Event-based Document Sensing for Insider Threats

Kenneth Anderson, Antonio Carzaniga, Dennis Heimbigner, Alexander Wolf

(kena,carzanig,dennis,alw@cs.colorado.edu)

CU-CS-968-04 February 6, 2004

University of Colorado at Boulder

Technical Report CU-CS-968-04
Department of Computer Science
Campus Box 430
University of Colorado
Boulder, Colorado 80309-0430

Event-based Document Sensing for Insider Threats

Kenneth Anderson, Antonio Carzaniga, Dennis Heimbigner, Alexander Wolf

6 February 2004

Abstract

In this report we describe a uniquely decentralized event-based system for online and offline tracking, analysis, and control of electronic document transactions in order to detect suspected insider misuse of those documents. Our system is focused on the problem of tracking and analyzing user's actions with respect to documents. On the user side, the proposed system will support the placing of document sensors into a heterogeneous collection of document editors (e.g., Microsoft Office) and browsers, including Web browsers. The system will provide a low entry barrier with respect to integrating new user-side tools. On the administrative side, the proposed system will support a dynamically changing set of online and offline analysis and control programs capable of providing such diverse capabilities as visualization, persistence, forensics, access control validation, and slicing of document access patterns based on, for example, user, or document, or transaction type, or time-line. The system as a whole will be portable, which we will demonstrate by providing integration of tools under both Linux and Windows.

1 Introduction

In this report we describe a uniquely decentralized event-based system for online and offline tracking, analysis, and control of electronic document transactions in order to detect suspected insider misuse of those documents.

Our system is focused on the problem of tracking and analyzing user's actions with respect to documents. On the user side, the proposed system will support the placing of document sensors into a heterogeneous collection of document editors (e.g., Microsoft Office) and browsers, including Web browsers. The system will provide a low entry barrier with respect to integrating new user-side tools. On the administrative side, the proposed system will support a dynamically changing set of online and offline analysis and control programs capable of providing such diverse capabilities as visualization, persistence, forensics, access control validation, and slicing of document access patterns based on, for example, user, or document, or transaction type, or time-line. The system as a whole will be portable, which we will demonstrate by providing integration of tools under both Linux and Windows.

Our system makes novel use of hypermedia technology to support analyses that rely on historical traces of document transactions. This component, *EventTrails*, will provide persistent storage for captured events and will organize these events into trails. Additionally, it will provide access/query and visualization services over these trails. Events can be arranged in hierarchies to support analysis at different levels of granularity; a set of "keystroke events" can be aggregated into a single "edit" event, for instance. EventTrails also supports queries against particular event patterns. This is especially useful in defining signatures for suspicious activities. For instance, a pattern might be used to detect "cut-and-paste" events between documents that have different

security classifications. When a query detects a "hit", a notification is sent to an analyst or analysis tool for further processing. Additionally, analysts can make use of an EventTrails browser to visually analyze the contents of an event trail, to compare a set of event trails "side-by-side", to examine the history of activities against a specific document, or to examine the activities of a specific user. Finally, the browser will support traversing to a document associated with a particular event via hypermedia links. Thus, an analyst can go from viewing an "edit event" for a document in the EventTrails browser to viewing that document directly in its native editing application.

Our system will also exploit our unique, scalable content-based routing technology (aka publish/subscribe or event notification technology) to support decentralized communication of document transaction notifications between sensors in the integrated applications and the administrative tools that analyze the notifications for suspicious event signatures. Because it uses wide-area publication of notifications, user tools can be monitored even though they are dispersed across many nodes of a network. Similarly, administrative activities can be initiated from a variety of network locations. Our communication infrastructure will also support the intentional transmission of control commands to document tools. This means that the commands sent to a tool are based on the current properties of that tool, such as its active document, its current access controls, or its specific user.

2 Approach

There are many approaches to developing document monitoring systems in support of hostile insider detection. For instance, one approach would be to develop a large, powerful database and require that all of an organization's documents be stored in this database. The problem of hostile insider detection would then be reduced to detecting attempts by an organization's employees to subvert the access controls maintained by the database. A second approach would be to develop a single tool or set of tools "blessed" by the organization for use by its employees in creating the organization's documents. This approach allows for access control and monitoring of employee actions to be embedded in this tool and the problem of hostile insider detection would be reduced to detecting attempts to subvert these mechanisms. These approaches essentially try to make one aspect of the organization's computing environment homogeneous and then attempt to exploit that homogeneity to offer a solution to the problem of hostile insider detection.

We propose a third approach that acknowledges the extreme heterogeneity found in these environments in terms of the types of documents and applications used by an organization. It also exploits the highly-distributed nature of these environments, especially in the context of a large organization with multiple, geographically-distributed sites.

Our approach relies on the content-based distribution of event notifications generated by sensors that monitor the actions applied to an organization's documents. These notifications are delivered to a dynamically changeable set of analysis and control programs that allow an organization to monitor the actions of their employees and respond to situations that indicate the presence of a hostile insider. We employ a variety of techniques to develop document sensors including application and operating system integration as well as the crawling of Web-based resources. The cost of integrating an application is low, allowing an organization to potentially integrate the majority of its applications.

We now examine a set of scenarios that illustrate the types of activities that hostile insiders may perform and then present an architecture that allows our approach to support these scenarios.

3 Scenarios

We have identified a number of scenarios representing some examples of potential insider threats and suspect activities. The scenarios are intended to highlight differences between desired/expected behavior and deviations from that behavior. As with any system searching for such anomalies, we anticipate the occurrence of false alarms. Nevertheless, we believe these scenarios are representative of potential insider attacks and we will use them to drive many of our example analyzers for our prototype system.

3.1 Document Scanning

An insider intent on performing malicious activity will at some point have to access relevant documents. By analog with network-level port scanning, we identify the notion of *document scanning*. The signature for such an activity would be the rapid reading of a number of related documents from some source. If a user scans a set of related documents (a whole directory, for example), that may be suspicious and would call for further investigation.

3.2 Scan and Store

As with port-scanning, a scan is usually associated with some other activity that embodies the actual malicious intent. If a user is scanning a set of documents, the associated activity might be saving a copy of each document or using cut-and-paste to leak information from a document into some other storage container.

3.3 Scan and Modify

Another example of malicious scanning might involve the modification of each scanned document. Of course sometimes such wholesale modifications are legitimate. On the other hand, if the modification is to remove all of the contents of a document, then it is relatively likely that the activity is malicious.

3.4 Changing Link Structures

The web provides a whole class of potential attacks. One such form of attack involves changing the information presented by an organization's website. While this type of attack typically involves "defacing" the home page of the website, a more insidious form of attack is to leave the content of a page alone and instead change the links of the page to point at content provided by the attacker. As such, if a user adds new documents to a website and changes existing document links to point at the new documents, the intent may be malicious and ideally such changes could be verified. More generally, changes to a website's link structures need to be monitored to guard against these types of attacks.

