, Canada, 1999.

[SACO02] Rachel L. Smith, George S. Avrunin, Lori A. Clarke, and Leon J. Osterweil. PROPEL: An Approach Supporting Property Elucidation. In *Proceedings of the 24th International Conference on Software Engineering (ICSE)*, pages 11–21, Orlando, FL, May 2002.

[SCFY96] Ravi Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2):38–47, Feb 1996.

[SEC+10] Borislava I. Simidchieva, Sophie J. Engle, Michael Clifford, Alicia Clay Jones, Sean Peisert, Matt Bishop, Lori A. Clarke, and Leon J. Osterweil. Modeling Faults to Improve Election Process Robustness. In *Proceedings of the 2010 Electronic Voting Technology Workshop/Workshop on Trustworthy Computing (EVT/WOTE '10)*, Washington, D.C., August 11–13, 2010. USENIX Association.

[SS75] Jerome H. Saltzer and Michael D. Schroeder. The Protection of Information in Computer Systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[SW09] Gunnar Stevens and Volker Wulf. Computer-Supported Access Control. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 16(3):12, 2009.

[U.S11] U.S. Securities and Exchange Commission. Circuit Breakers and Other Market Volatility Procedures. `http://www.sec.gov/answers/circuit.htm`, Current as of Oct. 4, 2011.

[WJ11] Q. Wang and H. Jin. Quantified Risk-Adaptive Access Control for Patient Privacy Protection in Health Information Systems. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 406–410, 2011.

[YT05] Eric Yuan and Jin Tong. Attributed Based Access Control (ABAC) for Web Services. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*, 2005.