REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 11, 2005	Final Report

4. TITLE AND SUBTITLE
Ferret Workflow Anomaly Detection System Final Report

5. FUNDING NUMBERS
C NBCHC-030082

6. AUTHOR(S)
Timothy J Smith
Stephany Bryant

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
MCNC-RDI
P.O. Box 12889
RTP, NC 27709-2889

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

ARDA
9800 Savage Rd
RA Suite 6644
Ft Meade, MD 20755-6644

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official ARDA position, policy or decision, unless so designated by other documentation.

12 a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12 b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The Ferret workflow anomaly detection system project [2003-2004] has provided validation and anomaly detection in accredited workflows in secure knowledge management systems through the use of continuous, automated audits. A workflow, process, or procedure, is the set of steps that need to be completed to accomplish a goal. Anomaly detection is the determination that a condition departs from the expected.
The baseline behavior from which the anomaly is measured is usually derived via statistical sampling or through a reference specification. Ferret uses an accredited workflow specification to determine baseline behavior. An audit is an independent review of records and activities to assess the adequacy of system controls, to ensure compliance with established policies and procedures.

Ferret has attempted to address three key security problems in complex secure systems. First, Ferret has placed a single audit event into the larger workflow context in which it occurs, and has tracked the progress of the workflow to completion. Second, Ferret has provided a mid-level security policy language that fills some of the gap between high level, human language oriented and low-level computer oriented policies. Lastly, Ferret has attempted to mitigate the insider threat, that is, activity from authorized users who have abused their legitimate authority, through the corroboration of audit events.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Computer Network Security, Insider Threat	21
	16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT	18. SECURITY CLASSIFICATION ON THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Ferret
Workflow Anomaly Detection System

Final Report

February 28, 2005

Issued by ARDA under:

Contract Number	**NBCHC-030082**
Prime contractor:	**MCNC-RDI**
Subcontractor:	**MCNC**
CAGE Code:	**3BFW5**
DUNS/CEC Number	**11-872-8505**
TIN Number:	**01-0702442**

Technical Contact :
T. J. Smith
MCNC-RDI
P.O. Box 12889
RTP, NC 27709-2889
Phone: (919) 248-1852
Fax: (919) 248-1455
tjsmith@mcnc.org

Administrative Contact:
Alicia Brown
MCNC-RDI
P.O. Box 12889
RTP, NC 27709-2889
Phone: (919) 248-9217
Fax: (919) 248-1455
durham@mcnc.org

Contents

List of Figures

1 Executive Summary

The Ferret workflow anomaly detection system project [2003-2004] has provided validation and anomaly detection in accredited workflows in secure knowledge management systems through the use of continuous, automated audits. A workflow, process, or procedure, is the set of steps that need to be completed to accomplish a goal. Anomaly detection is the determination that a condition departs from the expected. The baseline behavior from which the anomaly is measured is usually derived via statistical sampling or through a reference specification. Ferret uses an accredited workflow specification to determine baseline behavior. An audit is an independent review of records and activities to assess the adequacy of system controls, to ensure compliance with established policies and procedures.

Ferret has attempted to address three key security problems in complex secure systems. First, Ferret has placed a single audit event into the larger workflow context in which it occurs, and has tracked the progress of the workflow to completion. Second, Ferret has provided a mid-level security policy language that fills some of the gap between high level, human language oriented and low-level computer oriented policies. Lastly, Ferret has attempted to mitigate the insider threat, that is, activity from authorized users who have abused their legitimate authority, through the corroboration of audit events.

Understanding and characterizing authorized users who abuse their legitimate authority has been a key problem with insider threat. One major threat from insider attack has stemmed from the current state of practice that has employed overbroad access permission methods for most situations and has had isolated points of policy enforcement. In sensitive environments, the current security practice of granting access to information has been based on satisfying three key conditions: need to know, is the person authorized and need to have the information to complete tasks; least privilege, is the information within the person's minimum sphere of access rights to enable task completion; and separation of duty, is the person's responsibilities scoped to preclude unmonitored activities. These three security properties will limit an organization's exposure to inside attack.

The Ferret system has offered a rich set of features incorporating various technologies that allow users to identify and track anomalies in authorized distributed workflows within a secure environment. The main tasks of Ferret have been: event, workflow, and anomaly representation and visualization; event collection, ordering, and correlation; and workflow analysis for anomaly detection. The system will be dynamic, allowing changes in workflow and usage policy in near real time, enabling the system to adapt to current conditions. Since the Ferret system is passive, only taking the audit output of systems, it can be easily adapted to existing systems.

2 Introduction

The Ferret workflow anomaly detection system project [2003-2004] has provided validation and anomaly detection in accredited workflows in secure knowledge management systems through the use of continuous, automated audits. A workflow [13], process, or procedure, is the set of steps that need to be completed to accomplish a goal. Anomaly detection is the determination that a condition departs from the expected. The baseline behavior from which the anomaly is measured is usually derived via statistical sampling or through a reference specification. Ferret uses an accredited workflow specification to determine baseline behavior. An audit is an independent review of records and activities to assess the adequacy of system controls, to ensure compliance with established policies and procedures.

Ferret has attempted to address three key security problems in complex secure systems. First, Ferret has placed a single audit event into the larger workflow context in which it occurs, and has tracked the progress of the workflow to completion. Second, Ferret has provided a mid-level security policy language that fills some of the gap between high level, human language oriented and low-level computer oriented policies. Lastly, Ferret has attempted to mitigate the insider threat, that is, activity from authorized users who have abused their legitimate authority, through the corroboration of audit events.

The Ferret system has offered a rich set of features incorporating various technologies that allow users to identify and track anomalies in authorized distributed workflows within a secure environment. The main tasks of Ferret have been: event, workflow, and anomaly representation and visualization; event collection, ordering, and correlation; and workflow analysis for anomaly detection. The system will be dynamic, allowing changes in workflow and usage policy in near real time, enabling the system to adapt to current conditions. Since the Ferret system is passive, only taking the audit output of systems, it can be easily adapted to existing systems.

2.1 Insider Threat

Understanding and characterizing authorized users who abuse their legitimate authority has been a key problem with insider threat. One major threat from insider attack has stemmed from the current state of practice that has employed overbroad access permission methods for most situations and has had isolated points of policy enforcement. In sensitive environments, the current security practice of granting access to information has been based on satisfying three key conditions: need to know, is the person authorized and need to have the information to complete tasks; least privilege, is the information within the person's minimum sphere of access rights to enable task completion; and separation of duty, is the person's responsibilities scoped to preclude unmonitored activities. These three security properties will limit an organization's exposure to inside attack.