3.5 Coordinated Attacks

An organization may find itself under attack from a coordinated group of hostile insiders, in which each individual action taken by the group falls below the threshold of suspicious, but the result of all their actions taken together causes information to be leaked or changed in ways that harm the organization.

In one example of this type of attack, one user changes the access rights to a document such that it is available to an unauthorized user. This second user copies information from that document into a new document and saves the new document in one of their authorized directories. The first

user now changes the access rights of the compromised document to their original settings. In order to detect this type of attack, infrastructure is needed to allow the event streams of multiple users to be compared “side-by-side”.

Coordinated scan attacks are also a possibility. Any of the scan scenarios may be carried out by multiple cooperating insiders. For example, one person may scan and modify half of the documents in a repository and another person may do the same for the other half. Detecting such coordinated attacks will rely on a capability to focus on the documents and their organization rather than on individual users; that is, it will be the effect that signals an attack rather than a specific person’s activities.

3.6 Access Control Failures

Ideally, access control enforcement would be both precise and effective. Unfortunately, neither is true in all situations. Sometimes available access controls are too coarse to provide the precise control desired by an administrator. It might allow access permissions on a file but not be able to do it on a per-user basis, for example. Additionally, administrators make errors in assigning access controls or by forgetting to propagate existing controls to new document sources. The result is defined, but ineffective controls.

Both imprecise and ineffective controls can lead to scenarios in which a user inadvertently gains access to a document that should be restricted. It would be desirable to provide a separate validation path whereby accesses to documents were validated separately from the local access control mechanisms. In effect this advocates a “belt and suspenders” approach.

3.7 Workflow Violations

Trying to characterize the normal behavior of a user is quite difficult as shown in the anomaly-based intrusion detection systems. In more constrained situations, however, it may be possible to closely define the expected behavior and to generate alerts when actual behavior deviates from that expected behavior.

Workflow-driven activities are one such example of highly constrained behavior. Workflows associate well-defined sequences of activities with documents in order to make document processing simpler and more efficient. Many enterprises utilize workflows, including insurance companies, financial institutions, and manufacturers. We hypothesize that Federal government intelligence agencies do (or should) implement workflow-driven document processing.

A workflow violation involves an insider who normally should be following a workflow process, but deviates from that process in some significant way.

3.8 Role Violations

Legitimate document accesses often follow role lines, which in turn often follow organizational lines. Thus, one would not normally expect a bursar in the finance department to be accessing engineering documents. If that person was also modifying engineering documents, that would be even more alarming.

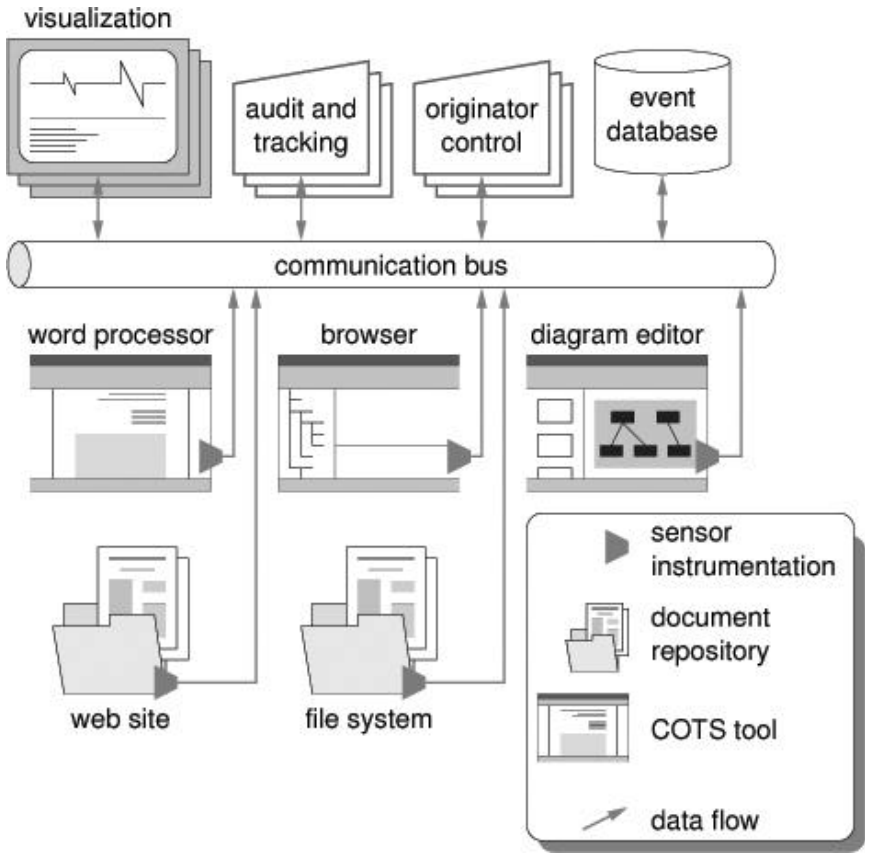
A role violation involves an insider who normally should be accessing documents appropriate to his role but is detected accessing other documents not within his purview.

4 Architecture

Our document tracking system employs a loosely-coupled architecture that directly supports the heterogeneous and distributed nature of modern computing environments (see Figure 1). Sensors monitor the actions performed on an organization's documents. These sensors gain access to these actions using a variety of techniques, including application integration, crawling of Web-based resources, and operating system integration. In response to a user action, a document sensor generates an event. Each event contains information about a particular document transaction (such as "file created") and is sent to our communication infrastructure for delivery to the administrative components of our architecture.

This infrastructure makes use of content-based routing, aka event notification technology, to deliver the document transaction notifications to only those components that have registered interest in them. Thus, a tool that monitors the actions applied to HTML documents need not receive notifications of actions applied to spreadsheets.

The administrative components consist of a dynamically changing set of online and offline analysis and control programs capable of providing such diverse capabilities as visualization, persistence, forensics, originator control, access control validation, and pattern slicing of document transactions. In addition, these tools can allow organizations to respond to hostile insiders by supporting the ability to issue control commands to document sensors asking, for instance, all sensors related to a particular user to provide more detail than normal in their event notifications. These control commands are routed using the content-based communications infrastructure allowing administrators to issue commands without having to know the locations of all the active document sensors distributed throughout an organization's computing environment.



