Role of an auditing and reporting service in compliance management

J. Ramanathan
R. J. Cohen
E. Plassmann
K. Ramamoorthy

Regulatory compliance has become a major focus in today's business environment as companies adapt to comply with regulations such as Sarbanes-Oxley, Basel II, and HIPAA (the Health Insurance Portability and Accountability Act). Runtime audit data that records information such as operational logs represents a key element needed for compliance management. An audit service that manages the life cycle of audit data is thus a critical component of any compliance management system. This service should support mechanisms to submit, centrally collect, persistently store, and report on audit data, as well as enable the archiving and restoration of audit data. This paper describes an audit service technology that is included in some IBM products to enhance their auditing capabilities and explains how this audit service can be used to support a company's compliance strategy. Using scenarios as examples, we show how reports provided by one of the products that uses this audit service can be instrumental in demonstrating compliance.

INTRODUCTION

Auditing is the process of maintaining detailed, secure records of critical activities in a business environment. Such records are referred to as *audit logs*. The critical activities recorded could be related to security, content management, business transactions, and so on. Security-related critical activities that could be audited include login failures, unauthorized access to protected resources, modification of security policy, noncompliance with a specified security policy, and the status of security servers. Business-related critical activities that could be audited include bank transactions, insurance claims processing, and order processing. Critical activities related to content management that could be audited include updates and deletions of critical documents.

Usage of audit logs

Customers must comply with many regulations, including the Sarbanes-Oxley (SOX) Act,[1] the Health Insurance Portability and Accountability Act (HIPAA),[2] the Basel II International Banking Accord (BASEL II),[3] and the Payment Card Industry Data

Security Standard (PCI-DSS).[4] To support their compliance initiatives from an information technology (IT) perspective, customers define and implement IT controls in their enterprise to enable them to test, measure, and understand how well they are complying with these regulations. Audit logs provide runtime data which demonstrates that such controls are being implemented. Thus, IT organizations can use information contained in audit logs to help demonstrate their compliance with certain aspects of these regulations.

In order to demonstrate a history of implementing IT controls, such audit logs must be archived (stored), sometimes for years, and procedures need to be put into place to allow the archived audit logs to be processed. As a result, the management of audit logs involves both technology and human resources. The National Institute of Standards and Technology's *Guide to Computer Security Log Management* outlines guidelines for managing audit logs.[5]

Audit logs are also useful in checking the enforcement and effectiveness of IT controls for accountability, vulnerability, and risk analysis. For example, an IT organization can use the audit logs of security-related critical activities to aid in the forensic investigation of security incidents. When a security incident occurs, the audit logs enable analysis of the history of activities that occurred before the incident so that appropriate corrective actions can be taken. This paper focuses exclusively on the use of audit logs for compliance management and does not discuss these other uses of audit logs.

Scenarios for use of audit logs in compliance

Best practices[6] recommend that companies set up IT governance structures to enforce compliance in their IT infrastructure. This governance structure involves setting up IT control structures to meet IT compliance objectives. Once controls are in place, procedures must be defined to test the controls and ensure that they are effective. Testing controls usually involves analyzing compliance data to determine that the controls are working properly. Compliance data can consist of configuration and policy data to show that the compliance policies have been implemented and history audit data to show that compliance policies are being enforced.

One key aspect of all compliance initiatives is gathering the compliance data needed to test IT controls. Gathering the required data has proven to be a problem area for customers because compliance data may be scattered throughout a company's IT infrastructure. Manual collection of the data can be a daunting task, in particular during an audit. IT employees frequently are required to discontinue all other work in order to satisfy an auditor's request for control test data.

Centralized collection of audit logs can help automate the collection of compliance history data, thus reducing the cost of collecting compliance data in support of control testing. In the following, we present several scenarios to illustrate how audit data can be used to support a company's compliance initiative.

The following compliance questions can be used as samples of the questions that need to be answered during an audit: (1) Are critical business applications being used by authorized users only? (2) Are user passwords being changed every 90 days? (3) Are user accounts removed in a timely manner after an employee leaves the company?

To answer the first of these questions, a control must be documented which defines the compliance objective that only authorized users should use critical business applications. In order to set up the tests for this control objective, critical applications must be identified, and a list of authorized users must then be created for each critical application. After the control test has been defined, the compliance tester can then look at the history audit data and ensure that all accesses to the critical applications were made by users on the specified list.

