Working Set-Based Access Control for Network File Systems

Stephen Smaldone, Vinod Ganapathy, and Liviu Iftode
Department of Computer Science
Rutgers, the State University of New Jersey

ABSTRACT

Securing access to files is an important and growing concern in corporate environments. Employees are increasingly accessing files from untrusted devices, including personal home computers and mobile devices, such as smart phones, which are not under the control of the corporation, and may be infected with viruses, worms, and other malware. In such cases, it is crucial to protect the confidentiality and integrity of corporate data from malicious accesses.

This paper proposes a novel scheme called Working Set-Based Access Control (WSBAC) to restrict network file system accesses from untrusted devices. The key idea is to continuously observe and extract working sets for users when they access files from trusted devices and use the working sets to restrict user file accesses from untrusted devices. This paper reports on the design and implementation of tools to automatically extract working sets, and transparently enforce WSBAC without requiring changes to the file system. Our experiments with realistic network file system traces lead us to conclude that WSBAC offers a flexible yet secure way to restrict access from untrusted devices, and that the runtime overheads of WSBAC enforcement are negligible.

Categories and Subject Descriptors

D.4.6 [**Security and Protection**]: Access controls; D.4.3 [**File Systems Management**]: Distributed file systems

General Terms

Design, Experimentation, Security

Keywords

access control, working set, network file system

1. INTRODUCTION

This paper concerns the problem of securing access to files on corporate Intranets. Employees are increasingly beginning to access such files from a variety of devices, including personal computers as well as mobile devices, such as smart phones (e.g., utilizing web-based file access). File access is typically secured using standard network file systems authentication mechanisms, such as VPNs and firewalls. However, a user may choose to access files from a device whose software stack is not under the control of the corporation. Such an untrusted device may contain malware, for example viruses and worms, that compromise the confidentiality and integrity of corporate files when they are accessed via these devices. For example, a worm on an employee's mobile phone may delete all her files when she accesses the corporate Intranet.

Prior work on securing access to resources on corporate Intranets has focused on verifying the software stack on employees' devices. Sailer *et al.* [26] present an attestation-based approach that uses Trusted Platform Module (TPM) hardware to acquire integrity measurements of the software running on an employee's device. User connections are allowed only from devices that run software configurations that have been approved by the corporate network. While this approach limits user connection origination from valid trusted devices, it assumes the existence of TPM hardware on those devices. Neither legacy devices nor most of today's handheld devices have such hardware installed. This necessitates an all-or-nothing approach to allow access from such devices—either prevent them from connecting to the corporate network or allow them to connect at the risk of exposing the file system to malicious accesses from these devices.

This paper proposes a novel approach, called *Working Set-Based Access Control* (WSBAC), which restricts file access from untrusted devices *without* requiring special hardware and in a manner that is fully compatible with legacy file systems. Our approach augments access control mechanisms implemented on network file systems with the notion of working sets. The key idea is to prevent accesses to files outside an employee's working set when this access happens from an untrusted device. [1] When an employee accesses files from a trusted device (e.g., on the corporate network), she is allowed free access to all her files, and is limited only by the native access control policy enforced by the network file system. Simultaneously, an agent on the corporate network observes her file access patterns and extracts her working set. This working set is used to construct an access control policy that is enforced upon access from an untrusted device.

Using the employee's working set to regulate file accesses

SACMAT'09, June 3–5, 2009, Stresa, Italy.

[1] For this paper, we will assume that devices administered by the corporation are trusted, and that employee's personal devices are untrusted. However, WSBAC is independent of this assumption and will work with any technique that can be used to differentiate between trusted and untrusted devices, e.g., devices equipped with TPM hardware and integrity measurement tools [26] could be considered trusted.

ensures that access control is neither overly restrictive, nor overly permissive. Intuitively, most file accesses are governed by the working set; indeed prior work has shown temporal patterns in user file accesses [30]. Because an employee will tend to access a file that she has recently accessed, using the working set does not overly restrict file accesses. In contrast, malware accesses to the file system, such as those by viruses and worms, typically exhibit no such patterns. Using the working set to guard accesses from untrusted devices ensures that files outside of the employee's working set are protected from such accesses by malware. Damage caused by malware is thus restricted to the employee's working set. Because the working set only contains files that have most recently been modified by the employee (e.g., during the course of a day), WSBAC can be coupled with version-control systems to recover quickly from the damage caused by malware.

Implementing WSBAC requires the design and implementation of two key agents, one that extracts the working set and formulates a file access policy (Polex), and one that enforces this policy (Polen). We have implemented both Polex and Polen for the network file system (NFS) protocol. Polex automatically extracts a user's network file system working set by observing that user's network file system accesses. Through this extraction, Polex generates per-user working set summaries, which are subsequently utilized by Polen. Polen is the WSBAC enforcement agent that interposes on the network file system client-server path and intercepts all messages passed between them. To perform WSBAC policy enforcement, Polen extracts, inspects, and modifies network file system message attributes. Finally, Polen provides speculation mechanisms to allow file creations and writes from untrusted devices to occur in the case of imprecise working set estimation. Speculations are reconciled and committed to the file server by the user (or user's delegate) from a trusted device. To be compatible with legacy file systems, we have implemented both Polex and Polen as network middleboxes utilizing our existing FileWall framework [28, 6]. However, WSBAC can also be implemented without a network middlebox by suitably modifying the file system.

This paper makes the following, novel, contributions:

- **Working Set-Based Access Control (WSBAC).** We propose and evaluate WSBAC, an access control technique that estimates per-user working sets to formulate an access control policy that is enforced during untrusted accesses.
- **Prototype implementation of WSBAC.** We present an implementation of WSBAC in the context of the Network File System (NFS). We have implemented Polex, an agent that continuously observes user file access patterns and formulates a working set-based access policy, and Polen, an agent that enforces this policy using a network middlebox.
- **Evaluation on network file system traces.** We present an empirical evaluation of Polex and Polen using real-world network file system traces. Our evaluation suggests that WSBAC is highly effective; in particular, the WSBAC policies extracted by Polex estimate file access behavior to within an error rate of 0.92%.

The remainder of this paper is organized as follows. Section 2 presents a motivating example for our work. We address several common concerns on the applicability of WSBAC in Section 3. Sections 4, 5, and 6 present the WSBAC system design, prototype implementation, and evaluation. Finally, Section 7 reviews related work and Section 8 concludes the paper.