Figue 1. Document Tracking and Control System Architecture

One important component of the administrative tools is the event database, known as *EventTrails*, that is responsible for making persistent each event generated by the document sensors. This database organizes events into "trails" that can be used to analyze, for instance, all the events of a particular user or all the events performed on a particular document. These trails can be accessed by human analysts using the EventTrails browser. Additionally, this component supports *originator control* by allowing document owners to specify when they should be notified about actions applied to their documents. It is possible, for instance, to have a document's originators notified whenever their document is opened, moved, or otherwise manipulated.

Our architecture represents a uniquely decentralized document monitoring infrastructure that supports both online and offline tracking, analysis, and control of electronic document transactions in order to detect the presence of hostile insiders. We now present each of the major components of our architecture: the document sensors, the communication infrastructure, and the administrative tools.

5 Document Sensors

A key component to our approach is a set of document sensors that monitor actions applied to the documents of an information space and generate events when interesting actions occur. The use of th term "sensor" is deliberately generic, since we intend to make use of a variety of approaches to develop a wide range of document sensors. The majority of our document sensors

will be developed by integrating the applications that both edit and view an organization's documents. By "integrating", we mean that applications will be customized (typically via plug-ins or scripts) to generate events to our event notification system as their associated documents are created, opened, edited, and saved. An additional approach is to make use of open hypermedia technology to monitor the link structures of an organization's website and detect when links are being added, deleted, or changed. A third approach is to integrate an operating system with our event notification system to generate events related to changes in a computer's file system, such as the manipulation of directories and the files they contain. We now discuss each of these approaches in detail.

5.1 Document Sensors via Application Integration

One way of implementing a document sensor is by integrating a small software component into a document's associated editors/browsers. This component or plug-in has access to an application's internal event stream and can detect when an application is opening, creating, editing, and saving its documents. These components typically have access to operating system services and can also determine, for instance, the identity of the user that launched its application. For the purposes of this project, these components will also be integrated with our communication mechanism; this integration allows them to generate events that capture the user's actions with the application. These events are then sent to our analysis tools.

This work will build on the experience of Anderson in the area of third-party application integration via his work on open hypermedia. For his work on the Chimera open hypermedia system [Anderson 1994; Anderson 1997; Anderson 1999a; Anderson 1999b; Anderson 2000a; Anderson 2000b], has integrated a wide variety of third-party applications including Microsoft Office, Internet Explorer, Netscape Navigator, sound players, MPEG players, text editors, and the like. Indeed, the open hypermedia community has developed a wide range of generic techniques to integrate open hypermedia services into third-party applications [Davis 1994; Whitehead 1997; Wiil 1996]. These techniques span the use of application wrappers, "screen scrapers", and software plug-ins. We intend to adapt these techniques to create document sensors in "real world" applications, initially targeting the widely used tools of Microsoft Office and Internet Explorer.

While the software component embedded in the application is "platform specific" and in some cases "application specific", certain generic techniques can be applied to keep the cost of each individual integration low. In particular, our choice of using an event notification system as our primary communication mechanism eases the task of an individual plug-in reporting the actions of its associated application. In addition, individual platforms provide generic mechanisms for communicating with and/or scripting the actions of individual applications. For the Windows platform, we intend to use a Java/DCOM bridge (JIntegra from Intrinsyc < http://www.intrinsic.com>) to design a generic application integration component that handles the tasks of publishing events that are received from application-specific VisualBasic scripts embedded in our target applications.

A key design issue related to our document sensors is the granularity of the events generated by the sensor. For instance, our approach to application integration will allow our sensors to generate events down to the granularity of individual keystrokes. In each instance, we will need to determine if that level of granularity is desirable. At a minimum, each document sensor will be able to generate time-stamped events that indicate when an application has opened a document,

closed a document, edited a document, saved a document, performed a "cut/copy" operation and performed a "paste" operation. Other application-specific events will be supported as makes sense for a particular application. Thus, for instance, it may be important to capture "image manipulation" events in Adobe Photoshop in order to detect when a hostile insider is deliberately degrading the image quality of an important piece of organizational data.

5.2 Document Sensors for Web-Based Resources

An additional class of document sensors are those associated with Web-based resources. Our primary approach to building document sensors involves integrating the applications which edit and view documents with an event notification system. This approach handles a large number of events associated with Web-based resources. For instance, by integrating a Web browser, such as Internet Explorer, we can generate events that track the actions of a user on-line, including the resources viewed, the link traversals performed, the use of the browser's bookmark mechanism, and the use of the browser's history mechanism. By integrating a Web server, we can keep track of the documents retrieved by users of an organization's website. Finally, by integrating a text editor or Web development tool, we can track when a website's documents are being created, opened, edited, and saved.

However, application integration is not enough to completely cover the types of changes that a hostile insider may make when attempting to subvert an organization's Web-based resources. In particular, it would be difficult to generate events that track changes to a website's link structures (e.g. the links that span the documents of an organization's website and the links that point to documents on external websites) using the application integration technique. In order to address this shortcoming, we propose to develop a document sensor for link structures that builds on the extensive experience of the PI in the field of open hypermedia [Østerbye 1996]. Over the past decade, the PI has developed the Chimera open hypermedia system [Anderson 1994; Anderson 1997; Anderson 1999a; Anderson 1999b; Anderson 2000a; Anderson 2000b]; Chimera is a middleware system that provides hypermedia services (such as anchor creation, link creation, and link traversal) to third-party applications. It is also integrated with the World Wide Web [Anderson 1997]. Chimera provides a conceptual model [Anderson 2000a] consisting of the following concepts: *applications*, *documents*, *views*, *anchors*, and *links*. These concepts allow the modeling of multiple views of documents when viewed by different applications or even when the same application can generate multiple views of the same underlying data (such as when a spreadsheet application provides a "chart view" of data stored in its default "rows and columns view"). Anchors specify "areas of interest" within a particular view (such as a particular phrase within a text document) and links are used to group related anchors together. With these concepts, it is possible to model the link structures found in modern websites.

As such, our document sensor for link structures will involve extending Chimera to crawl the contents of an organization's website to build a model of the site's link structures. The crawling of the site can be done on a periodic basis, or it can be triggered automatically by detecting "edit" and "save" events associated with documents or directories known to be included in the website. Each time the crawler is invoked, it creates a new model of the website's link structures. This model is then compared to the previous model and Chimera will generate a series of "link created", "link deleted", and "link edited" events for all such updates detected by the comparison. The previous model will then be archived, and the new model will become current.

Once this sensor has been implemented, it will be possible to implement analyzers that monitor events related to Web-based resources looking for anomalous signatures that may indicate an attack on an organization's website by a hostile insider (as discussed in Section 3.4).