Users have been given overbroad access permissions, usually to accommodate exceptional situations, without regard to context, far greater than they really need to perform their typical job functions. This has undermined the least privilege security attribute. In addition, the host systems have had isolated enforcement points, which have not knowledge of a global security plans. The policy enforcement points may have contained security policies that are not strongly enforced because of their scope of knowledge. This lack of systemic policy enforcement has undermined the separation of duty security attribute. Insiders have exploited these holes in security by accessing information from various unrelated sources and have used that information in unintended ways that are outside of the scope of their current job responsibilities.

Effective security systems have balanced the need of people to get their jobs done with the mitigation of risk. One way has been to create security policies that encourage transparency for higher risk activities. Transparency has been achieved through the creation of audit trails for all high-risk activity. Risk has been mitigated with policies that have checks and balances. For example, separation of duty and oversight properties have been provided by involving at least two people in high-risk processes.

2.2 Ferret Solution

2.2.1 Novel and Related Aspects

Automated mining of audit logs for security purposes has been proposed by James Anderson's work in 1980 on automated batch analysis of host audit records to look for intrusions [1]. Ferret has recast audit analysis and correlation from a detective role into preventative, deterrent, compensative roles with active monitoring.

In the intrusion detection field, a general theory was developed [6], and two schools of thought have emerged: signature-based [18] and statistical-based [11]. Intrusion and anomaly detection systems have been categorized events into three buckets: normal, anomalous, and uncharacterized. Most work in the field has focused on strongly characterizing anomalous events from outsiders to produce security alerts.

In contrast, Ferret has focused on the insider attack threat, estimated to be the largest and hardest attack set to deter, since the attackers are authorized users of the system and frequently performing authorized functions. Ferret has focused on strongly characterizing normal events from insiders into workflows, providing a richer context for a single event in order to identify misuse.

Ferret has had a different temporal perspective, focusing on longer-term event correlation, on the order of days, weeks, or months. Ferret has had a higher level perspective semantically, having been able to attach a higher-level meaning to event sets through correlation to workflows.

Ferret has been oriented towards generic application level monitoring, and has been able to describe complex, arbitrary systems with the twenty patterns in an extremely expressive workflow model. The generic framework has been targeted towards a specific area such as packet, network flows, or firewalls through modular extensions.

Ferret has been able to address the gap between high level security policy and lower-level implementations through the use of flexible workflow audit language as mechanism to describe middle level procedures to implement higher-level security policy. This direct linkage between policy, procedures, and implementation has allowed for evaluation of policy coverage and effectiveness.

The second generation Ferret has been able to refine context access control model for just in time and in context capability enablement and disablement. The Ferret approach, a refinement of Clark-Wilson [5] integrity model, has stated: You can only access or modify information through applications within a certain workflow context, with separation of duty, and auditing.

2.2.2 Workflows Context

Determining the context of a process has been very difficult in multi-process and multiuser systems, such as knowledge management systems. Ferret has established workflow context and policy compliance of workflows through audit checklists of workflow artifacts [21] and corroboration of evidence from multiple sources. The checklists, known as workflow audit models (WAM), has been flexible specifications of the workflow artifacts and process policy. The audit models has been expressed as finite state machines (FSM) in the software. The finite state machines have tracked the collection of the audit artifacts, which mirror the progress of the workflow to completion.

Workflows has cover a wide variety of tasks and situations, so that no one single ontology has been able to express them all. A particular workflow language has been able to effectively express a process within a certain domain. Likewise, the Ferret workflow audit model has been a general-purpose language, designed to be able to express general concepts, such as control flows and workflow patterns. The Ferret project has designed its WAM schema in XML, allowing for extensions to the WAM at deployment time to provide necessary granularity for specific domains.

2.2.3 Security Policy

Effective security policy has been challenging to create; you must balance the sometimes conflicting security and operational goals of the organization. Policies that are rigid, or interfere with productivity, have been frequently subverted or eliminated. Administrative and supervisory positions frequently have had the ability to override security controls for exceptional situations. The ability to circumvent security controls has been abused. To counterbalance potential abuses,

audits of actions have been conduced to ensure prudent use of authority. Security policy has been typically distributed to administrative, physical, and technical domains for efficiency and robustness of administration and enforcement.

Ferret has provided a means for auditing interaction between systems for compliance with security policies. Using a flexible business process description language, we have been able to capture and encapsulate authorized workflows. During workflows, certain auditable milestones have been checked for compliance with authorized policy. Auditing has given security policy implementation a much greater flexibility.

Ferret has had the ability to combine multiple behavior models for the analysis. Security model, business process, organizational structures, and audit artifacts have been combined for assessment. Combinations of different information sources have provided the means for corroboration of information in distributed systems.

Auditing has worked in concert with context-based access control systems. For example in a context-based access control system, a party could have claimed a certain context; how would the access control system, or other parties verify that claim? That claim of trust in a context has sometimes been predicated on the trust of the host operating system. The host operating system is vulnerable to authorized users, especially in the insider threat scenario.

3 Underlying Technologies

3.1 Java Technologies

The Java programming language, platform, and its accompanying technologies [14] are the main implementation mechanisms for the Ferret prototype system. The Ferret prototype is implemented in the Java programming language taking advantage of its platform independent capabilities. The visualization portion of the Ferret project was implemented using a combination of Java objects and Java server pages (JSP) to generate dynamic web page content to enable viewing Ferret analysis results from any where on the encompassing network using existing browser technologies.

3.2 Extensible Markup Language (XML)

Extensible Markup Language (XML) is a general purpose markup language that may used to express the structure and contents of information in a self-identifying format. It is compatible with SGML, yet simpler to use and provides more expressive power that the fixed schema of HTML. XML is expressed in text format that is processed by a suitable XML parser that adheres to the established XML document processing standards. The XML file, like the HTML, expresses data in a tree format of arbitrary length and depth [10] consisting of element tags enclosed in angled brackets with required and optional tag attributes. XML validating processors typically strictly enforce the requirement that XML documents contain well-formed syntax. For example, each start tag should have a matching end tag or be an empty tag.