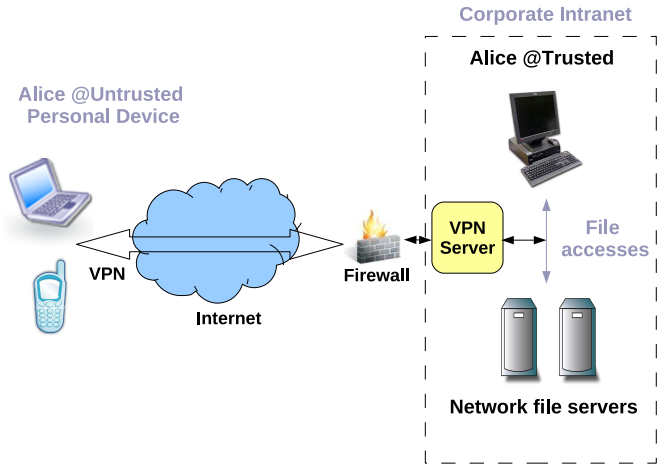

Figure 1: Example scenario showing access to corporate files from untrusted personal devices.

2. EXAMPLE SCENARIO

To motivate WSBAC, consider the file accesses made by a typical employee, Alice, of a corporation that manages files using a network file system (Figure 1). At work, Alice may access several files during the course of a day using her desktop PC, which the corporation trusts. For instance, she may develop source code using an IDE, look up or modify documentation using an editor, or use a spreadsheet application. Each of these applications involves several accesses, both reads and writes, to files stored on the network file server. She may also wish to access these files from a personal computer that her employer does not trust, e.g., from home or during travel. Such accesses typically happen via a VPN connection to the corporate Intranet, and may either be from a personal laptop, a PC on the network of a business partner or client of her employer, or, increasingly, via mobile devices such as smart phones (e.g., web-based file access).

To ensure safe yet easy access to files in such scenarios, two conflicting requirements must be met. First, Alice must be allowed access to read/modify her files, being constrained only by the access control policy of the network file system. This requirement is necessary to ensure seamless access to files both within and outside the corporate network. Second, "insecure" accesses to Alice's files must be denied. This requirement is necessary to prevent accesses that may potentially be performed by malicious software on Alice's personal computer or mobile device using her credentials. These requirements conflict because of the limited power of existing network file system administration tools.

Network file system administration tools offer an all-or-nothing choice to enforce secure accesses to files on a corporate Intranet. First, existing tools do not offer any fine-grained method to determine the file accesses performed by Alice, either from within the corporate network, or from her personal computing device. Consequently, these tools can neither observe Alice's file system access patterns nor enforce policies that disallow anomalous file accesses. Second, network file systems do not store the network context of a user's accesses with the file system context. This makes it difficult to differentiate Alice's file system accesses from

different devices, such as from her work PC within the corporate network or from her personal laptop at home.

WSBAC addresses the above shortcomings by extracting the working set of Alice's file accesses from trusted devices and enforcing access control based upon her working set when she accesses files from untrusted devices. WSBAC's POLEX agent continuously approximates Alice's working set by observing her file access patterns when she is connected from a device that her corporation trusts. This working set is used as the basis for enforcing access control when Alice connects from her smart phone or personal laptop. WSBAC's POLEN agent enforces policy, in this case as a network middlebox, thereby ensuring transparent enforcement, even with legacy network file systems.

The ability of WSBAC to *continuously* and *automatically* adapt to changes in Alice's working set is its central, novel, feature. This feature ensures that untrusted file accesses are not governed by a static access control policy that is restrictive, hard to formulate, and hard to maintain. Working sets offer a flexible yet secure abstraction to restrict untrusted file accesses. However, enforcing access control by denying all accesses outside the working set may be too restrictive in certain scenarios. For example, the IDE or spreadsheet application that Alice uses to access her files may create temporary files, such as locks, that may not be in her working set. Alice may be unable to usefully access her files from her personal device if the creation of such files is disallowed. WSBAC handles such cases by allowing for *speculative* file accesses during policy enforcement. Speculative file accesses allow the IDE or spreadsheet application to create and modify temporary files, thereby permitting Alice useful access to her files.

3. APPLICABILITY OF WSBAC

With the background above, we now address some common concerns on the applicability of WSBAC.

Suppose that a user, Alice, accesses files from an untrusted device. How can she access files that are not in the working set extracted by Polex? We consider separately the case of reads and writes. Writes to files that are not in the working set, including the creation of new files, are handled speculatively by POLEN; changes made to such files are visible only to Alice. When Alice accesses these files again from a trusted device, POLEN commits the speculative accesses after they have been verified by Alice.

To handle reads to files that are not in the working set, WSBAC supports the inclusion of a reliable secondary authentication mechanism [1, 10, 14, 24] for Alice to add files to her working set. The inclusion of this type of mechanism would ensure that malware on untrusted devices cannot automatically add files to her working set. For example, Alice could call a help desk to add files to her working set using a secret PIN number during the call to prove her identity to the help desk. Afterward, Alice can access the newly added files in her working set, thereby allowing WSBAC to be practical even in scenarios where Alice does not have access to a trusted device for extended periods of time, e.g., during travel.

Can Alice share speculative updates to files with other users? As described earlier, WSBAC handles writes to files that are not in Alice's working set using speculation; these updates are normally only visible to Alice until she commits them. However, there may be cases where these updates must be visible to other users. For example, suppose that Alice is collaborating on a paper with others.

WSBAC offers Alice two choices to ensure that an update to the paper made from an untrusted device is visible to her co-authors. Alice can include the file in her working set using a reliable secondary authentication mechanism; because POLEN does not restrict accesses to files in the working set, updates to the paper will be visible to the entire group. However, this approach has the disadvantage of committing possibly malicious updates to the files, e.g., by malware on Alice's untrusted device, to the file system.

Alice can avoid this problem by instead using WSBAC's mechanism to *share speculative file updates*. Using this mechanism, Alice can share speculative updates to selected files with her co-authors. However, this mechanism also ensures that *all subsequent updates* to the file (including those made by her co-authors) will be speculative. Changes are committed only when one of the group members (not necessarily Alice) working on a trusted device verifies the file updates. In this way, by sharing speculative updates with her co-authors, Alice is also delegating commit permission to them, should they be working from trusted devices.

Can Alice defeat WSBAC protection by artificially inflating the size of her working set? Yes, this is possible. For example, Alice could write a script that runs overnight from a trusted terminal and accesses all her files, thereby forcing POLEX to include all her files into her working set. Alternately, Alice could add all her files to her working set via the Web interface.

However, it is to *Alice's advantage* to use WSBAC. Much as Alice can disable her virus scanner at the risk of being infected, she can disable WSBAC protection at the risk of exposing the files in her working set to malware. Even in this case, damage is limited to Alice's files, because WSBAC augments traditional network file system access control.

Still, there are several procedural regulations that Alice's employer could use to prevent her from artificially inflating her working set. For example, POLEN could record her accesses from an untrusted device and compare them against the POLEX working set. If these accesses differ significantly, the employer could use this as an indication that Alice is artificially inflating her working set, and issue her a warning.

One special case of this could be caused directly by a valid process that accesses files while acting on Alice's behalf, for example, a client-based virus scanner that scans Alice's mounted network file systems.[2] This problem can be solved in two ways. One way is to take advantage of a distributed Mandatory Access Control system [16] such that any network file system messages issued by the virus scanner can be distinguished by their accompanying labels and ignored by POLEX. For legacy clients, the virus scanner process can run in the context of a special user, and then all network file system messages issued by that user ID can be safely ignored.