5.3 Document Sensors via Operating System Integration

Inserting sensors into an operating system is desirable because it can capture any event that affects the file system. These events are of necessity less specific than the kinds of events captured by an integrated application because file system operations are at a lower level of specificity; they can track file open, close, modify, and delete. They cannot easily track, for example, specific keystrokes or cut-and-paste events. In addition to file system events, it is likely to be useful to capture security related events such as access attempts that fail (or succeed), and access to system files.

Capturing operating system level events is important, however, because the operating system can capture events for any application, integrated or not. Thus, they provide an important base level of event capture. As we discuss in Section 9, these operating system events can also be used to cross-check application events to ensure that a document is being manipulated using an approved integrated tool.

For the base part of this project, we propose to insert sensors into the Linux operating system. Our approach is to exploit the recently developed Linux Security Module mechanism that supports the insertion of hooks into various operating system activities, including file system activities.

6 Content-Based Communication and Component Integration

Content-based publish/subscribe communication is a service in which the flow of messages, from senders to receivers, is driven by the content of the messages, rather than by explicit addresses assigned by senders and attached to the messages [Carzaniga, Wolf 2001b]. Using a content-based communication service, receivers declare their interests by means of selection predicates, while senders simply publish messages. The service consists of delivering to any and all receivers each message that matches the selection predicates declared by those receivers.

Content-based communication provides a powerful paradigm upon which to base a document tracking system. It supports decentralized communication of document transaction notifications between sensors in the integrated editors and browsers and the administrative tools that analyze the notifications for suspicious event signatures. User tools can be monitored even though they are dispersed across many nodes of a network, and administrative activities can be initiated from a variety of network locations. Content-based communication can also support the *intentional transmission* of control commands to document tools; that is, control messages are sent to tools based on the current properties of the tool, such as its active document, its current access controls, or its specific user.

In a content-based service model, message content is structured as a set of attribute/value pairs, and a selection predicate is a query over the values of individual attributes. For example, a message might have the following content.

[class="alert", severity=6, device-type="web-server", alert-type="hardware failure"]

This would match, for example, a selection predicate such as this one.

[alert-type="intrusion" and severity > 2 or class="alert" and device-type="web-server"]

Content-based communication has two major advantages over traditional “address-based” client/server or peer-to-peer communication:

- Content-based communication does not require mutual knowledge between components. Because of this feature, content-based communication naturally supports dynamic architectures built from loosely coupled, “pluggable”, possibly distributed components. For example, in the architecture described in Section 4, a content-based communication bus allows for seamless integration of new analysis and visualization components.
- Communication is controlled by receivers on the basis of their interests. This means that the communication medium offers a powerful selection mechanism that filters out uninteresting information from the data streams on the behalf of receiver components.

These features make content-based communication an ideal integration mechanism for the components of a monitoring and analysis system.

We plan to provide content-based publish/subscribe communication in this project through a system called Siena, which we have been developing and publicly distributing since 1998 (http://www.cs.colorado.edu/serl/siena). Siena was designed specifically for scalable performance under varying numbers of subscribers, publishers, and message-traffic loads. Its goals have been met through an extensive research thrust focusing on performance optimization, both in local-area and wide-area deployments [Carzaniga, Rosenblum, Wolf 2001a; Carzaniga, Wolf 2001b; Carzaniga, Wolf 2002]. We propose to apply the results of that research to this project and to explore additional optimizations that may be enabled by exploiting the specific domain of document transactions.

We also plan to use Siena to distribute control commands to clients, such as integrated tools or online analyzers. Our approach is unique in that it supports directing commands to clients based on the current properties of the client. We refer to this as intentional control or query/advertise [Heimbigner 2001; Heimbigner 2003]. In our approach, user tools provide dynamically changing descriptions of their current properties, such as the user or the document being edited, or their multilevel security status. Administrators can then direct commands to tools based on those properties without having to know a priori the specific set of clients matching that set of properties. Thus, for example, a single command can reach all clients currently operating at the top-secret security level.

7 Analysis Capabilities

We envision two broad classes of administrative tools: analyzers, and controllers. Analyzer tools are designed to receive and process streams of events about document transactions. The purpose of these tools is to detect possible suspicious insider activity. Control tools are designed to configure other tools, especially user-side tools to control their operation, typically in response to suspicious activities. This section discusses analysis tools, while control is discussed in Section 8.

An important feature of our proposed system is its ability to provide an easily extendable and dynamically changeable set of analysis tools. We achieve this by providing an event stream of all document transactions and allowing tools to dynamically attach themselves to the event stream and receive selected substreams of events of interest to each tool As illustrated in Figure 1, this

capability comes from our use of a publish/subscribe (content-based routing) system. Tools can easily connect themselves to the publish/subscribe system and immediately begin to receive exactly the events in which they have expressed interest.

We divide the class of analyzer tools into *online* and *offline* tools. Online tools tap into the event stream and process it in essentially real-time. They are used to watch for such things as document scans (Section 3.1), or to validate access control enforcement. Offline tools operate on captured histories of events. They are used for such things as data mining or post-mortem analysis. They require the support of a persistence mechanism that can capture and organize the events. In the next two sections, we will discuss our support for both kinds of analyzers.

7.1 Online Analysis: The Analysis Engine

We anticipate that many online analyzers can share a common generic architecture, which we will refer to as the analysis engine architecture (Figure 2). As a rule, we expect these engines to operate independently over different substreams of events; they are in effect analysis agents looking for suspicious behavior and reporting it.

The basic operation of the engine is to analyze document transaction events in real-time for suspicious activity. If such activity is detected, the engine generates one or more alert events signaling the activity of interest. As with document transaction events, these alerts are also propagated by our publish/subscribe system so that other analyzers or controllers can receive them and respond in turn. In fact, one may have a hierarchy of engines each analyzing alerts from the engines in the next level down in the hierarchy.

In many cases, an analysis engine may take up to three parameters: a signature, a state, and a state model. The signature defines the kind of suspicious activity for which the engine is searching. It could be defined declaratively as a set of rules, or procedurally as a small program that acts as a plug-in to the engine. Again we are drawing on intrusion detection systems which first defined the notion of signature.

The state provides a data structure for recording information about the stream of events passing through the engine. The state model of course defines the structure of the state. Often, the signature will be defined in terms of the state model.

We have identified some examples of engines based on some of the scenarios described in Section 3.