XML technologies encompass a set of complementary technologies that together provide the ability design and express and process the structure of information in flexible manner. The Ferret project uses of two XML technologies - the XML schema and the XML document. XML schema is an XML document that describes all allowable elements and their ordering of an implementing XML file. The XML file is an implementation of the XML schema that adheres to the constraints of the encompassing schema.

Ferret workflow audit language is an XML schema that describes the scope and elements expected in an workflow audit model. Each workflow audit model is an XML document that adheres to the workflow audit language. The model describes a specialized workflow process by expressing differing flow relationships between the allowed state elements along their expected audit information artifact(s), as well as different conditions and references to organization security policies.

3.3 Castor

The Castor project [4] provides facilities for translations between XML entities and Java classes. The application uses Java reflection capabilities to dynamically create Java objects based on the parsed XML information in any workflow audit model text file. The Ferret project uses this technologies to translate the Workflow Audit Model representations in XML into Java objects during analysis phase and translating Java objects into the proper information to be stored in the repository databases at the conclusion of processing.

3.4 Extensible Access Control Markup Language (XACML)

Extensible Access Control Markup Language (XACML) is an OASIS standard that defines a XML schema for expressing authorization policies in XML. "XACML enables the use of arbitrary attributes in policies, role-based access control, security labels, time/date-based policies, indexable policies, 'deny' policies, and dynamic policies" [16]. Ferret uses the XACML schema to express dynamic security policies. The Sun reference implementation of the XACML evaluation engine is used to process dynamic access policies to ensure that audit information attributes are policy compliant when the audit event occurred.

4 Ferret Implementation

4.1 Background

The Ferret system allows users to perform authorized work processes while mitigating the insider threat problem. It analyzes the audit artifacts generated by the workflow process detecting any diversions from expected artifacts and generates report information of possible intrusion or abuse. The system uses a multilayer approach to identify probable insider attack. The system relies on a set of existing organization approved applications, which are adequately annotated to provide useful audit data artifacts for analysis, and executing with in the organizations private network.

This section describes the implementation of the Ferret System prototype. It describes the general architecture, the Workflow Audit Description Language developed as part of Ferret and the Event mechanism that examines and reasons over the audit artifacts.

4.2 Ferret Architecture

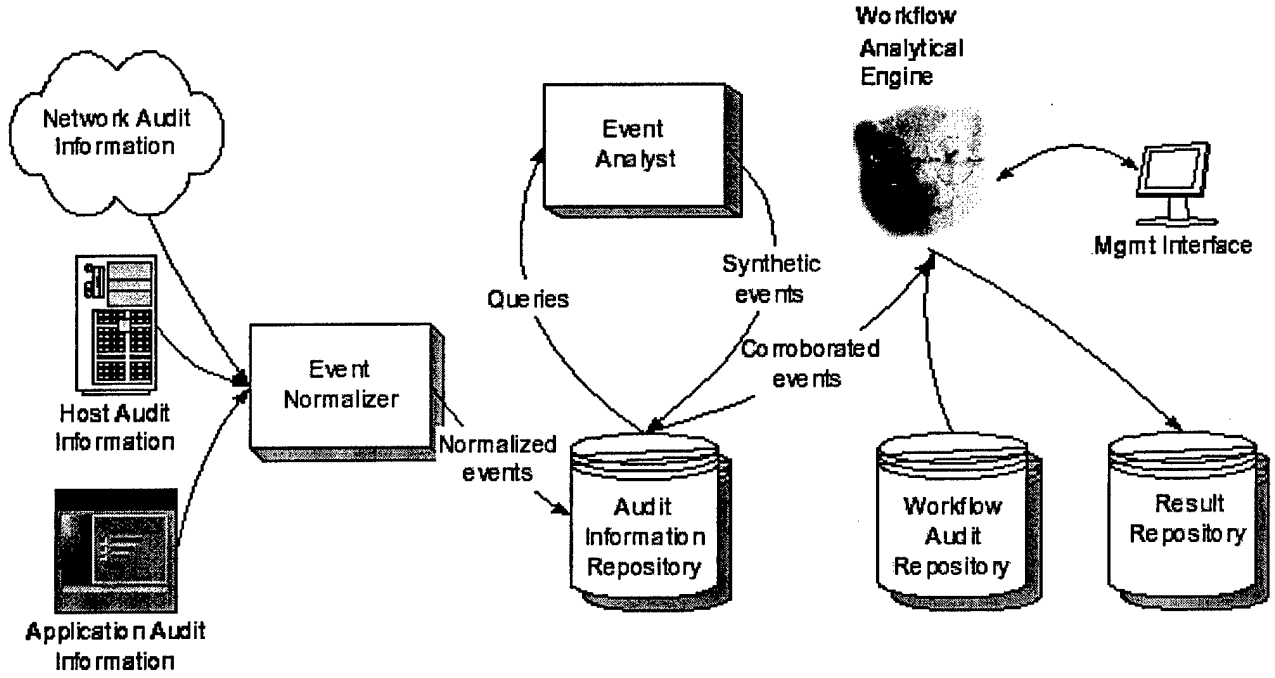

Figure 1: Ferret Architecture

The Ferret design (Figure 1) is typical enterprise anomaly detection system architecture. The Ferret prototype system is implemented as a Java-based framework. Ferret accomplishes its anomaly detection through the use of several distributed modules, which are decoupled by a relational database. The distributed architecture with flexible input formats, pattern matching, and output formats is suitable for generalized procedural monitoring system. The event normalization system transforms events from their native format into a structured XML format that Ferret can process. The normalizer temporally orders incoming event from multiple sources. The normalizer isolates and decouples the remainder of Ferret from monitored systems.

The Gumshoe system provides event analysis, corroboration, and production of inferred synthetic events. In order to handle hosts and applications compromised or misused by insiders, Gumshoe will corroborate events from different hosts

and applications to produce synthetic events to increase the confidence of the validity of information collected. Gumshoe can corroborate information from other collected events or from active interrogation of information sources.

The Ferret workflow analytical engine has three components: a workflow finite state machine (WFSM), a security policy validator, and an arbitrary condition checking system. The WFSM tracks the progress of the workflow via finite state machines from beginning to completion. WFSM uses recursive FSMs to handles loops, split, and join patterns. The WFSM gives Ferret the flexibility to handle twenty workflow patterns. The WFSM are loaded from accredited workflows specified in the Workflow audit modeling (WAM) language. Ferret checks the workflow events against a XACML security policy to validate compliance. This verifies that discretionary access control mechanisms in applications or hosts work as expected. Ferret can be extended for checking that arbitrary conditions are met during the execution of a workflow. This is where Ferret can be extended for validation of domain specific conditions in workflows.