Many companies have policies regarding how frequently passwords must be changed. A corresponding control objective would state that procedures must exist to enforce frequent password changes. In order to set up the test for this control objective, the desired password-change frequency must be determined. This should be specified in the company security policies. To test for compliance with this objective, an audit report must be run and the resulting data analyzed to see if there are any passwords that are not being changed frequently enough.

Finally, to test the control objectives related to user management procedures, the control tester must

first obtain a list of all the employees that have left the company during the test period. A report can then be run against the audit data to show all userid deletion events. If a userid deletion event cannot be matched up with each employee that has left the company, the test results indicate a compliance violation.

There are various responsible parties in an organization who are involved in the preceding process. The chief compliance officer specifies an organization's high-level compliance policies. The security policy officer specifies security-related policies, including those that involve the generation of runtime audit data which implement the high-level compliance policies. The security administrator ensures that the systems are appropriately configured to collect the runtime audit data. The auditor uses the reports on runtime audit data to test whether the control objectives are met.

The preceding scenarios are related to IT controls that apply to the security aspect of the IT infrastructure. However, this is just one of the areas that need to be considered from a compliance perspective. Other areas that need to be considered are change management, business process management, document management, and so forth.

Requirements for managing audit logs

Because file-based audit logs cannot be easily processed for producing operational and trend reports, audit logs are typically made available in relational databases. The databases can be easily queried to generate operational and trend reports. Reporting facilities such as IBM DB2* Alphablox[7] or other popular business intelligence tools can then be used to analyze the data. Trend audit reports provide summarized audit data that enable a user to determine whether there are any long-term trends in critical activities. Operational audit reports enable detailed reviews of audit data to help analyze specific critical activities.

Based on the usage scenarios of audit data discussed previously, the management of audit logs has the following (not exhaustive) list of critical requirements:

1. Provide a process for managing audit logs that is tamper-resistant; that is, audit logs must be kept safe when generated, during movement, and when stored.
2. Collect and store large volumes of data for long periods of time.
3. Ensure that audit records are not lost or duplicated.
4. Stage the data periodically (daily or weekly) into report tables for data analysis.
5. Produce trend and operational reports on recent audit data.
6. Allow various reporting tools to be used to produce "out of the box" (ready to use without modification) and custom reports that show audit data.
7. Archive the audit data for long periods of time (months or years) with archiving scheduled on a regular basis.
8. Enable the review of audit logs for critical activities that occurred in the past (e.g., what a particular user did a month ago), which requires the ability to process archived audit logs.
9. Produce trend and operational reports on archived audit data.
10. Provide auditing functionality for changes to the configuration and policies for collecting audit data and for pruning audit logs.

An audit service technology

IBM products for distributed platforms currently generate audit logs that contain different formats for audit data of a specific type (e.g., authentication data, change management data, etc.). In addition, each product has its own way of managing the life cycle of audit data. An *audit service* provides mechanisms to manage the life cycle of audit logs. It provides mechanisms to submit, centrally collect, and persistently store, archive, restore, and report on current and restored audit data, and it satisfies the previously mentioned requirements for managing audit logs. A common audit service technology reduces the development effort of building an audit service for each specific domain and also enables customers to manage the life cycle of audit data consistently for all domains.

The audit service discussed in this paper supports auditing of data that is in a base format for which extensions can be defined for various domains. Currently, this audit service supports audit logs that are in the Common Base Event (CBE) Version 1.0.1[8] format. It also implements the extensions to the Common Base Event format for security events that

have been defined by IBM. It stores the audit logs in a relational database that is tailored for storing large volumes of data and also provides utilities that help with the life cycle (reporting, archiving and restoration, etc.) of audit logs.

The audit service discussed in this paper is a technology that can be used by a given IBM product to enhance its auditing capabilities. This audit service is currently used by the following products: IBM Tivoli* Access Manager for e-business Version 6.0,[9] the IBM Tivoli Access Manager for Operating Systems Version 6.0,[10] and the IBM Tivoli Federated Identity Manager Version 6.1.[11] This audit service is not available on its own as an IBM product offering. For multiproduct log aggregation capability for security events, IBM offers the Tivoli Security Operations Manager product.[12]

In the remainder of this paper, we first discuss the programming model, architecture, and various components of the audit service. We then discuss how this audit service is currently used by some IBM products to enhance their auditing capabilities. We conclude by providing sample scenarios of how a specific IBM product (Tivoli Access Manager for e-business Version 6.0) manages its audit data by using this audit service and reports on its audit data in support of compliance management.