Document Scanning Detectors: One of our scenarios involved an insider who was scanning a related set of documents, possibly to leak their contents. In order to detect such activity we would need a state that records a sequence of document accesses. Further, our state model must

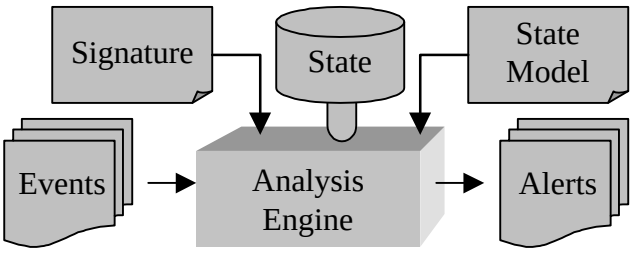


Figure 2. Analysis Engine Architecture.

define document groupings in order to specify what it means for a set of documents to be "related". A grouping might be defined to be a subtree of a file system, or it might be defined to be all documents with the same URL prefix and accessible through a given web server. Our signature would define specific parameters indicating when an alert should be generated. These parameters might include percentage of the group that is scanned and the time frame for the scan. Similar engines might be used to detect specific kinds of scans such as scans modifying large numbers of documents, or scans that appear to be leaking information.

Access Control Validation Detectors: One scenario involved ineffective or imprecise access control enforcement. The corresponding analysis engine would examining actual document accesses looking for evidence of potentially ineffective controls. The signature would specify the desired access controls. The state model would define the set of document transactions which are expected to be controlled.

Workflow Validation Detectors: For our workflow-based scenario, we can easily envision an analysis engine that monitors selected documents to ensure that their processing actually conforms to the expected workflow. Unexpected transactions might signal a suspicious insider activity. Our state model in this case is the set of documents and the workflow state, and our state model is the space of controlled documents and the workflow model. The signature might specify the class of documents and users to watch.

Role Access Validation Detectors: The role scenario leads to an engine which has an organization chart (including roles) as its model and some mapping of groups of documents to people and roles on that chart. Given this information, it would be relatively simple to identify many kinds of suspicious document accesses.

7.2 Offline Analysis: EventTrails

Many kinds of analyzers require access to large document transaction histories in order to perform offline, non-real-time analyses. Examples include post-mortem forensics, profiling user-oriented activities and profiling document-oriented activities. In order to carry out such activities, it will be necessary to provide for the capture and persistent storage of document transactions. In our system, that becomes the capture and storage of the event traffic of our publish/subscribe communications infrastructure.

Our approach to developing sensors that deliver events related to an organization's documents is extremely flexible and scalable. However, it leads to the generation of a significant number of events. The *EventTrails* component of our architecture is responsible for capturing the events of our document sensors and organizing them into "event trails" to support insider threat analysis. Thus, it will be possible for an analyst to view the event trail associated with a particular document (or set of documents) or with a particular user. A document's event trail will reveal when a document was opened, edited, saved, or otherwise manipulated and by whom. A user's event trail will make it possible to trace their actions through an information space.

EventTrails is a critical component to our overall approach. It ensures that the effort to develop a new document sensor or to integrate a new application is not wasted, since their associated events will not be "lost". Furthermore, once the events have been saved, indexed, and organized into trails, its access/query APIs complement the work of the analysis engine in detecting anomalous patterns in the event streams. Finally, once an anomaly is detected, the EventTrails browser will support detailed analysis by humans to help guard against false positives or to

support additional analysis to detect if a potential hostile insider is working in isolation or collaborating with others.

There are three main components to the EventTrails system: event persistence, information access/query, and visualization.

7.2.1 Event Persistence

The event persistence component is responsible for capturing and indexing each event generated by the document sensors. Each event will be stored in a database (the type of database to be used is an open research issue). An index will be maintained for each of the core attributes ("document", "user", "timestamp", etc.) guaranteed to be associated with each sensor-generated event. These indexes will make it possible to quickly assemble event trails for individual users and documents and will also assist in generating aggregate trails over sets of documents and/or teams of users. In addition, the event persistence component will support the specification of event hierarchies, in which multiple instances of a particular type of event can be aggregated into an event with higher-level semantics. Thus, a series of keystroke events generated by Microsoft Word for a particular document can be aggregated into a "document edited" event. A series of "document edited" events can be aggregated into a "new document version" event, etc. Event hierarchies allow analysts to choose the level of granularity at which insider threat analysis is performed. Thus, a threat signature may be defined with respect to "document edited" events; once an anomalous sequence of these events is detected, the analyst can access the related keystroke events to gain an understanding of how a particular document (or set of documents) was altered by a potentially hostile insider.

7.2.2 Query API

The query API will allow events to be retrieved from event trails based on the values of their attributes. It will also support searching trails for instances of event patterns. For instance, a particular analysis algorithm may focus on the "cut-and-paste" events of a user especially when these events occur among documents that have different security classifications. These event pattern queries can be issued on demand or be stored persistently such that they continually apply themselves to event trails as new events arrive. When a persistent query detects a "hit", EventTrails can be configured to send a notification to the analysis engine to trigger additional analysis of the pattern (to guard against false positives) or to the originators of the documents involved in the "hit". This latter capability is especially important in supporting *originator control* of documents. The authors of a document can store persistent queries in EventTrails such that they are automatically notified when one of their documents has been viewed, moved, or otherwise manipulated by other members of their organization. These queries provide additional benefits beyond support for insider threat analysis. In particular, they can be used to increase a team's awareness of the actions of fellow team members. Thus, a team can set up queries such that all team members are notified when a shared document is manipulated, perhaps when the team is initially creating the document and want to stay informed about the document's rapidly changing status.

The information access/query component consists of a set of application program interfaces (APIs) that allow event trails to be assembled and analyzed. The information access API will allow specific event trails to be retrieved from the EventTrails database and will provide access routines for examining individual events or iterating over sets of events within a particular trail. In addition, the information access API will allow access up and down event hierarchies

allowing programs to switch granularities as needed during an analysis session. The information access API will also provide routines for creating "snippets" of event trails that can be named and stored for later analysis or perhaps even used as evidence against a potential hostile insider. These routines will provide the ability to create, name, save, and delete individual event snippets as well as the ability to add and delete events to these snippets. In addition, set operations (such as union, intersect, etc.) will be provided to create new snippets from existing ones.

7.2.3 Visualization

The visualization component will provide a browser for human analysts to directly view the contents of particular event trails. The browser will be able to show the structure of a particular event trail, allow analysts to view the values of an event's attributes, see the events of multiple trails "side-by-side" (which may be useful in determining when two hostile insiders are performing separate but coordinated actions to attack an organization's information space), and expand and collapse event hierarchies. In addition, the browser will support the viewing of event trails at different levels of resolution to support analysts zooming out to get the big picture or zooming in to analyze a particular section of a trail in depth. In addition, the EventTrails browser will be integrated with the Chimera open hypermedia system [Anderson 2000a] to allow analysts to quickly access the documents and directories referenced by the events contained in an event trail. Thus, an analyst can go from viewing an "edit event" for a particular document in the EventTrails browser to viewing that document directly in its native editing application via a link traversal handled by Chimera.