Does WSBAC compromise Alice's privacy? Both POLEX and POLEN continuously monitor Alice's accesses. While Alice's employer could potentially use these tools to compromise her privacy, such compromise is possible even otherwise. The network file server, accesses to which WSBAC protects, is administered by Alice's employer, who therefore can observe all her accesses even without POLEX and POLEN. Several reports suggest that corporations perform such monitoring on their employee's work habits [2, 22, 27] and are unlikely to change this practice. We argue that the

[2]We can safely ignore the effects of a server-based virus scanner since they operate locally at the file server and do not cause network file systems messages to be issued.

use of POLEX and POLEN at least makes employees aware of such monitoring by their employers. WSBAC also requires that encrypted file accesses originating from untrusted devices be decrypted at POLEN (rather than at the network file server). However, because both POLEN and the network file server are administered by Alice's employer, decrypting file accesses at POLEN compromises her privacy no more than decrypting them at the file server.

In spite of these restrictions, Alice can protect her privacy by using a scheme that only encrypts the *contents* of packets (and not their *headers*, which contain information that is used by POLEX and POLEN).

Why is virus scanning at the network file server not enough? Although is it common for virus scanning to occur at network file servers, this technique is insufficient to protect a user's data. Virus scanning at the server can intercept file writes and inspect the modifications to search for virus signatures, should a virus attempt to propagate itself in the content of the writes. Virus scanning cannot completely protect the integrity of a user's data since it cannot detect malicious deletes made by a virus from an untrusted device. Furthermore, it does not protect the confidentiality of user data, since any virus masquerading as a legitimate user would have complete access to all of that user's data.

4. DESIGN

In this section, we provide a detailed overview of the design and architecture of our WSBAC system.

4.1 Working Set-Based Access Control

We define a user's file access working set (WS) to be the set of files (and directories) the user has accessed over some recent period of time (typically 1 day). Within this set, files belong to subsets that define the access permissions for that file. For example, all the WS files for which a user has read permission are included in the read permission subset of the WS. These subsets are not necessarily disjoint, as a file may be included in multiple permission subsets (e.g., a file for which a user has both read and write permissions). Therefore, a file included in any subset of the user's WS implies that the user possesses the permission defined by that subset for that file.

Since working sets are specific to each user and typical network file systems scale to serve several hundred users, it is impractical for an administrator to manually define the WS for each user of the system. Alternately, allowing users to self-define their WSs defeats the primary purpose of the WSBAC system. Users will likely over-estimate their WSs for fear of lacking access to files they may need, even if there is a low probability that they will actually need to access the files while working from an untrusted device. To address these concerns, we approximate the per-user WSs automatically. This removes the potential burden on administrators, while reducing the ability of a user to over-estimate their WS. WS extraction is performed by the POLEX agent of our system.

Automatic extraction of user WSs may lead to inaccuracies because of two possibilities. First, the WS may include files that the user will not need to access from an untrusted device. Second, required files may be excluded from a user's WS. In either case, users will never gain access to a file that they do not already have access to since WSBAC only augments the standard network file system access policy, and cannot make a user's access rights more permissive. In the limit, a user's WS may only grow to match the set of all files that she already has permission to access from a file server.

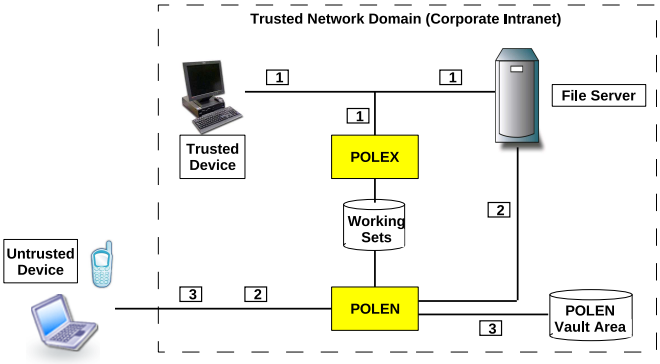

Figure 2: WSBAC Overview. Requests from trusted (1) and untrusted (2 and 3) devices are shown. Request 3 is handled *speculatively*.

In general, there are two read cases and two write cases that must be handled by the WSBAC system. File and directory reads may be attempted for files either included in or excluded from a user's WS. For these cases, WSBAC allows reads only for those files included in a user's WS and denies all other read attempts (although, a user is able to add files to her WS using a reliable secondary authentication mechanism). Note that this covers both data and meta-data reads. File and directory writes include both data and meta-data writes (create, remove, rename fall into this category). Writes to files and directories included in a user's WS are performed normally, while writes to files and directories not included in a user's WS are performed *speculatively* (described in Sections 4.4 and 5.2.2). WSBAC enforcement is performed by the POLEN agent of our system.

4.2 WSBAC Policy Extraction (POLEX)

WSBAC working set extraction and policy formulation is performed by the POLEX agent. POLEX observes a user's network file system accesses when they are performed from a trusted device. These accesses, which travel between network file system clients and servers in messages, are captured by POLEX and inspected. POLEX utilizes the file system attributes contained in these messages to construct and maintain per-user working set summaries.

Figure 2 illustrates how this occurs. In the figure, a trusted device performs a network file system access, shown in the figure as a network message labeled 1. This message travels through the network and ultimately reaches the network file server where the file access is performed on the file system. Along the way, the message is captured by a network element and a copy is sent to POLEX for processing. Any network element, such as a network switch, could perform this message capture and copy forwarding function. To handle the case of encrypted or signed network traffic, POLEX (and POLEN) shares the encryption keys with the network file server.

POLEX maintains the per-user WS summaries on stable storage, as they are built and updated. These summaries are built through the use of compact summary data structures (Bloom filters, in our implementation) and they are exported by POLEX for use in POLEN. POLEX also utilizes the WS summaries to provide per-user virtual file system namespaces. These virtual namespaces provide administrators a mechanism to view and modify the WSBAC permissions for individual users as approximated by POLEX. Further discussion of POLEX virtual namespaces is in Section 5.1.

4.3 WSBAC Policy Enforcement (POLEN)

WSBAC enforcement is performed by the POLEN agent. POLEN resides on the network, interposed between the network file system clients and servers, and intercepts all messages passed between them. POLEN utilizes the file system attributes contained within the messages and the per-user WS summaries to enforce WSBAC for those users accessing the network file system from an untrusted device.

Figure 2 shows POLEN interposed between network file system clients and a file server. Messages from untrusted devices, such as message 2 in the figure, are evaluated against the WSBAC policy. For file accesses that are allowed, POLEN forwards them to the file server (as is the case for message 2 in the figure), otherwise, POLEN responds directly with a "permission denied" message.