This paper uses the terms audit data, audit log, and event to denote an audit record. Also, the term *exploiter* is used to refer to an IBM product that uses this audit service to enhance its auditing capabilities.

AUDIT SERVICE PROGRAMMING MODEL

The audit service provides interfaces for the creation and submission of audit records. These interfaces are available only for IBM products and are not available outside of IBM. The interfaces separate the creation of audit data and the submission of audit data into two separate steps.

Creation of audit data for C-based exploiters is done by using the C client. Creation of audit data for Java**-based exploiters is done using "event factories." An *event factory* is a set of interfaces in a particular language that allows creation of events in a given format. The audit service technology provides a security event factory that allows creation of security events which conform to the security event format that has been defined by IBM. In addition, exploiting applications can develop event factories for other domains.

Audit data can be submitted by using any of the following software. (In the text below, an *emitter* is a software entity that emits an object, in this case an event. A Java emitter is meant for use by Java-based exploiters. *A provider* implements the interface defined by the emitter.)

1. *Java-emitter lightweight provider*—The lightweight provider allows submission of audit records that are in the CBE format to a local CBE text-file log. This provider can be used by Java-based exploiters for their proof-of-concept solutions. No tools are provided to process audit data that is stored in this text-file log.
2. *Java-emitter enterprise provider*—The enterprise provider allows submission of audit records that are in the CBE format to a relational database by using a Common Audit Web Service. The Common Audit Web Service in turn submits these CBEs to the Common Event Infrastructure (CEI) server. Whether a specific event is audited is determined by the event group configuration, and "auditable" events are sent to the audit plug-in to which the audit database is registered as an audit provider for the CEI server. This enterprise provider can be used by Java-based exploiters to add enterprise audit capability for their product offerings.
3. *C client*—This client submits CBEs to a relational database by using the Common Audit Web Service. This interface is used by C-based exploiters.

Exploiters can embed the audit-service client files into their product so that these files are installed as part of their product's installation. The Common Audit Web Service is not externalized directly to IBM products or customers.

AUDIT SERVICE ARCHITECTURE

As shown in ***Figure 1***, the server side of the audit service consists of the CEI server, the Common Audit Web Service, the relational database to store audit logs (referred to as the audit database), the audit plug-in for the CEI server, the staging utility, and the XML store utility. The Common Audit Web Service receives CBEs from the C client and the Java-