8 Control

Control of documents is the complement to monitoring of document transactions, and is of crucial importance. This project assumes that all traditional security controls are employed to control document access. These include enforcement of access rights, and *sand-boxing* of tools to the degree that the operating system supports them. They represent the first line of defense against insider attacks.

But it is often the case that existing controls are not as discriminating as might be desired. Further, many tools have no access controls of their own; they rely on the underlying operating system to enforce correct access. It is rare, however, for an operating system (or an application) to enforce, for example, read-once semantics on a document.

Our approach supports two notions of control: document access control and sensor control. Document access control will provide a request/response mechanism to allow integrated applications (and operating systems) to request permission to perform some activity on a document before it actually occurs. The exact activities so controlled is determined by the implementation of sensors in the application. By using our communication substrate, the request can be broadcast so that many controllers can respond to it. In the event that multiple responses occur, the most restrictive would apply.

Our other notion of control, sensor control, refers to a capability to dynamically modify the operation and existence of sensors in integrated applications and operating systems. Of course, this is only possible when the tool can be made to support such modifications. This capability is useful, for example, in the presence of suspicious activities. An administrator may wish to reconfigure various tools so that they respond in a non-standard way. Thus, if some particular user is under suspicion, then additional sensors may be deployed when that user invokes various

tools or accesses important documents. This capability can draw on the results of our prior research under the Willow project [Wolf, et al 2000; Knight, et al 2001a; Knight, et al 2001b] which supports the dynamic reconfiguration of a system to "harden" it against attacks.

We believe that this ability to change the sensor configuration is an important and unique capability that will significantly enhance the value of our system.

9 Protecting the Infrastructure

We recognize that insiders will attempt to bypass and subvert our infrastructure, but we have a number of approaches available to us to detect or prevent such actions.

The basic communication infrastructure supports both privacy and authentication. The channels between clients and the content-based routers can be encrypted using SSL. Additionally, the messages that are passed can themselves be signed to guarantee that the received content has not been tampered with and came from a known source.

Once messages leave our sensors they will propagate through one or more routers to the analysis tools. We assume that at some point in that progression they become effectively "safe" in the sense that an insider could not affect them. The implicit assumption here is that the analysis tools and most of the routers reside on trusted machines that are not accessible to the insider.

Thus our problem reduces to preventing spoofing of messages before they become safe. This effectively means we must detect any attempt to bypass the integrated tools or to use sensor-free tools, or to use tools that generate incorrect messages.

A man-in-the-middle attack might set up a fake router to filter messages from legitimate tools. This can be handled by use of authentication protocols so that the tool knows that it is talking to a trusted router.

The use of sensor-free tools presents a different challenge. In those cases where the document is coming from a server, we can institute additional authentication protocols to enforce use of the integrated tools. In other cases, we can detect the use of sensor-free tools through coordinated validation of document transactions. When we have instrumented both the tools and the operating system, we have two independent, but related, sets of document transaction events. We can analyze these two streams for inconsistencies that might signal the use of a non-compliant tool. If, for example, the OS reports that an important document has been opened, but we do not receive a corresponding notification from the appropriate integrated tool, then that is an indication that the integrated tool has been made ineffective, and we can raise an appropriate alert.

10 Evaluation

We plan to evaluate our approach with respect to feasibility, and with respect to its effectiveness in supporting the scenarios discussed in Section 3. We intend to perform our evaluation using an iterative process that will provide continuous feedback of our efforts throughout the development process.

With respect to feasibility, we intend to develop "proof-of-concept" prototypes of each component of our architecture. This includes integrating a set of "real world" applications into our communications infrastructure (we intend to target Microsoft Office and Internet Explorer initially), developing a Web crawler for the Chimera open hypermedia system to support the

monitoring of changes to a website's link structures, and altering the Linux operating system to deliver events related to file system events. Furthermore it involves developing prototypes of EventTrails and its associated browser. Additionally, it involves developing a prototype implementation of the analysis engine and creating several different instances of the engine configured to support different types of state models and event signatures (as discussed in Section 7.1). Finally, it involves the development of example control programs to demonstrate our ability to respond to situations in which a potential hostile insider has been identified.

The order in which the prototypes are developed will be driven by an iterative process that targets a particular scenario and develops the software needed to support it. For instance, we may start by selecting a subset of the document scanning scenarios and develop the Linux operating system integration, the communications infrastructure, the analysis engine, and the relevant state models and event signatures. We would then deploy this software and simulate various scanning activities. The goals of the simulation will be to ensure that our document scanning analyzers can detect scanning operations and then to improve their ability to cope with false positives.

We will then pick another scenario and iterate, developing the next demonstration, using feedback from the previous iteration to avoid problems and enhance our techniques. We will continue this iterative process until all of our required proof-of-concept prototypes have been developed and all of our scenarios have been evaluated.

11 Related Work

11.1 Open Hypermedia

Open hypermedia [Østerbye 1996] is a technology that supports the creation and traversal of navigational relationships among a heterogeneous set of applications. In the same way that the hypermedia services of the World Wide Web [Berners-Lee 1996] enable navigation among related Web pages, open hypermedia allows users to navigate among the documents of integrated applications. Open hypermedia, however, provides a higher level of hypermedia functionality than that found on the Web.

In particular, open hypermedia is not limited to a read-only environment; users have the power to freely create links between information regardless of the number and types of data involved. There are three technical characteristics of open hypermedia systems that enable this ability: advanced hypermedia data models, externally managed relationships, and third-party application integration.

Advanced hypermedia data models: The advanced hypermedia data models of open hypermedia systems allow them to create links between multiple sets of documents. Links can be typed, and meta-information about the documents being related can be stored. These models are more powerful when compared to the hypermedia data model of the Web (as provided by HTML), which consists of one-way pointers embedded inside documents.

Externally managed relationships: Externally managed relationships imply that links are stored separate from the content being related. This approach, again, contrasts to the approach taken by the Web, in which link information is embedded directly within the content of HTML documents. The benefits of externally managed links include the ability to link to information located on read-only media or within material protected by read-only access permissions and the ability to associate multiple sets of links with the same information.

Third-party application integration: Finally, third-party application integration allows hypermedia services to be provided in the tools used by end-users on a daily basis. This approach also contrasts with the Web where typically the only applications that provide hypermedia services are Web browsers. With the open hypermedia approach, end-user applications provide hypermedia services directly, allowing users to create anchors and links over a wide range of applications and data. The open hypermedia field has developed a wide range of techniques for integrating legacy systems into an open hypermedia framework [Davis 1994; Whitehead 1997; Wiil 1996] while new applications can be designed and implemented to directly support and provide open hypermedia services.