4.4 POLEN Speculation

Writes from an untrusted device to files not included in a user's WS are allowed by POLEN, but are not committed to the network file system. Instead, POLEN holds these writes aside by logging them in a *vault area*. This area is a stable storage location where speculative writes can be sequestered from the network file system. Access to the vault is reserved for POLEN only. This partitioning of speculative writes from the network file system provides safety for these writes, while denying visibility of the speculative accesses for all other users. Speculative writes are visible to the users who issue them. All user accesses flow through POLEN, which exports per-user virtual namespaces to the users. These namespaces are the union of the real file system merged with the speculative operations performed by the user.

POLEN allows speculative writes to data and meta-data to be performed by a user from an untrusted device for two reasons. First, the user may wish to create new or temporary files. If POLEN limits writes to files within the user's WS, it will not permit this. Second, although POLEX may have observed reads to a file and included the file in the WS for reads, POLEX may not have observed any writes by that user to that file. Since it is possible the user may also have write permission, we allow these writes to occur speculatively.

Figure 2 shows a speculative access as it traverses the network from an untrusted device to a network file server (shown as message 3). When the message is intercepted by POLEN, it is logged and stored in the POLEN vault area, which may reside on any stable storage accessible to POLEN.

Speculative writes may pose a problem in the cases of write-after-write and read-after-write sharing between users. This problem is mitigated by two factors, though. First, for typical deployments of network file systems, both types of sharing have been found to be very low (between 0.9% and 0.6% of all file systems operations as reported in [18, 31, 19]). Second, network file systems, such as NFS [7], in the absence of locking, do not provide strong file consistency between users. They typically provide close-to-open consistency where update propagation is only guaranteed at the time of file closure. In spite of this, there are likely to be cases where update delays due to speculation pose a problem between users sharing files. To handle this, we extend sharing to speculative accesses through our *speculation sharing and delegation mechanism*. We expect (and in fact show in our evaluation) write speculation to be the exception and not the common case in practice. Further description of this mechanism is in Section 5.2.2.

Reconciliation of speculative accesses occurs when a user returns to a trusted device and resumes interacting with the network file system. At this point, POLEN starts reconciliation for speculative changes stored for the user in the vault area. POLEN allows speculative creates and writes to temporary files to be automatically committed to the server once reconciliation begins, since they will not potentially destroy or modify any existing data. We assume that all network file servers perform virus scanning for all file writes before committing them, as this is common practice. In the event that virus scanning is not performed at the server, automatic reconciliation can be completely disabled.

All other speculative updates to existing files must be manually verified by the user, before POLEN will proceed to commit them. This may occur in a number of ways, including a web-based interface or an automated email service. Once verified, speculative updates are presented to the network file server as if issued by the user, and reconciled by the system in a manner similar to the CODA File System [19].

5. IMPLEMENTATION

We have implemented both POLEX and POLEN by extending FileWall [28, 6], a network file system middlebox that we developed in prior work. Using a middlebox permits WSBAC enforcement without modifying the file system. However, WSBAC can also be implemented without a network middlebox by suitably modifying the file system.

5.1 Implementation of POLEX

In this section, we describe the implementation of our policy extraction agent. We have two primary requirements in designing this system. First, it must run online in order to continuously and automatically adapt to changes in users' network file system working sets. Second, it should be non-intrusive to both clients and servers. By requiring no server modifications, POLEX can be easily deployed without impact (in terms of software maintenance, or performance effects) to the critical resource being monitored. We further restrict the system to not require any software modifications to the clients. For this system to be most useful, it must be deployable in any scenario, including situations that involve legacy deployments and untrusted devices.

POLEX implements a framework to extract approximations of user network file system working sets, which form the basis of WSBAC. We built POLEX as a network agent that receives and processes copies of network file system messages to examine message attributes and extract per-user working sets. Figure 2 shows how POLEX is deployed for a typical network file system infrastructure, where copies of network file system messages are forwarded to POLEX by a cooperating network element (e.g., network switch port mirroring, POLEN, etc.) The remainder of this section describes the working set extraction mechanism, the administration interface provided by POLEX, and WSBAC state maintenance.

5.1.1 Working Set Extraction

As shown in Figure 2, POLEX extracts per-user working sets by observing the network file system messages that flow between trusted devices and file servers. This is accomplished by using semantic knowledge about the network file system protocol. Since POLEX can observe the servers' responses to the devices' requests, it can discover, over time, the file permissions users have to various portions of the file system (files and directories). Once discovered, these permissions are stored in a set of per-user compact summary data structures (Bloom Filters). POLEX creates and maintains six Bloom Filters per user, three for file permissions

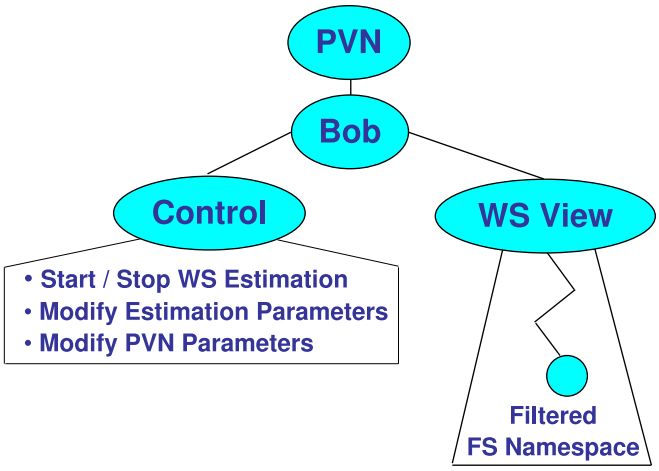

Figure 3: Subset of the Policy View Namespace (PVN). For each user (e.g., Bob), there are *control* and *WS view* namespaces.

(read, write, and execute) and three for directory permissions (read, write, and execute). The Bloom filters comprise the WSBAC user policy summaries and are stored on local persistent storage. Once generated, they are held by POLEX for later use in POLEN.

5.1.2 Policy View Namespace (PVN)

POLEX defines a virtual namespace, called the *Policy View Namespace* (PVN), for WSBAC working set summary administration. It is an interface that provides views of users' effective network file system access control policies. Through the PVN, administrators can interact with POLEX over the network file system interface using a familiar set of tools. In fact, one of the main purposes of using the standard file system protocol as the policy view interface was to enable building tools that could easily take advantage of this well-known interface. Access to the PVN is restricted to administrators. In a secure deployment, such access would occur over a secured private network, reserved for this purpose.

The functionality of the PVN is similar to that of the Linux /proc file system. View Handlers are invoked at POLEX on receiving file system requests for virtual objects. For read-only requests, for example, read, readdir, getattr, etc., the handlers query the PVN state and generate the file contents dynamically. Write operations update the PVN state and are used to modify the POLEX and PVN configuration parameters, or manually modify per-user working set summaries. Figure 3 shows an example PVN for user Bob. There is a similar pair of namespaces for every other user in the system. The figure shows the two primary components to Bob's PVN: the *control* namespace and the *WS* namespace.