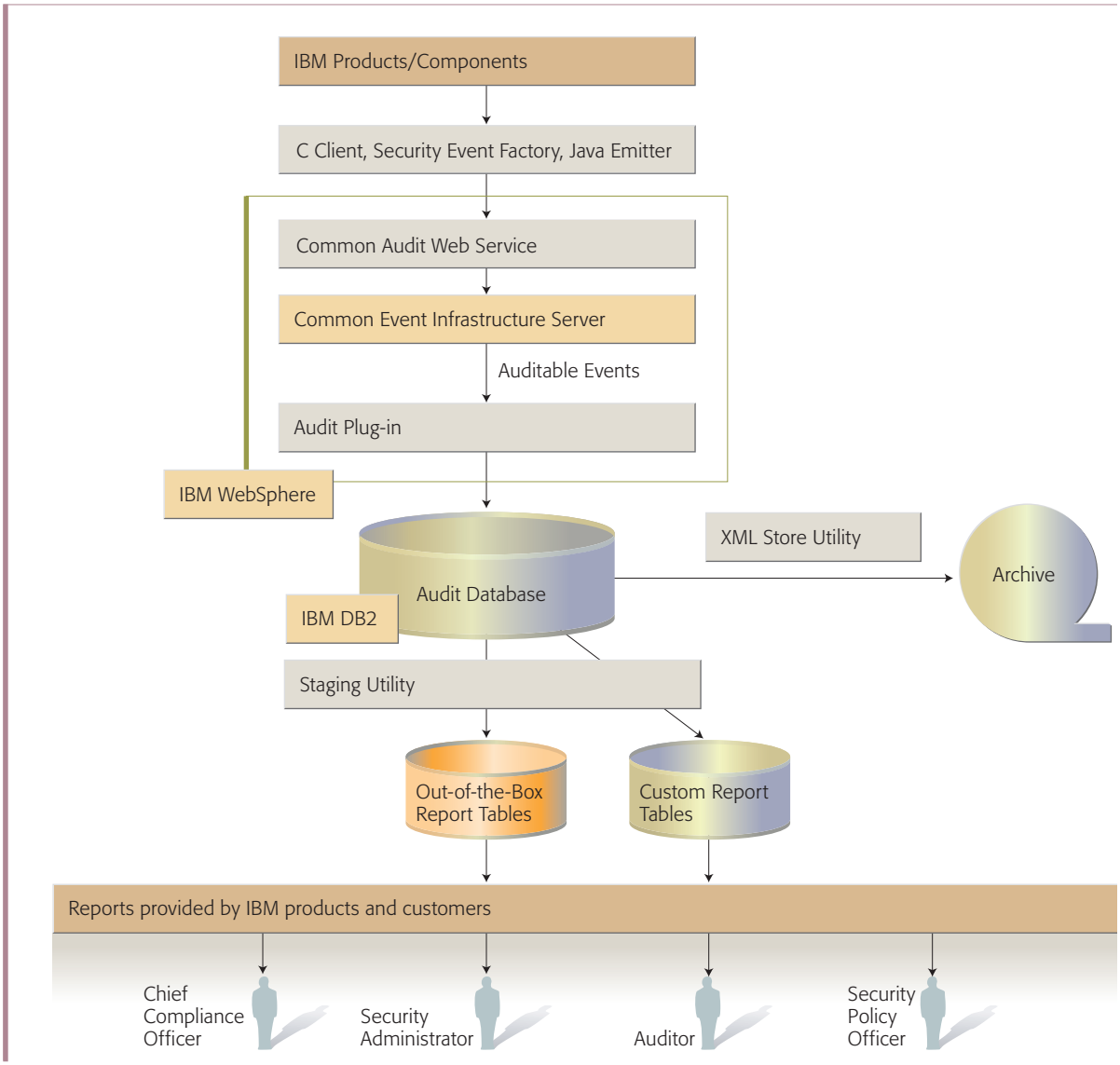

Figure 1
Audit service architecture

emitter enterprise provider and submits them to the CEI server. The server routes the CBEs that are configured to be "auditable" (using event groups) to the audit plug-in, which then stores them in the audit database. The staging utility stages the audit data from the main tables to report tables. These report tables could be the out-of-the-box report tables needed by the exploiting product's audit reports or custom report tables created by customers for their custom audit reports. The XML store utility helps with the archiving and restoration of the audit data from and to the audit database by using third-party archival tools.

Audit service components

The components of the audit service are the following: the C client, the security event factory, the Java emitter, the Common Audit Web Service, the audit plug-in, the audit database, the staging utility, and the XML store utility.

The C-client component provides the interfaces for use by C-based exploiting products for both creation and submission of security events. This component allows creation of security events that conform to the IBM-defined security extensions for CBE. It submits the CBEs to the Common Audit Web

Service. The C client and the Common Audit Web Service communicate by means of the SOAP (Simple Object Access Protocol) protocol. This component also provides a disk cache that caches CBEs locally when the Common Audit Web Service is unavailable or nonresponsive and submits them at a later time. In order to enable the audit service server to detect duplicate CBEs sent by this component (e.g., due to a timeout), this component generates a unique identifier for each CBE. The interfaces provided by this component are not made available outside of IBM.

The security-event-factory component enables the creation of security events in the IBM standard format for such events. This component allows Java exploiters to generate security events in the CBE format.

The Java-emitter component supports two providers that allow submission of CBEs. One of them is the lightweight provider that stores CBEs in a local text file. No form of processing of CBEs stored in this text file is provided. This implementation is meant to be used by exploiting products to meet their requirement for an out-of-the-box file-based audit log and for their proof-of-concept solutions. The other provider is the enterprise provider that submits CBEs to the Common Audit Web Service, which in turn stores them in the relational database to enable reporting and archiving. This provider includes a local disk cache for storage of CBEs when the Common Audit Web Service is nonresponsive or unavailable. In order to enable the audit service server to detect duplicate CBEs sent by this component, for instance, due to a timeout, this component generates a unique identifier for each CBE. These functions are supplied by the Java emitter for Java applications and by the C client for C-based applications.

The security event factory and the Java emitter are together referred to as the embeddable Java client. This client can be embedded by Java-based exploiting products as part of their product's installation files. Configuration for these components is provided through objects of the Java `Properties` class. Exploiters need to use product-specific configuration mechanisms to obtain configuration information and pass it to these components.