4.3 Ferret Audit Description Language

Workflows cover a wide variety of tasks and situations, so that no one single ontology is able to express them all. A particular workflow language is able to effectively express a process within a certain domain. Likewise, the Ferret workflow audit model is a general-purpose language, designed to be able to express general concepts, such as control flows and workflow patterns. The Ferret project designed a Workflow Audit Model Language (WAM) schema in XML, allowing for extensions to the WAM at deployment time to provide necessary granularity for specific domains, such as specification of external references to information such as organization policies that are expressed using XACML.

WAM is composed of three major categories: data/structure types, flow control elements, and conditions. The workflow audit model language (WAM) is derived from the Extensible Markup Language [20] schema (XML-Schema). The XMLSchema foundation provides extensible, flexible, strongly typed data types and structures. Consequently WAM provides an extensive set of nineteen primitive data types, such as boolean, string, integer, and float; and arbitrary derived types. WAM supports polymorphic values with union data types. WAM also provides the notion of an arbitrary aggregation or collection data types.

Workflows can be decomposed into twenty generic activity patterns, categorized as: sequence, loop, branch, merge, synchronization, and cancellation. WAM flow control elements are derived from the Business Process Execution Language for Web Services (BPEL4WS)[3], which providing a rich control set and familiar naming to enterprise workflow analyst.

Ferret provides a schema compiler for the WAM language and derivatives, and a code generator for the java classes necessary to incorporate extensions. Ferret provides an extensible run-time validating parser for transforming the workflow audit model instances into runtime finite state machines instances. Many anomaly detection systems have the procedure or protocol logic embedded into specialized software or firmware logic, typically increasing processing speed at the expense of flexibility. Ferret uses a general-purpose engine to analyze the process specifications encoded in WAM. WAM schema enables the encoding of very complex patterns, using all twenty of the common workflow patterns. This flexibility provides one of the key differentiators of Ferret to many other detection systems, the ability to handle events from OSI physical layer (1) through the stack to applications layer (seven). It also enables Ferret to be adapted to the policies and procedures of your organization at this time and in the future. One of the drawbacks for specification-based anomaly detectors is the expertise needed to encode and debug the specification.

4.4 Event Processing

Following Event Normalization, event processing applies the normalized audit events to finite state machines, constructed from workflow audit model descriptions. The audit events are applied to the finite state machine in time order. If the application of the audit events cause the FSM to reach its end state, then the audit events represents a successfully matched and completed workflow. Failure to move the FSM to an end state may indicate that all of the audit events for the workflow are not present in the database yet or may indicate some attempts to circumvent the workflow. In any cases were workflow FSMs fail to reach end state, more cycles of event processing may be needed and/or further investigation by an operated may be required. Figure 2 shows the Ferret event processing strategy.

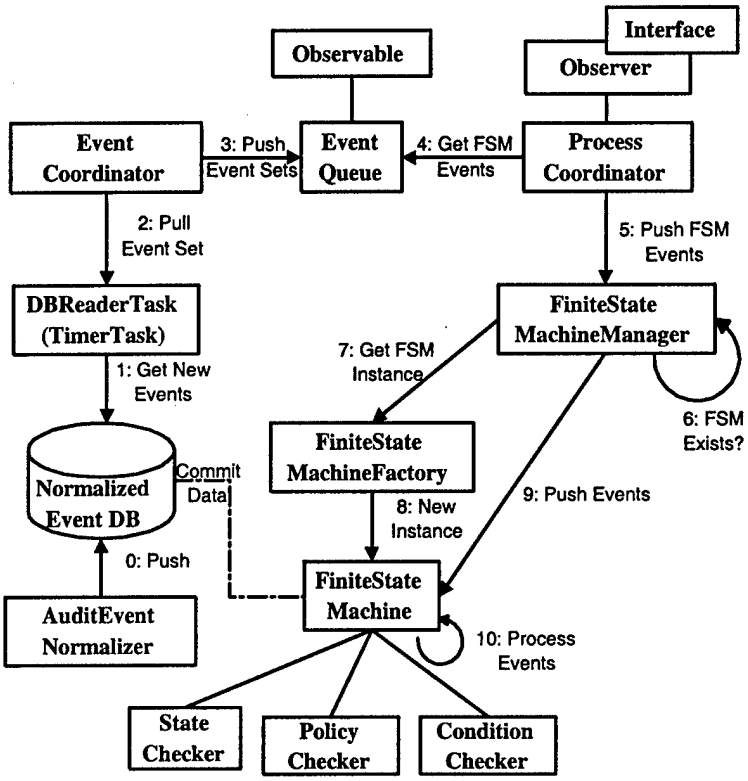

Figure 2: Normalized Audit Event Processing

Event processing begins any time after normalized audit events are committed to the Normalized Event Database. The EventCoordinator retrieves the normalized audit events from the database using the DBReaderTask, which connects to the database and performs the actual data retrieval. Events retrieved from the database are stored in the EventQueue, which sorts the events according to workflow name and ID fields. The EventQueue extends the Observable class so that registered Observer objects can be notified when the internal state of the EventQueue changes, such as when new events are added to the queue.

The ProcessCoordinator coordinates the transfer of audit events from the EventQueue to the various executing FSMs. This coordinator is a registered Observer of the Event Queue and is notified by EventQueue when new audit events are available. Once notified by the EventQueue, the ProcessCoordinator retrieves the set of audit events associated with each workflow name and id pair.

The ProcessCoordinator passes each event set to the FiniteStateMachineManager, which manages all executing FSM instances. The manager determines whether a FSM instance exists for the workflow name and id pair associated with the event set. If a FSM does not exists, the FiniteStateMachineManager directs the FiniteStateMachineFactory to create a FiniteStateMachine based on the workflow name, which is mapped to a Workflow Audit Model Description file, and set its id. Then the FiniteStateMachineManager passes the event set to the appropriate FSM instance for processing.

Beginning at its start state, the FSM applies the each member of the time-ordered normalized event set to it current state and attempts to transition to its next state. Transitioning to the next state occurs if the information contained in the audit event when applied to the FSM state context passes all the applicable checking testing. Audit events are processing this manner until state transition fails, due to checking failure, or the FSM reaches its end state. If the FSM can not transition

to its next state, then it waits for a new set of audit events. When the FSM reaches its end state, it writes the updated audit events, containing the match results, back to the Normalized Event Database or another database repository for report generation and analyst viewing and investigation.