Through the PVN control namespace, administrators can tailor view configurations to meet their needs. Additionally, they can modify WS estimation parameters and start/stop the WS estimation process. These tasks can be performed for individual users (e.g., Bob in the figure) or globally for all users. The PVN WS namespace provides a view of the real exported network file system namespace filtered by the user WS summary permissions. The names for objects in the WS namespace are derived directly from the files they represent in the observed network file system, but only the files and directories that exist in a user's estimated working set are visible in the WS namespace. The WS namespace provides an interface for administrators to query and modify the effective permission assignments for each of a user's files

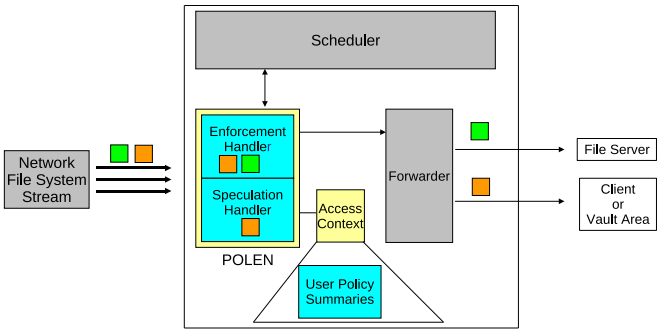

Figure 4: Primary components of Polen. Included are the Enforcement Handler, Speculation Handler, User Policy Summaries (access context), and Vault Area.

and to manually add and remove files from a user's estimated working set.

5.1.3 State Maintenance

POLEX maintains two different forms of persistent state. First, it maintains the system configuration based on any parameter/control changes issued through the PVN. Second, POLEX stores the WSBAC per-user working set summaries extracted from the network file system messages it observes. Keeping the summaries as persistent state is more for convenience rather than a strict requirement. Since this state is built through observation of network file system messages, it can be continuously regenerated over time. The primary penalty, in the case of lost state, is the time that it takes to regenerate that state.

5.2 Implementation of POLEN

In this section, we describe the implementation of the policy enforcement agent. POLEN includes mechanisms for WSBAC policy enforcement, write speculation, speculation sharing and delegation, and update reconciliation.

5.2.1 WSBAC Policy Enforcement

POLEN operates on network file system messages as they are exchanged between clients and servers, as shown in Figure 4. As messages are captured, they are handled by POLEN for WSBAC evaluation and enforcement. POLEN only operates on messages from untrusted devices (see Figure 2).

Messages from untrusted devices are passed to the *Enforcement Handler* (see Figure 4). This handler extracts network file system attributes from the message to determine the file system operation (fop), file identifier ($fileid$), user identifier (uid), and request identifier ($reqid$). The handler then retrieves the user's working set policy summary from a local persistent state store using the user's uid, and checks if the $fileid$ is included in the WS of this user with the necessary permission to perform the requested operation fop. For reads, the network file system message will either be forwarded to the file server (as in the figure) or POLEN will generate a "permission denied" response message and forward it to the client. For writes, the message will either be forwarded to the file server or will be speculatively allowed.

5.2.2 Write Speculation

Whenever POLEN encounters a write operation to a file from an untrusted device that is not explicitly allowed based on the WSBAC evaluation, POLEN handles it speculatively. File system updates are stored within a predefined *vault area*, located on stable storage. POLEN includes an inter-

face for a user to view her speculative updates from the untrusted device that performed them. This interface can be accessed via the standard network file system protocol or via a web-based mechanism, and is implemented by the *Speculation Handler* as shown in Figure 4.

5.2.3 Reconciliation

Reconciliation must occur when a user has any outstanding speculative updates stored for her by POLEN in the vault area. During reconciliation, speculative updates must first be verified by the user who issued them (or by a user's delegate), before they will be committed to the server. Such verification guarantees that any speculative updates submitted by malware will be identified by the user prior to commit.

Users manage their speculative updates through the same interface (network file system or web-based mechanisms) provided by POLEN for write speculation. Through this interface, a user (or administrator) can manually inspect her speculative updates currently stored in the vault area, approve the updates to be committed to the file server, and handle any exceptional conditions that arise. The granularity of the update is per file, and a user can view the individual changes prior to approving or denying them.

5.2.4 Speculation Sharing and Delegation

As previously mentioned in Section 4.4, to support the existing sharing model common to modern network file systems, we provide mechanisms to allow speculative updates to be shared between users. The first mechanism has been discussed in the Section 5.2.3. A user who has performed a speculative update can directly commit those updates through the POLEN-provided interface. Once committed, they are available to all other users immediately.

For more tightly coupled sharing situations, though, it might be tedious for each untrusted user to manually commit changes to a small set of files. For this case, we allow a user to share a portion of their speculative updates with other untrusted devices. Using the POLEN-provided interface, a user can directly specify, per file, other untrusted devices that should have immediate visibility to her speculative updates. Under this sharing model, users experience traditional close-to-open network file system semantics and are not burdened with manually committing updates. In this case, the users' updates remain speculative and WSBAC protection holds.

For trusted devices to access speculative changes from untrusted devices, we provide a delegation mechanism. The owner of the speculative changes can allow a user of a trusted device to commit the updates to the shared files (through the POLEN user interface). This gives a user the ability to commit updates as if she was the update owner. Finally, for frequent sharing between trusted and untrusted devices, we allow a trusted device to connect to the file server through the same path as an untrusted device, to gain visibility to the speculative update similar to an untrusted device.[3]

6. EVALUATION

In this section, we present the evaluation of the WSBAC system. The goals of the evaluation are as follows. First, we measure the processing time and storage costs to perform WS extraction with POLEX. Second, we measure the accuracy of per-user working set extraction. Third, we perform sensitivity analysis on working-set extraction accuracy. Fourth, we quantify the amount of speculation that occurs during POLEN enforcement. Finally, we measure the overheads imposed by the POLEN enforcement mechanism.

[3]For the purposes of this paper, we assume any trusted device that requires such access can either connect through the secure VPN to be treated as an untrusted device, or may connect to the file server through the POLEN path.

Size of Trace	Time to Analyze	State Size
1 Day	52 min	145MB (1.16MB per user)
1 Hour	2.49 min	145MB (1.16MB per user)

Figure 5: Processing time and storage costs for working set estimation in Polex.

In our experimental setup, all systems are Dell SMP systems with two 2.4GHz Intel P4 CPUs, and 3GB of RAM. All systems run a Linux 2.6 kernel and are connected using a Gigabit Ethernet switch. POLEX is configured to listen to all NFS v3 requests and responses. This is accomplished by enabling port monitoring on the switch. POLEN is configured to interpose on all NFS v3 requests and responses.