The Common Audit Web Service component services one or more CBE events from the C client and the Java-emitter enterprise provider and submits them to the CEI server. It is secured by role-based access control so that only authorized entities can submit audit events to it. In addition, the communication between the C and Java clients and this Web service can be secured with secure socket layer (SSL) mechanisms.

The audit plug-in component allows the audit database to be registered as an audit provider for the CEI server. By doing so, events that are denoted to be "auditable" by means of event group configuration are routed by the CEI server to this audit plug-in, which stores them in the audit database.

Audit database

The schema of the audit database must allow it to store a large number of events in a space-efficient format. It also needs to allow customers to generate operational reports. This schema uses several sets of tables to accomplish these objectives: active, inactive, restore, and report tables and metatables.

Active tables store events as the audit service server receives them. *Inactive tables* contain data that is to be archived and purged. Once the inactive tables are purged (emptied), they are eligible to become active. The customer decides, based on the volume of events and available disk space, the frequency with which tables can become active. *Restore tables* restore previously archived data. Typically, users would restore data from an earlier archive if they wanted to run a report against the archived data. *Report tables* store a subset of attributes from an event for operational reporting purposes. The customer decides the subset of attributes to store in the report tables. *Metatables* store meta-information, such as the set of XML event tables in the "active bucket" (i.e., the set in which incoming events are stored), the number of tables, the version of the CEI code that the schema supports, and so forth.

The schema for the audit database also uses a sequence object to ensure that an event is unique across active, inactive, and restore tables. When the schema was designed, the following functional and performance requirements were taken into consideration:

1. *Flexibility*—The schema allows for various versions of events to be stored in the same table. This is accomplished by storing each event completely and normalizing only a minimal

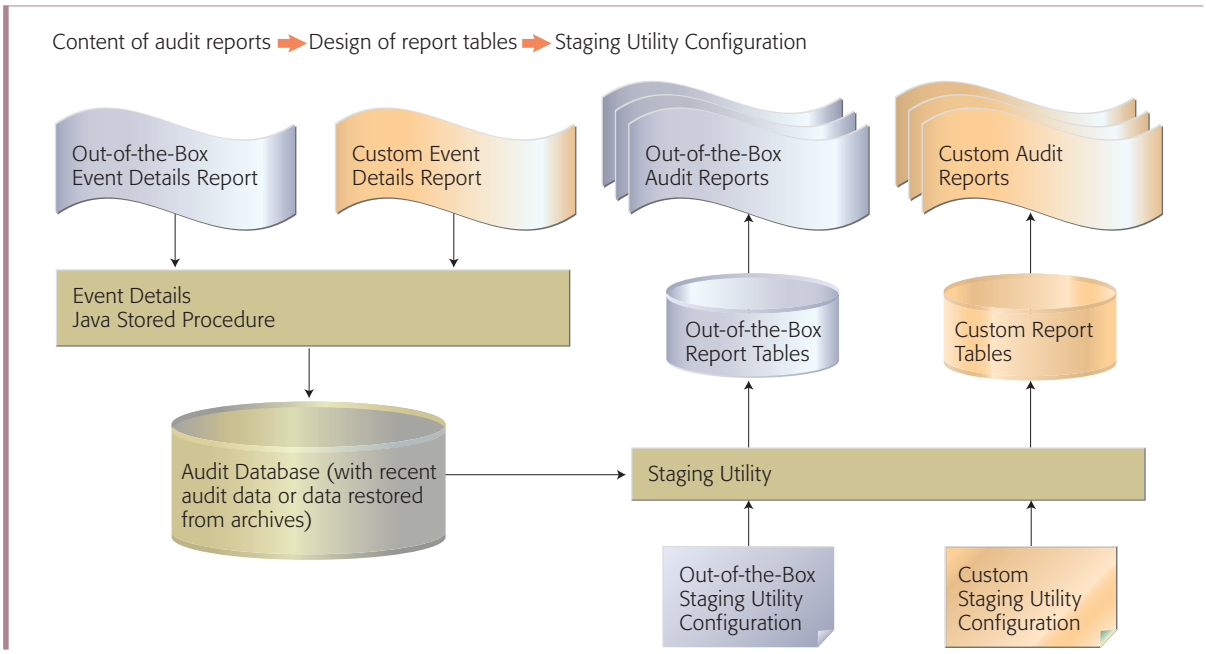

Figure 2
Staging utility and custom audit reports

number of attributes. The schema must also handle variable-sized events. Events (i.e., audit reports) that are less than 7793 characters are stored in a main table. Events that do not fit in the main table are stored in an overflow table. When compression for storage of audit events in the audit database is enabled, longer events will normally fit in the main table.