Open hypermedia systems provide the extensible infrastructure that enable hypermedia services over an open set of data, tools, and users [Østerbye 1996]. The first open hypermedia system appeared in 1989 [Pearl 1989] and the field has been evolving rapidly ever since. Østerbye and Wiil present a good history and overview of open hypermedia systems in [Østerbye 1996]. There are many open hypermedia systems in existence. Examples include Chimera [Anderson 1994; Anderson 1997; Anderson 1999a; Anderson 1999b; Anderson 2000a; Anderson 2000b], Microcosm [Hall 1996], DHM [Grønbæk 1993], HOSS [Nuernberg 1996], and HyperDisco [Wiil 1996].

For the purposes of the proposed research, we have selected Chimera to help support our prototype development efforts. Chimera's data model will serve as a means for capturing the state of a website's link structures; this state can then be used to detect changes to those structures over time. Furthermore, Chimera's ability to support link traversal over many different types of applications will let the EventTrails browser link the visual representation of document transaction notifications to the actual documents they reference. This functionality will allow the notifications to serve as an index into an organization's information space. Finally, Chimera was developed by the principal investigator and therefore we are intimately familiar with its characteristics and services, and thus we will incur no "learning costs" in applying it to the proposed research.

11.2 Publish/Subscribe Communication

Publish/subscribe communication, as a general approach, is growing in importance and popularity. Several publish/subscribe communication services have appeared during the last ten years or so, including commercial products such as SoftWired's iBus, TIBCO's TIB/Rendezvous, Talarian's SmartSockets, Hewlett-Packard's E-speak, and the messaging system in Vitria's BusinessWare. Researchers have also provided innovative systems such as Gryphon [Aguilera et al. 1999; Banavar et al. 1999], Elvin [Segall, Arnold 1997], JEDI [Cugola, Di Nitto, and Fuggetta 2001], Keryx [Wray and Hawkes 1998], and Herald [Cabrera, Jones, and Theimer 1999].

These systems differ quite markedly in how they provide the publish/subscribe service. For example, some work by having clients subscribe for messages by attaching themselves to an identified stream of messages ("channel-based" subscription), by posing a predicate over a required field of all messages ("subject-based" subscription), or by posing a predicate over all fields of a message ("content-based" subscription). The last of these is clearly the most powerful, and is the approach taken by Siena, as well as Elvin, Keryx, and Gryphon.

Attention to publish/subscribe middleware is growing with the appearance of the CORBA Notification Service [OMG 1999] and the Java Message Service [Sun 1999]. It is important to note, however, that both these efforts do not themselves realize a communication service. Rather, they simply specify interfaces to be implemented by a service, with the critical issue of how scalability is to be achieved left unaddressed. Nevertheless, they indicate the broad interest in this style of communication, which in turn further motivates the need for an appropriate network infrastructure.

Most existing publish/subscribe middleware is built around a centralized message dispatcher that registers all subscriptions and processes every message. This solution is conceptually simple and allows for various optimizations in the matching function, however it cannot scale to large, wide-area, high-demand applications. A few of the systems offer better scalability by using multiple interconnected dispatchers, as in Siena. However, all these systems exhibit limitations in the allowed interconnection topology. In particular, some require acyclic topologies, while others use hierarchical topologies that result from a simple extension of a centralized dispatcher. These constraints on the topology may not be problematic *per se*, but it should be recognized that they do impose genuine limitations in the way that the dispatchers can be dynamically configured and evolved.

11.3 Microsoft Information Rights Management System

Recently [Microsoft 2003], Microsoft has proposed an architecture for managing access to enterprise information. They refer to it as the Rights Management System (RMS). RMS involves a centralized repository of access rights information. Every application that wishes to access a document also accesses the repository in order to validate the access to that document.

By comparison, our approach is substantially more heterogeneous and flexible than the RMS approach. We do not require a central repository. Further, our approach has significantly more focus on the analysis side in that we can easily add new analyzers, and we can track historical accesses. It does not appear that either capability is currently available or even planned for RMS. Nor does it appear that the operating system participates in RMS. It appears to be purely an access control enforcement mechanism. It does have the advantage that Microsoft can provide a deeper integration of their own products into RMS.

12 References

[Aguilera et al. 1999]
Aguilera, M.K., R.E. Strom, D.C. Sturman, M. Astley, and T.D. Chandra. Matching Events in a Content-based Subscription System. Proc. of the Eighteenth ACM Symposium on Principles of Distributed Computing, pp. 53-61, Atlanta, Georgia, May 1999.

[Anderson 1994]
Anderson, K. M., R.N. Taylor, and E.J. Whitehead Jr. Chimera: Hypertext for Heterogeneous Software Environments. Proc. of the Sixth ACM Conference on Hypertext, pp. 94-107. Edinburgh, Scotland. September 18-23, 1994.

[Anderson 1997]
Anderson, K. M. Integrating Open Hypermedia Systems with the World Wide Web. In Proceedings of the Eighth ACM Conference on Hypertext, pp. 157-166. Southampton, UK. April 6-11, 1997.

[Anderson 1999a]
Anderson, K. M. Data Scalability in Open Hypermedia Systems. Proc. of the Tenth ACM Conference on Hypertext, pp. 27-36. Darmstadt, Germany. February 21-25, 1999.

[Anderson 1999b]
Anderson, K. M. Supporting Industrial Hyperwebs: Lessons in Scalability. Proc. of the 21st Int'l Conference on Software Engineering, pp. 573-582. Los Angeles, CA, USA. May 16-22, 1999.

[Anderson 2000a]
Anderson, K. M., R.N. Taylor, and E.J. Whitehead Jr. Chimera: Hypermedia for Heterogeneous Software Development Environments. ACM Transactions on Information Systems, 18(3): 211-245.

[Anderson 2000b]
Anderson, K. M., C. Och, R. King, and R.M. Osborne. Integrating Infrastructure: Enabling Large-Scale Client Integration. Proc. of the Eleventh ACM Conference on Hypertext, pp. 57-66. San Antonio, TX, USA.

[Anderson 2002]

Anderson, K. M., S.A. Sherba, and W.V. Lepthien. Towards Large-Scale Information Integration. Proc. of the 24th Int'l Conference on Software Engineering, pages 524-535, Orlando, FL, USA.

[Banavar et al. 1999]
Banavar, G., T.D. Chandra, B. Mukherjee, J. Nagarajarao, R.E. Strom, and D.C. Sturman. An Efficient Multicast Protocol for Content-Based Publish-Subscribe Systems. Proc. of the 19th IEEE Int'l Conference on Distributed Computing Systems, pp. 262-272, Austin, Texas May 1999.

[Berners-Lee 1996]
Berners-Lee, T. WWW: Past, Present, and Future. Computer, 29(10): 69-77.