6.1 Evaluation of POLEX

Time and Storage Costs: The utility of POLEX to administrators is determined by the benefits of the functionality it provides and the costs associated with this functionality. We measure costs for POLEX in terms of the time it takes to process the network file system messages it receives, as well as the size of the state that must be stored to maintain a fixed amount of history (in terms of time). To quantify the accuracy, we perform offline analysis using a set of large file system traces provided by Harvard University [9]. These traces represent a month of network file system usage from the EE/CS Department at Harvard University.

Figure 5 shows the results in terms of processing time and the size of the resulting state, once processing has been completed. We measure these costs for trace samples of size corresponding to one day and one hour. One day of the trace represents 3.3GB of trace storage space corresponding to $6,308,023$ NFS request/response pairs. To process 1 day took 52 mins and generated 145MB (1.16MB per user on average) of state as history. This equates to an approximate time compression factor of 96%. Since we utilize Bloom filters to store per-user working set state, the storage costs remain fixed at approximately 145MB regardless of trace size. Therefore, we observe that POLEX imposes negligible costs in terms of processing time and state storage requirements.

6.2 Evaluation of POLEN

Accuracy: A key factor in the utility of POLEN is the accuracy with which the system operates. The accuracy is presented as the ratio of the set of file permissions that we approximate incorrectly to the total that we observe (including correctly approximated permissions). This provides the error or false positive rate for POLEN. To quantify these costs, we perform offline analysis using the Harvard traces.

To determine accuracy using the network file system traces, we choose two consecutive days worth of trace data. We randomly select ten users from those in the traces. We use the traces from the first day to perform per-user working set extraction using the POLEX algorithm. Then we run a test against the WS summaries using the traces from the second day, and measure the number of errors. An error is generated by a file or directory access, if the access is to a file (or directory) for which the user does not have the appropriate permission to execute the specific access, according to the user's WS summary. These errors represent file access attempts by the user that would be denied by the WSBAC

	Average Error Rate	Over-Estimation Rate
Run 1	1.08%	31.6%
Run 2	0.76%	41.2%
Run 3	1.02%	42.5%
Run 4	0.79%	36.5%
Run 5	0.97%	42.9%
Avg	0.92%	38.9%

Figure 6: WSBAC working set error and over-estimation rates.

	Day 2	Day 3	Day 4	Day 5	Day 6
User 1	0.262%	0.027%	0.017%	0.012%	0.096%
User 2	0.309%	4.40%	0%	3.30%	0.274%
User 3	0.386%	0.356%	0.818%	2.51%	0.608%
User 4	0.479%	1.82%	0.548%	0.655%	0.105%
User 5	0.181%	0.278%	0.177%	0.343%	0.270%
Avg	0.323%	1.38%	0.312%	1.36%	0.271%

Figure 7: WSBAC working set sensitivity.

system, but allowed by the network file server. We repeat this experiment five times using a different two days worth of trace data and randomly choosing a new set of ten users each time. The results are reported in Figure 6.

The figure shows the average error rate for each of the five runs. The maximum error rate is 1.08%, the minimum is 0.76%, and the total average is 0.92%. From these results, we observe that the average per-user accuracy is high (low error rates), and this confirms that WSBAC is not overly restrictive.

We further assess the rate at which an approximated working set is overly permissive. We measure this as the percentage of the estimated working set that is included during the estimation phase, but never accessed in the testing phase. These results are reported in the third column of Figure 6. As shown in the figure, the maximum over-estimation rate is 42.9%, the minimum is 31.6%, and the average is 38.9%. Although WSBAC is more restrictive than the standard network file system discretionary access control system, the results of this experiment show that there is still room for improvement.

Sensitivity: To understand how frequently a user's working set estimate must be updated, we perform a sensitivity analysis on the working set accuracy. For this experiment, we perform offline analysis similar to the accuracy measurements. This time we choose six consecutive days worth of traces. We use the traces from the first day to perform per-user working set extraction and test with the remaining five days worth of traces. Accuracy measurements are determined as before, and are reported in Figure 7.

The figure shows the average error rate for five randomly selected users over the five days following the working set extraction day. From these results, we observe that the average per-user accuracy remains fairly stable over a reasonable period of time. Users 1 and 5 are very stable across the five days, while users 3 and 4 are stable four out of five days. Finally, user 2 is the least stable fluctuating every other day, which possibly indicates that a multi-day extraction period might benefit some users. In general, the accuracy results are very promising, given the low error rates across the board.

Speculation: Since the amount of speculative accesses a user performs is related to the number of updates that must be validated by the user when they return to a trusted device, we measure the average rate of speculation for ten randomly chosen users from the traces. Due to space constraints, we do not provide a figure for these results. For this experiment, the average speculation rate is 1.44%, the maximum is 2.4%, and the minimum is 0.018%. These results are also promising since they imply a low burden on the average user with respect to the amount of manual validation that must be performed for speculative updates. Even for heavy users (500 or more requests per day), the results imply relatively low levels of manual intervention (7 average speculative requests per day). From these results, we conclude that WSBAC is highly automated, and does not require an administrator or the end-user to laboriously configure additional access control policies.

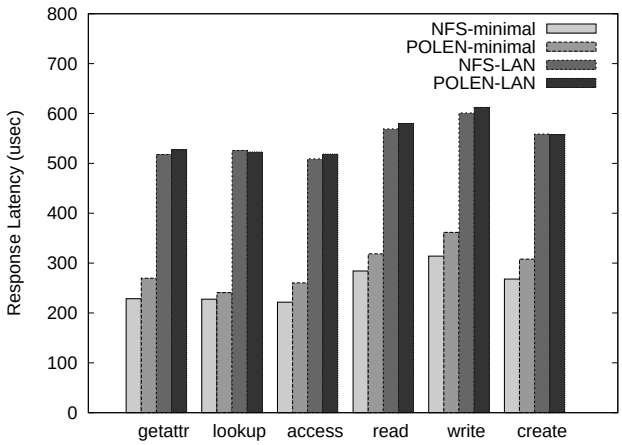

Figure 8: Microbenchmark results for Polen.

Performance–Microbenchmark: To study the fine-grained overhead of Polen with respect to the network file system, we utilize our own microbenchmark. This benchmark is an NFS client that issues requests without relying on the client file system interface. Using this benchmark eliminates any noise due to the client buffer cache and allows fine-grained measurements to be collected.

We present the operation latency of the default NFS protocol, as a baseline. Figure 8 shows the client observed latency for the most common NFS operations, as reported by various file system workload studies [9]. In the figure, each group of bars has four members, NFS and Polen in a minimal configuration, and NFS and Polen in a typical LAN configuration. The average round-trip time is 30μs for the minimal configuration and 300μs for the LAN configuration. The latency measurement for the minimal configuration represents the average latency measured through a 1Gbps network switch. Each bar presents the average response latency over 1000 instances.