2. *Concise storage*—The large number of events necessitates that the schema provide a concise storage mechanism. The schema achieves this by allowing the events to be stored in a compressed format and allowing the users to select and store only a subset of attributes in the report tables for operational reporting purposes.
3. *Externalizing the schema and ease of migration*—The schema has to be made available to end users for archival purposes. One of the problems typically associated with externalizing a schema is the requirement to migrate data when the schema changes. Because all the attributes of an event are stored in a single database column, the data migration due to any change in the format of events is eliminated.
4. *High insertion performance*—Because the exploiting products' performance could be impacted, the schema allows for high performance during event insertion (for typical events) because of reduced disk I/O due to compressing the event.
5. *Fast purging of archived data*—The schema enables the purging of old audit data without causing the DB2 transactional log to become full. It accomplishes this by keeping active and inactive tables and by using the DB2 load utility to purge the data instead of using SQL (Structured Query Language) commands for this purpose.
6. *Incremental staging of data to report tables*—The volume of data requires that only new events, that is, those recorded since the last staging operation, are staged to the report tables. In order to support this requirement, the events need to be identified by a key that is unique and monotonically increasing in nature. The schema accomplishes this by using a record identifier generated by a sequence object.

Staging utility

As shown in ***Figure 2***, the staging utility supports a procedure for defining the subset of data for each audit event that is recorded in the report tables. Custom reports can then be created to analyze the custom subset of data for each event. These custom definitions need to be based on the compliance needs of the organization and are designed to demonstrate compliance with the IT controls that