During event processing, the Ferret implementation allows for the specification of three types of checking for normalized audit event information. The existing types of checking are state checking, policy checking, and condition checking. The actual checking that occurs depends on the information present in workflow audit model description for the state. Discussion of each type of checking follows.

4.4.1 State Checking

State checking compares the values of the normalized audit event attributes to the expected attributes, excluding the checking results field, and values in the workflow audit model description. The attributes are arbitrary strings and values based on the workflow audit description. Ferret uses the reflection capabilities of the Java programming language to extract each attribute and value pair so that it can be compared with same attribute and value pair from the normalized audit record.

The result field is excluded from comparison because it always specified as an empty string in the workflow audit model. The field is populated during the checking. So the value will not match what is specified in the audit model which is a blank string.

Failure to successfully pass state checking directly prevents the FSM from moving to the next FSM state.

4.4.2 Policy Checking

Policy Checking allows an organization's policies to be referenced in a workflow audit description. Policy Checking information is expressed and evaluated in the same manner as other conditions, see below. However, its implementation includes the instantiation of a XACML policy decision mechanism which accesses the policy and compares the appropriate state or event attributes and values to the policy's specified properties and values. The resulting decision is translated to a boolean value, true or false, and that result is returned as the result of the policy checking.

4.4.3 Condition Checking

The third aspect of checking consists of checking conditions associated with entire workflow descriptions or individual expected audit events. The conditions are expressed according to the Composite design pattern[8], which allows all conditions to be expressed and evaluated in a generic manner. In addition, complex conditions can be expressed in tree hierarchy of arbitrary levels and evaluated to one result. Condition evaluation returns a boolean value indicating whether the expressed condition evaluated to true or false. The result of the condition evaluation is saved as part of the state evaluation and is included in the information report for the state.

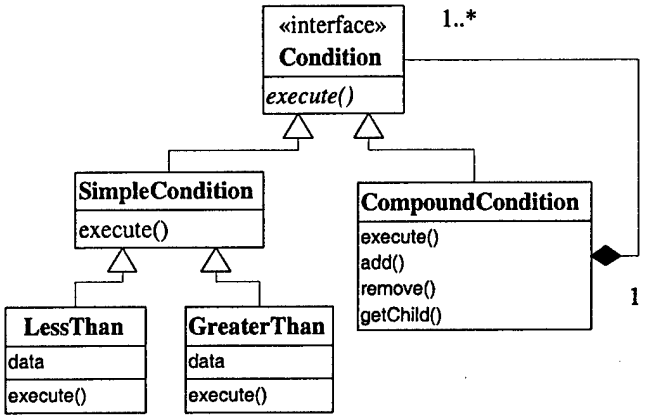

Figure 3: Ferret Condition Class Hierarchy

Figure 3 shows the class hierarchy implementation for Ferret conditions. All conditions inherit from and abstract "Condition" interface that contains the execute() method, whose implementation is defined in a derived class. The "CompoundCondition" class represents a set of conditions that are to be treated as a single grouping. This allows multipart conditions to be evaluated and return a single result. The "SimpleCondition" is a single condition, which contains the information to be used to evaluate the condition as well as the implementation specific directives to perform the condition evaluation. Ferret contains two derived "SimpleCondition" derived classes - LessThan and GreaterThan. In this implementation, these derived classes can be used individually or together in order to express a range condition. For example, a time value can be expressed as a range condition containing a lower bound value and an upper bound value.

4.5 Ferret Visualization

The Ferret allows users to view the results of the workflow analysis in a Web browser. The visualization application is a Java web application that consists of a set of Java server pages and Java language classes that dynamically generate HTML pages to display workflow and analysis information. The application examines the information collected during the analysis and corroboration phases, displaying the information in the form of hyperlink text and graphical depictions of workflows. Figure 4 shows an sample of the display information.

The hyperlinked text display represents the states of a workflow audit model, including the conditions associated with each state. Selecting any hypertext link, representing a summary of the state information, displays the detailed state information. Selecting the "hide" hyperlink displays the state summary information without the detailed state information. Selecting any "Merged flow" state displays hyperlinks to its child(ren) workflow(s). Selecting a child workflow link displays the states contained in that child workflow; the child state hyperlinks work the same the hyperlinks for the parent FSM states as described above. The hypertext workflow display is capable of display information for workflow of arbitrary length and depth, enabling the display of simple as well as complex workflows.