We observe that Polen imposes modest overhead when compared to the NFS case for the minimal configuration. The largest overhead measured for the minimal case was 40μs, which represents a 15% performance degradation. Since typical LAN round-trip times are larger than 30μs, this represents the worst case performance for our system. Very little of the Polen processing time is hidden by the network latency. As we introduce delay in the network, most if not all of the Polen processing time is hidden due to the relative time of processing to network latency. In fact, at and beyond the 300μs mark (as shown in the figure as the LAN bars) the relative performance between base NFS and Polen is within 10μs on average which corresponds to, at most, a 2% overhead. For users accessing a corporate Intranet over a WAN, the situation is even better (e.g., from a satellite office or remote access situation). Since the typi-

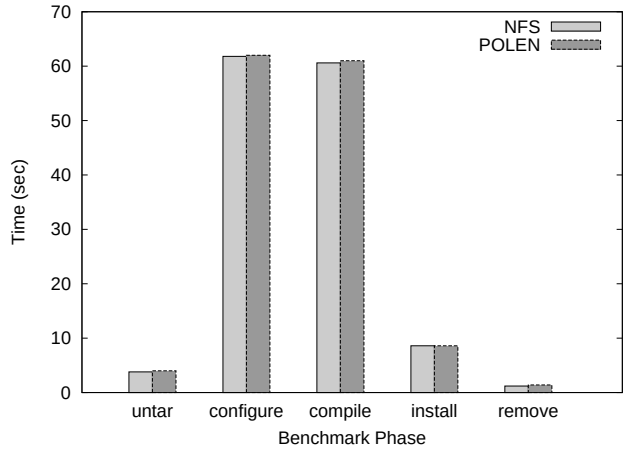

Figure 9: OpenSSH compilation for Polen.

cally observed network latencies in a WAN deployment are between 15ms - 30ms, we expect there to be no perceived performance impact on WAN users due to this system. We have also validated this experimentally, but do not present the results due to space constraints.

Performance–Application Benchmark: In the following, we compare the performance of NFS with POLEN WSBAC enforcement against the default NFS for an OpenSSH build benchmark similar to a modified Andrew Benchmark [13]. We measure the time taken to complete each of five phases (untar, configure, compile, install, and remove). To simulate the LAN conditions a typical user would experience, we introduce additional delay in the network, such that we increase the approximate round trip-time between hosts in the experiment to 300μs. The results are reported in Figure 9.

This figure shows the average time over five runs with a cold client cache for each phase of the benchmark from left to right. The bars in each group are base NFS and NFS with POLEN. We observe that POLEN imposes very low overheads (less than 2% in all cases) when compared to the normal NFS case for remote users over a LAN. We conclude from these and the microbenchmark results, presented earlier, that POLEN introduces no perceived performance overheads on network file system users.

6.3 Discussion

Although we utilize a well-understood set of network file system traces in our evaluation, they unfortunately do not allow us to fully evaluate the effectiveness of WSBAC. In particular, since the traces do not identify the types and capabilities of client devices, (e.g, trusted/untrusted, handheld/desktop/mobile), there are a number of dimensions that we cannot assess quantitatively in our evaluation. Although it is possible for differing capabilities in a user's device to affect how she accesses her file system data, it remains to determine whether it also affects which data items she will access. We intend to investigate this aspect further in the future by capturing real-world user traces that provide the necessary details to analyze the effects of various devices and their differing capabilities. However, we note that technological trends point to increasing capabilities of personal devices, such as laptops, handhelds and PDAs, which in turn provide a rich set of capabilities for users to access files in much the same way as they do from a desktop PC, such as a trusted corporate terminal. For example, the OpenMoko Freerunner has OpenSSH installed by default, and is capable of mounting network file systems (e.g., NFS, CIFS, etc.). We therefore hypothesize that WSBAC will be broadly applicable as the capabilities of such devices continues to increase.

7. RELATED WORK

This section describes the work related to WSBAC, which we divide into two categories: *(i)* Policy Extraction and Inference, and *(ii)* Context-Aware Access Control.

Policy Extraction and Inference: Role-Based Access Control (RBAC) [15, 11] has been proposed as a standard for network file system access control. Although there are many attractive features in favor of RBAC, legacy installations have to handle the onerous task of migrating their existing access control information (e.g., ACLs, access permissions, etc.) to Roles and Object Permission Matrices. For large installations, this task is a major road block towards acceptance of RBAC. Kuhlmann, *et al.*, [20] utilize Data Mining techniques to address this problem automatically, by learning common patterns in the existing ACLs or permissions lists in order to automatically define the roles in the RBAC system and place the appropriate users into these roles. In this fashion, they are able to extract the RBAC policy principles from existing legacy data. There has also been work to infer access control properties specified in the XACML language [3, 21], to infer and confine process privileges through the observation of process syscall behavior [23], and to automatically generate SELinux policies based on observing dynamic program behavior [29].

Although there has been substantial work in the general area of firewall and packet filter analysis, there are a number of works related specifically to firewall policy inference [12, 32]. Tongaonkar, *et al.*, [32], describe a method to infer high-level policies from low level firewall rule sets. They describe a method to generate a flattened rule set (high-level rules) by first generating an automaton based on the low-level rules. Golnabi, *et al.*, [12], use data mining techniques to learn a set of firewall rules from packet logs, based on packet frequencies. These observed rules are used to analyze the existing rule set configurations for firewalls as an aid to determine a new, more efficient, set of firewall rules.

Our work shares a similar goal to these works, in that we are attempting to determine efficient representations of the access control policies. We differ since we operate on network file systems, rather than firewalls. Additionally, we are not attempting to generate static rules sets, but are approximating the set of resources (files and directories) that a user needs to access in the near future based on their past accesses. It is not clear that the working set approach generalizes to network resources other than network file systems.

Finally, in the general area of policy inference and control, there has been work in using gray-box techniques to monitor and control the enforcement of operating systems policies (e.g., buffer cache, memory access control, file layout, etc.) [4]. The general technique is to either understand the policy under control a priori, or to infer it through external (to the system) observation. Additionally, it is acceptable to perturb the system under observation, in order to aid the process or to actually enact control over system policies. Our goal differs from typical gray-box techniques in that we do not assume a priori to understand the effective access control policy, instead we determine the WSBAC policy automatically per-user.

Context-Aware Access Control: Substantial work has been performed in the area of context-aware access control. The concept has been applied in mobile and pervasive computing to provide secure collaborations [33] for wireless and

mobile devices, to provide anonymous context-aware access control for ubiquitous services [34], for ubiquitous service provisioning [8], and adaptive context-aware access control for ad-hoc networks [25]. A number of works have been proposed in the area of context-aware access control for web services [17, 5]. These works all attempt to include context in the access control decision-making process, in some cases for mobile computing. Our work shares this general approach, but we utilize different context and apply it in a different manner to a different domain. First, we leverage a user's network file system access behavior to further restrict their access to those files that they are actually using. Second, we only apply this restriction for user access from untrusted devices.