[Cabrera, Jones, and Theimer 1999]
Cabrera, L.F., M.B. Jones, and M. Theimer. Herald: Achieving a Global Event Notification Service. Proc. of the Eighth Workshop on Hot Topics in Operating Systems, pp. 87-94, May 1999.

[Caporuscio, Carzaniga, Wolf 2003]
Caporuscio, M., A. Carzaniga, and A.L. Wolf. Design and Evaluation of a Support Service for Mobile, Wireless Publish/Subscribe Applications. Department of Computer Science Technical Report CU-CS-944-03, University of Colorado, January 2003.

[Carzaniga, Rosenblum, Wolf 2001a]
Carzaniga, A., D.S. Rosenblum, and A.L. Wolf. Design and Evaluation of a Wide-Area Event Notification Service. ACM Transactions on Computer Systems 19(3):332-383 (August 2001).

[Carzaniga, Wolf 2001b]
Carzaniga, A., A.L. Wolf. Content-based Networking: A New Communication Infrastructure. NSF Workshop on an Infrastructure for Mobile and Wireless Systems, Lecture Notes in Computer Science No. 2538, pp. 59-68, October 2001. Springer-Verlag (pub).

[Carzaniga, Wolf 2002]
Carzaniga, A. and A.L. Wolf. Fast Forwarding for Content-Based Networking. Department of Computer Science Technical Report CU-CS-922-01, University of Colorado, Revised September 2002.

[Cugola, Di Nitto, and Fuggetta 2001]
Cugola, G., E. Di Nitto, and A. Fuggetta, The JEDI Event-Based Infrastructure and its Application to the Development of the OPSS WFMS. IEEE Transactions on Software Engineering 27(9): 827-850 (September 2001).

[Davis 1994]
Davis, H. C., Knight, S., and Hall, W. Light Hypermedia Link Services: A Study of Third Party Application Integration. Proc. of the Sixth ACM Conference on Hypertext, pp. 41-50. Edinburgh, Scotland. September 18-23, 1994.

[Grønbæk 1993]
Grønbæk, K., Hem, J. A., Madsen, O. L., and Sloth, L. Designing Dexter-based Cooperative Hypermedia Systems. Proc. of the Fifth ACM Conference on Hypertext, pp. 25-38. Seattle, Washington, USA. November 14-18, 1993.

[Hall 1996]
Hall, W., Davis, H., and Hutchings, G. Rethinking Hypermedia: The Microcosm Approach. Kluwer Academic Publishers, Norwell, MA, USA.

[Heimbigner 2001]
Heimbigner, D. Adapting Publish/Subscribe Middleware to Achieve Gnutella-like Functionality. Proc. of the 2001 ACM Symposium on Applied Computing (SAC 2001). Las Vegas, Nevada, March 2001.

[Heimbigner 2003]
Heimbigner, D. An Efficient Implementation of Query/Advertise. Department of Computer Science Technical Report CU-CS-948-03, University of Colorado, March 2003.

[Knight, et al 2001a]
Knight, J., D. Heimbigner, A. Wolf, A. Carzaniga, J. Hill, and P. Devanbu. The Willow Survivability Architecture. Proc. of the Fourth Information Survivability Workshop, Vancouver, B.C, March 2001.

[Knight, et al 2001b]
Knight, J., D. Heimbigner, A.L. Wolf, A. Carzaniga, J. Hill, P. Devanbu, M. Gertz. The Willow Architecture: Comprehensive Survivability for Large-Scale Distributed Applications. Department of Computer Science Technical Report CU-CS-926-01, University of Colorado, December 2001.

[Microsoft 2003]
Microsoft Corporation. Microsoft Rights Management Solutions for the Enterprise: Persistent Policy Expression and Enforcement for Digital Information. February 20, 2003.

[Nuernberg 1996]
Nuernberg, P. J., Leggett, J. J., Schneider, E. R., and Schnase, J. L. Hypermedia Operating Systems: A New Paradigm for Computing. Proc. of the Seventh ACM Conference on Hypertext, pp. 194-202. Washington DC, USA.

[OMG 1999]
Object Management Group. Notification Service. August, 1999.

[Østerbye 1996]
Østerbye, K., and Wiil, U. K. The Flag Taxonomy of Open Hypermedia Systems. Proc. of the Seventh ACM Conference on Hypertext, pp. 129-139. Washington DC, USA. March 16-20, 1996.

[Pearl 1989]
Pearl, A. Sun's Link Service: A Protocol for Open Linking. Proc. of the Second ACM Conference on Hypertext, pp. 137-146. Pittsburgh, PA, USA. November 5-8, 1989.

[Rutherford et al]
Rutherford, M.J., K. Anderson, A. Carzaniga, D. Heimbigner, and A.L. Wolf. Reconfiguration in the Enterprise JavaBean Component Model. Proc. of IFIP/ACM Working Conference on Component Deployment, Berlin, Germany, June 2002.

[Segall, Arnold 1997]

Segall, B. and D. Arnold. Elvin Has Left the Building: A Publish/Subscribe Notification Service with Quenching. Proc. of the 1997 Australian UNIX Users Group, pp. 243-255, Brisbane, Australia, September 1997.

[Sun 1999]
Sun Microsystems, Inc. Java Message Service. Mountain View, California, November, 1999.

[USAF 2002]
Scientific Advisory Board. Building The Joint Battlespace Infosphere. Technical Report SAB-TR-99-02, U.S. Air Force, December 2000.

[Wang, et al 2002]
Wang, C., A. Carzaniga, D. Evans, A.L. Wolf. Security Issues and Requirements for Internet-Scale Publish-subscribe Systems. Proc. of the Thirty-Fifth Annual Hawaii Int'l Conference on System Sciences, Hawaii, Hawaii, January 2002.

[Whitehead 1997]
Whitehead, E. J., Jr. An Architectural Model for Application Integration in Open Hypermedia Environments. Proc. of the Eighth ACM Conference on Hypertext, pp. 1-12. Southampton, UK, April 6-11, 1997.

[Wiil 1996]
Wiil, U. K., and Leggett, J. J. The HyperDisco Approach to Open Hypermedia Systems. Proc. of the Seventh ACM Conference on Hypertext, pp. 140-148. Washington DC, USA. March 16-20, 1996.

[Wolf, et al 2000]
Wolf, A.L., D. Heimbigner, J. Knight, P. Devanbu, M. Gertz, and A. Carzaniga. Bend, Don't Break: Using Reconfiguration to Achieve Survivability. Proc. of the Third Information Survivability Workshop, Boston, Mass., October 2000.

[Wray and Hawkes 1998]
Wray, M. and R. Hawkes. Distributed Virtual Environments and VRML: An Event-Based Architecture. Proc. of the Seventh Int'l WWW Conference, Brisbane, Australia, 1998.