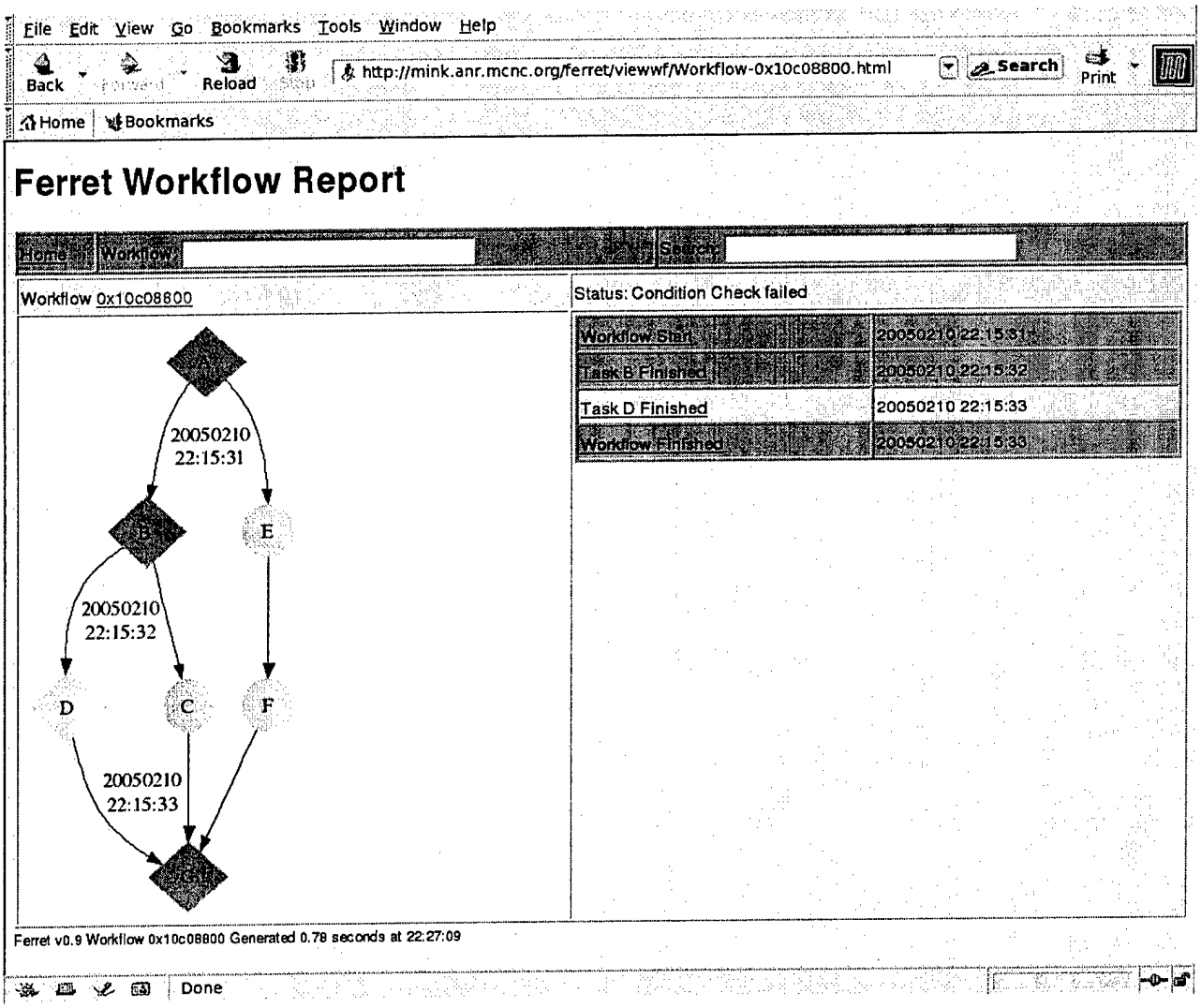

Figure 4: Workflow detail information

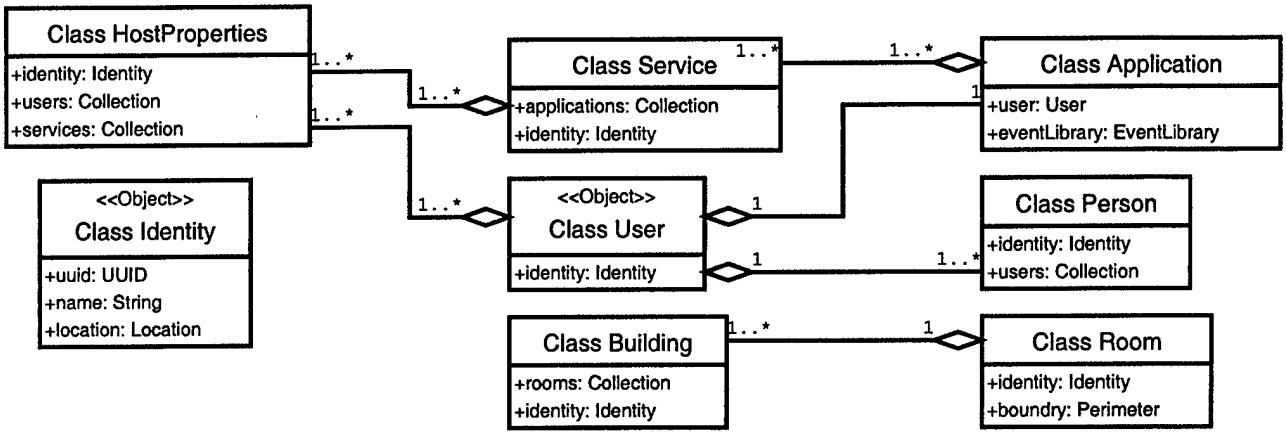

Figure 5: Simplified Enterprise Computer Domain

5 Case Studies and Testing

The cases presented were chosen to illustrate the effectiveness of specification-based anomaly detection of malicous insider using the Ferret framework on existing computer systems and procedures. Alternative solutions do or could exist, however they are typically intrusive to users or envasive to the procedure or system. Ferret is presented as a simple means of acheiving our goal. Human interaction with the test systems was simulated because of the cost and complexity of compliance with human subject testing and privacy laws and regulations.

5.1 Ferret Application Domain

Ferret is a general framework for the tracking and validation of workflows. To apply ferret to a specific domain a knowledge base must be constructed. The actors, actions, objects and relationships of the domain must be identified and characterized as in Figure 5.

5.2 Electronic Mail Transmission Validation

5.2.1 Phishing Scenerio

One of the critical elements of enterprise processes is electronic mail. Ensuring that email sending and delivery integrity by validation of established procedures is vital to enterprise security. "Phishing", that is the sending of authentic appearing email to users to solicit sensitive information, is one of the most successful and widespread social engineering techniques yet developed to date. The phishing technique can be even more effective when used by a trusted insider, because of access to resources and their intimate knowledge of specific people and processes.

5.2.2 Testing Environment

The testing environment utilizes a cluster of five machines: *mustela, mink, ermine, stoat,* and *weasel* to simulate three large domains: *alpha.dom, beta.dom,* and *delta.dom.* Each domain has one thousand simulated users, sending and receiving electronic mail messages. The users are simulated via unix shell scripts. The postfix server was used for the sending and delivery of the SMTP messages. There are three scenarios with different infrastructure configurations ranging from easy to difficult to spoof phishing messages to illustrate how ferret could contribute to the security of system and detect malicious insider activity.

The first scenario involves the default or typical configuration, postfix servers configured to send SMTP messages in the clear between domains. The servers are configured to reject SMTP relay requests, so that only email originating from the SMTP server of that domain, as configuted by DNS, will be received.