8. CONCLUSIONS

In this paper, we presented the Working Set-Based Access Control scheme for network file systems. WSBAC is an access control technique that extracts per-user working sets through the observation of users' network file system accesses. We also presented the design and implementation of our WSBAC system, which is composed of two network agents: POLEX and POLEN. POLEX monitors network file system accesses for users of trusted devices, extracts per-user working sets, and produces compact per-user working set summaries. The summaries are used by POLEN to enforce WSBAC for untrusted devices. We evaluated our system using a set of real-world traces and our experiments validate our approach. The average accuracy for working set estimation is high (low error rates) and the costs in terms of POLEX processing time and storage requirements are low. Finally, we measured the performance overheads of POLEN, which were shown to be very low for typical LAN deployments and completely hidden for typical WAN deployment scenarios.

As future work we plan to deploy WSBAC in real-world scenarios to conduct user studies to better understand the effectiveness of the system. Additionally, a user study will provide us the opportunity to study the qualitative impact of WSBAC on real users as they interact with the system. Finally, it will allow us to produce further trace data to better analyze the impact of device capabilities on the behavior of users of network file systems.

9. ACKNOWLEDGEMENTS

We thank our shepherd Konstantin Beznosov and the anonymous reviewers for their valuable comments and suggestions. This work is supported in part by the National Science Foundation through grant number NSF-CNS-0831268.

10. REFERENCES

[1] Simple Distributed Security Infrastructure (SDSI) web page. `http://groups.csail.mit.edu/cis/sdsi.html`.

[2] American Management Association. 2007 Electronic Monitoring and Surveillance Survey. AMA Press Release – `http://press.amanet.org/press-releases/177/2007-electronic-monitoring-surveillance-survey/`, February 2008.

[3] A. Anderson. XACML Profile for Role Based Access Control (RBAC). *OASIS Access Control TC Committee Draft*, 1:13, 2004.

[4] A. C. Arpaci-Dusseau and R. H. Arpaci-Dusseau. Information and Control in Gray-Box Systems. In *SOSP'01*, 2001.

[5] R. Bhatti, E. Bertino, and A. Ghafoor. A Trust-Based Context-Aware Access Control Model for Web-Services. *Distributed and Parallel Databases*, 18(1), July 2005.

[6] A. Bohra, S. Smaldone, and L. Iftode. FRAC: Role Based Access Control for Network File Systems. In *NCA'07*, 2007.

[7] B. Callaghan, B. Pawlowski, and P. Staubach. *NFS Version 3 Protocol Specification, RFC 1813*. IETF, June 1995. http://www.ietf.org/rfc/rfc1813.txt.

[8] A. Corradi, R. Montanari, and D. Tibaldi. Context-Based Access Control for Ubiquitous Service Provisioning. In *COMPSAC'04*, 2004.

[9] D. Ellard, J. Ledlie, P. Malkani, and M. I. Seltzer. Passive NFS Tracing of Email and Research Workloads. In *FAST'03*, 2003.

[10] C. Ellison. IETF RFC 2692: SPKI Requirements, September 1999.

[11] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Trans. on Information and System Security*, 4(3):224–274, 2001.

[12] K. Golnabi, R. K. Min, L. Khan, and E. Al-Shaer. Analysis of Firewall Policy Rules Using Data Mining Techniques. In *NOMS'06*, 2006.

[13] J. H. Howard, M. L. Kazar, S. G. Menees, D. A. Nichols, M. Satyanarayanan, R. N. Sidebotham, and M. J. West. Scale and Performance in a Distributed File System. *ACM Trans. on Computer Systems*, 6(1):51–81, 1988.

[14] R. S. Inc. RSA SecurID Authenticators web page. `http://www.rsasecurity.com`.

[15] International Committee for Information Technology. Role-based access control. ANSI/INCITS 359-2004, Feb. 2004.

[16] T. Jaeger, D. King, K. Butler, S. Hallyn, J. Latten, and X. Zhang. Leveraging ipsec for mandatory per-packet access control. In *SecureComm'06*, 2006.

[17] V. Kapsalis, L. Hadellis, D. Karelis, and S. Koubias. A Dynamic Context-Aware Access Control Architecture for e-Services. *Computers and Security*, 25(7), October 2006.

[18] M. Kim, L. P. Cox, and B. D. Noble. Safety, Visibility, and Performance in a Wide-Area File System. In *FAST'02*, 2002.

[19] J. J. Kistler and M. Satyanarayanan. Disconnected Operation in the Coda File System. *ACM Trans. on Computer Systems*, 10(1):3–25, 1992.

[20] M. Kuhlmann, D. Shohat, and G. Schimpf. Role Mining - Revealing Business Roles for Security Administration Using Data Mining Technology. In *SACMAT'03*, 2003.

[21] E. Martin and T. Xie. Inferring Access-Control Policy Properties via Machine Learning. In *POLICY'06*, 2006.

[22] B. Petersen. Employee Monitoring: It's Not Paranoia You Really Are Being Watched! PC Magazine – `http://www.pcmag.com/article2/0,1759,2308369,00.asp`, May 2008.

[23] N. Provos. Improving Host Security with System Call Policies. In *SECURITY'03*, 2003.

[24] T. Pullar-Strecker. NZ bank Adds Security Online. `http://www.smh.com.au`, November 2004.

[25] A. Saidane. Adaptive Context-Aware Access Control Policy in Ad-Hoc Networks. In *ICAS'07*, 2007.

[26] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Attestation-based Policy Enforcement for Remote Access. In *CCS'04*, 2004.

[27] M. Schulman. LittleBrother is Watching You. Santa Clara University – `http://www.scu.edu/ethics/publications/iie/v9n2/brother.html`.

[28] S. Smaldone, A. Bohra, and L. Iftode. Filewall: A Firewall for Network File Systems. In *DASC'07*, 2007.

[29] B. T. Sniffen, D. R. Harris, and J. D. Ramsdell. Guided Policy Generation for Application Authors. Technical report, The MITRE Corporation, 2006.

[30] C. Soules and G. Ganger. Connections: Using Context to Enhance File Search. In *SOSP'05*, 2005.

[31] M. Spasojevic, T. Corporation, and M. Satyanarayanan. An Empirical Study of a Wide-Area Distributed File System. *ACM Trans. on Computer Systems*, 14(2):200–222, 1996.

[32] A. Tongaonkar, N. Inamdar, and R. Sekar. Inferring Higher Level Policies from Firewall Rules. In *LISA'07*, 2007.

[33] A. Toninelli, R. Montanari, L. Kagal, and O. Lassila. A Semantic Context-Aware Access Control Framework for Secure Collaborations in Pervasive Computing Environments. In *ISWC'06*, 2006.

[34] S. Yokoyama, E. Kamioka, and S. Yamada. An Anonymous Context Aware Access Control Architecture For Ubiquitous Services. In *MDM'06*, 2006.