The first scenario has several attacks. The first attack is the easiest, a valid domain user (*user1@alpha.dom*) composes, and sends email proportedly from another user in the same domain (*user2@alpha.dom*) to a third user in the same domain (*user3@alpha.dom*).

```
[user1@weasel]$ cat fake.txt
From: User Two <user2>
To: User Three <user3>
Subject: Faked Mail testing

This is the body of the email message.
[user1@weasel]$ cat fake.txt | /usr/lib/sendmail -t
```

The phishing attack can even be effective in situations where strong cryptographic techiniques are used for document integrity and authentication. For example, the Secure / Multipurpose Internet Mail Extension (S/MIME) is a security enhancement to the MIME e-mail format standard. S/MIME only protects certain portions of the body of the message and not the header, which contains the *To*, *From* and *Subject* fields of the RFC-822 SMTP message. A variation of the first attack can be used, where false *From* field information can be used in the header. The attacker can than attach multiple signed sections to an email. The first signed section being from the supposed sender of the email. The remaining sections will also be validly signed by scapegoated users and contain the phishing information. Mail user agents evaluate all the signed sections of the email and possibly display the identity of the signer, typically from the first section.

One scenario is the attacker assumes the victims identity and appends spoofed email messages directly to the victims email store. The emails could have completly authentic header information from the supposed sender. There would be no SMTP log trail of sending or receiving to review for irregularities.

5.2.3 Results

A set of direct results and collateral information was gained during the experiment. Ferret tracked the sending and delivery of SMTP messages from the postfix server logs from the Unix syslog facility. Two workflows were established, one from local delivery of email, and the second for delivery of email to another SMTP host.

```
1.1 Feb 17 13:39:44 mink postfix/pickup[28550]:
38D4E60864: uid=501 from=<user1>
1.2 Feb 17 13:39:44 mink postfix/cleanup[28661]:
38D4E60864: message-id=<20050217183944.38D4E60864@mink.anr.mcnc.org>
1.3 Feb 17 13:39:44 mink postfix/qmgr[3182]:
38D4E60864: from=<user1@mink.anr.mcnc.org>,
size=319, nrcpt=1 (queue active)
1.4 Feb 17 13:39:44 mink postfix/smtp[28663]:
38D4E60864:   to=<user3@weasel.anr.mcnc.org>,
relay=weasel.anr.mcnc.org[152.45.4.48],
delay=0, status=sent (250 Ok: queued as B350F1447B)
1.5 Feb 17 13:39:44 mink postfix/qmgr[3182]:
38D4E60864: removed
```

The listing (1) above shows the Unix system log from mink.anr.mcnc.org for the remote delivery of email from *user1@mink.anr.mcnc.org* to *user3@weasel.anr.mcnc.org*. Ferret normalizes the system logs from both hosts, and corroborates a number of pieces of information. Element **1.1** from the postfix/pickup command informs the system that a new email workflow numbered 38D4E60864 has begun from user1 on host mink. The user1 label and uid 501 association is confirmed. Element **1.2** contains the email message id, and is linked to the workflow via the 38D4E60864 identifier. Element **1.3** enqueues the message for delivery and contains the full from email address, and is linked to the workflow via the 38D4E60864 identifier. Element **1.4** records the delivery of the SMTP to weasel.anr.mcnc.org and contains the full receiptiants email address, and is linked to the workflow via the 38D4E60864 identifier and gives the weasel workflow identifier B350F1447B. Element **1.5** completes workflow 38D4E60864.

```
2.1 Feb 17 13:39:44 weasel postfix/smtpd[22975]:
connect from mink.anr.mcnc.org[152.45.4.100]
2.2 Feb 17 13:39:44 weasel postfix/smtpd[22975]:
B350F1447B: client=mink.anr.mcnc.org[152.45.4.100]
2.3 Feb 17 13:39:44 weasel postfix/cleanup[22978]:
B350F1447B: message-id=<20050217183944.38D4E60864@mink.anr.mcnc.org>
2.4 Feb 17 13:39:44 weasel postfix/smtpd[22975]:
disconnect from mink.anr.mcnc.org[152.45.4.100]
2.5 Feb 17 13:39:44 weasel postfix/local[22979]:
B350F1447B: to=<user3@weasel.anr.mcnc.org>, relay=local,
delay=0, status=sent (delivered to mailbox)
```

The listing (2) below shows the Unix system log from weasel.anr.mcnc.org for the same transaction and contains two workflows. The first workflow, elements **2.1** and **2.4** contains the SMTP connection from mink. The second workflow, elements **2.2**, **2.3**, and **2.5**, contain the delivery of the email message from *user1@mink.anr.mcnc.org* to *user3@weasel.anr.mcnc.org*.

5.3 Web Server Login Authentication

5.3.1 Validation of authentication information from several sources

One goal of strong enterprise security is the use of strong, positive authentication mechanisms for access control to senstive facilities, systems, and materials. To acheive this goal, Ferret could be utilized to integrate and validate authentication information from existing commercial and goverment off-the-shelf systems. One simple example of a security policy workflow is the requirement for a person to badge into a building and secure rooms prior to logging into a secure terminal in the building. To utilize Ferret a procedure would be established to encode the rules for integration of authentication information. The procedure would have three major steps: (1) use of Photo ID Smart badge to gain access into the sensitive facility, (2) use of Photo ID Smart badge to gain access into secure room, and (3) use of Photo ID Smart badge to log into secure terminal. Each step in the procedure would generate an audit event to be collected by Ferret from that system. These badge accesses would be recorded on a computer in the building security office. The user would then logon to their secure terminal with a smart card and PIN; this would be recorded in the hosts access log. The Ferret system would then take the audit output from the various access/entry control systems, create a flow context, and establish if the user followed the described security policy workflow. If the user logged into the system with a partial or empty context, an anomaly would be issued by Ferret. Ferret in this example couples information from unmodified legacy physical and computer access controls systems for a systemic validation of conformance of the users activities to the security policy. The smart badge itself could itself be a fourth audit source, if it contained an access log and could be queried. Ferret turns two or three factor into a composite n-factor authentication method with multiple independent corroboration points. The testing environment utilizes a cluster of five machines: *mustela, mink, ermine, stoat,* and *weasel* to simulate three large domains: *alpha.dom, beta.dom,* and *delta.dom*. Each domain has one thousand simulated users, accessing three web servers. The users are simulated via unix shell scripts. In each test, Ferret correctly identified the instances in which the login authentication workflow process was violated.

6 Related Work

6.1 Insider Threat Detection Techniques

Research in the area of insider threat detection has emerged as an important area of examination as the potential damage from incidents of inside threat continues to escalate. Investigation of insider threat [2] has classified insider threat detection techniques into a small set of general categories: develop profiling as a technique, detect misuse of applications, provide traceability for system objects, identifiy critical information automatically, and design systems for detectability. Various projects exist that have explored these themes in detecting insider threat. Nguyen [15] discover insider threat by examining the relationships between users and file system access, which are fairly regular. Through the examination of raw data, they have endeavored to develop generic models that detect large sets of attacks. Snapp [19] have developed "a prototype Distributed Intrusion Detection System (DIDS) that combines distributed monitoring and data reduction (through individual host and LAN monitors) with centralized data analysis (through the DIDS director) to monitor a heterogeneous network of computers". Marin[12] have developed a hybrid method that initially uses expert rule based system and then clustering mechanisms to classify system users based in the frequency and types of commands that they employ within a certain period of time.

Ferret has a combined approach to detecting insider threat. It has used process workflow models to describe proper use of a set of system objects. The system objects have reported their use through audit events which are collected from multiple sources and have been correlated to bolster evidence of correct use. However, deviation from the correct use of system objects and privileges has also been detected as by product of the examination of system information, which may not adhere to correct patterns of use. Those incidences of non-compliance use have been reported as indications of unintentional misuse or potential insider attack.

6.2 Workflow and Description Languages

Workflows have described the ordering of tasks and organizational components in a business process [7]. Process models have been used to describe routing techniques for the process include decision chain and event flow models. Decision chain model has used milestone and decision points to describe a business process in detail. The event flow model has described a business process as a connection of manual and automatic events involved in a business process.

A variety of workflow description languages has existed to express organization workflows. Business Process Execution Language for Web Services (BPEL4WS), developed jointly by BEA, IBM and Microsoft, is language for formal specification of business process and build interaction protocols. Business Process Modeling Language (BPML) has been a language for specifying the processes of an enterprise. BPML has been developed by BEA, Sun Microsystems, and SAP. XML Process Description Language (XPDL) governed by Workflow Management Coalitions (WfMC) facilitates process description interchange between workflow vendors. Wf-XML, also from WfMC, has defines the runtime interactions between process automation system to support process enactment across multiple domains. While these are just a few example of workflow description languages, other research projects, open source and commercial languages have existed. However, BPEL4WS has emerged as a widely accepted language that is capable of expressing business processes in a variety of domains.

6.3 Process Workflows as Finite State Machine

Process workflows have defined the activities, constraints, and supporting applications required of a business process without regard to its implementation. Event-based process workflows have been implemented as state transition machines, which map activities to states with associated constraints or conditions and express transitions between the states containing pre- and post conditions. Several research projects have existed that implement process workflows as state transition machines. Apache has developed Lenya [9], which is open source content management system with the capability to express workflows. Workflows in Lenya have been expressed as state transition machines described in XML text. OSWorkflow, a Open Symphony project, has been "a flexible workflow engine based on the finite state machine concept" [17]. The workflows are described in XML.

The Ferret prototype has implemented the process workflows as finite state machines derived from the XML representations of the process workflows defined by an XML schema. Each finite state machine instance has tracked the completion of each task that comprises the workflow instance by gathering and examining the audit artifacts for that ask. If the artifacts are as expected for the state, the FSM instance has transitioned to the next state or the end state if all states have been visited.

References

[1] J. P. Anderson. *Computer Security Threat Monitoring and Surveillance.* James P. Anderson Co., Fort Washington, Pennsylvania, April 1980.

[2] Robert H. Anderson. Research and Development Initiatives Focused on Preventing, Detecting, and Responding to Insider Misuse of Critical Defense Information Systems. Technical Report CF-151-OSD, Rand Corporation, 1999.

[3] Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, and Sanjiva Weerawarana. *Specification: Business Process Execution Language for Web Services Version 1.1*, May 2003.

[4] The Castor Project. http://castor.codehaus.org/.

[5] D. D. Clark and D. R. Wilson. A comparison of commercial and military computer security policies. In *Intrusion IEEE Symposium on Security and Privacy,*, pages 184–194, Oakland, April 1987.

[6] D. Denning. An intrusion detection model. *IEEE Transactions on Software Engineering*, 13(2):222–232, February 1987.

[7] Ann DiCaterino, Kai Larsen, Mei-Huei Tang, and Wen-Li Wang. An Introduction to Workflow Management Systems, November 1997.

[8] Mark Grand. *Patterns in Java*, volume 1. Wiley and Sons, Inc., New York, 1998.

[9] Apache Lenya project. http://lenya.apache.org, December 2004.

[10] Xiaogang Li and Gagan Agrawal. Supporting high-level abstractions through xml technology. In *LCPC*, pages 127–146, 2003.

[11] T. Lunt and R. Jagannathan. A prototype real- time intrusion detection expert system. In *Proceedings of the 1987 IEEE Symposium on Security and Privacy.*, Oakland CA, April 1988.

[12] J. Marin, D. Ragsdale, and J. Surdu. A hybrid approach to the profile creation and intrusion detection, 2001.

[13] Dan Cristian Marinescu. *Internet-Based Workflow Management.* Wiley, 2002.

[14] Sun Microsystems. Java technologies. http://java.sun.com/.

[15] N. Nguyen, P. Reiher, and G.H. Kuenning. Detecting insider threats by monitoring system call activity. Technical report, IEEE Information Assurance Workshop, United States Military Academy West Point, New York, June 2003.

[16] OASIS. Extensible Access Control Markup Language (XACML). http://www.oasis-open.org/committees/xacml, January 2005.

[17] OSWorkflow Group. OSWorkflow Project. www.opensymphony.com/osworkflow/.

[18] E. Sebring, E. Shellhouse, M. Hanna, and R. Whitehurst. Expert systems in intrusion detection: A case study. In *Proceedings of the 11th National Computer Security Conference*, Washington, DC, October 1988.

[19] Steven R. Snapp, James Brentano, Gihan V. Dias, Terrance L. Goan, L. Todd Heberlein, Che lin Ho, Karl N. Levitt, Biswanath Mukherjee, Stephen E. Smaha, Tim Grance, Daniel M. Teal, and Doug Mansur. DIDS (distributed intrusion detection system) - motivation, architecture, and an early prototype. In *Proceedings of the 14th National Computer Security Conference*, pages 167–176, Washington, DC, 1991.

[20] W3C. Extensible Markup Language (XML) 1.0 (Second Edition) . http://www.w3.org/TR/2000/REC-xml-20001006.

[21] Workflow Management Coalition. Workflow Management Coalition Audit Data Specification. Document Number WFMC-TC-1015, September 1998.