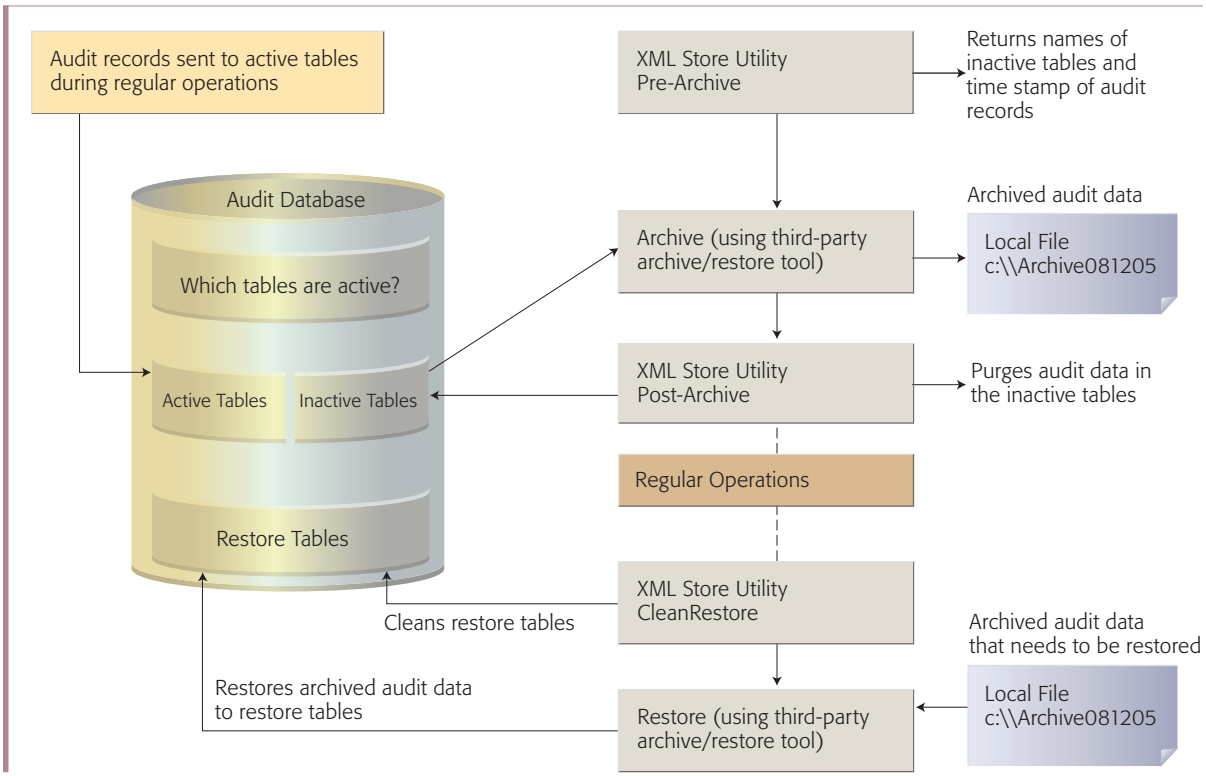

Figure 3
Audit-database archive and restore example

are being tracked in the organization and require the use of audit data. Any reporting tool that queries data from a relational database can be used to build the custom reports that process this data. As custom reports are developed, the event types and specific elements of each event type that are of interest need to be identified and specified for configuring the staging utility. The customer needs to create and manage the additional report tables to hold data for custom reports. The staging utility can then be executed to stage custom data into these newly defined report tables.

Each event stored in the audit database has a unique identifier associated with it. All details of a particular event can be obtained by providing this unique identifier to the Java stored procedure which handles event details.

To support custom reports, a custom Data Definition Language (DDL) file needs to be written and run to create custom report tables in the audit database. Specific guidelines need to be followed in creating the custom report tables to allow joining of data from these tables with data from the default report tables and to allow the staging utility to prune data in the custom report tables. The configuration of the staging utility must be updated in order to stage the additional attributes needed for the custom report when the staging utility is run.

XML store utility

The XML store utility enables the archiving and restoration of audit data contained in the main tables of the audit database by using third-party archival tools. ***Figure 3*** shows an example scenario of archiving and restoration.

The archive operation includes the following steps: (1) execute the XML store utility with the "pre-archive" option to get information about the names of the inactive tables and the time stamps of the events being archived; (2) archive the inactive tables by using the third-party tool; and (3) execute the XML store utility with the "post-archive" option to purge events from the inactive tables that were just archived.

The restore operation includes the following steps: (1) optionally, execute the XML store utility with the "clean-restore" option to purge existing events in the restore tables, and (2) restore events to the restore tables by using a third-party tool.

The period of time for which archives are maintained depends on the business policy of the organization. The life cycle of audit data needs to include proper disposal of the archives as specified by this business policy.

Functionality

In this section, we discuss how the functionality supplied by the audit service addresses most of the requirements for managing audit logs that were listed in the section "Requirements for managing audit logs."

The requirement for a tamper-resistant process for managing audit logs is met by the audit service in the following manner. The audit service uses authentication, access control, and secure communication to ensure that only authorized clients can submit audit records to the audit-service server, which stores such records in the audit database. Access to the audit database must be restricted by using database access control mechanisms. Access control must be enforced also for the original sources of audit data (e.g., Tivoli Access Manager for Operating System binary audit log files).

The requirement for collecting and storing large volumes of data is met by the design of the schema of the audit database as discussed earlier in this paper. The requirement for preventing lost or duplicate audit records is met by the disk cache mechanism and by generation of a unique ID per audit event, implemented in both the C client and the Java emitter enterprise provider, and is also met by using DB2 mechanisms to detect duplicate records in the audit database and discarding them without storing them again.

The requirements for periodic staging of data into report tables, producing trend and operational reports and enabling various reporting tools to produce out-of-the-box and custom reports are met by the ability of the staging utility to stage recent audit data for custom operational reports.

The requirements for archiving and processing of archival logs are met by the schema design of the audit database and also by providing the XML store utility that assists in the archival and restoration process. Producing trend and operational reports on archived data is enabled for operational reports by the staging utility's ability to stage archived audit data into custom report tables. Note that the archived audit data is restored from archives to the restore tables of the audit database, and the staging utility then stages the data from those tables to the report tables.

Staging for trend reports is not supported for either current or archived audit data.

Guidelines for using the audit service

To use the audit service, a product must define the out-of-the-box reports for audit data to be generated and the report tables needed to support the out-of-the-box reports. The audit data needed for the defined reports must be created in the CBE format. If security events are being generated, the security event factory must be used.

The audit data is then submitted by using the C or Java interfaces. It is staged in the report tables based on a defined configuration by the staging utility. Audit reports are then developed by using a reporting tool. The files for the clients of the audit service are embedded in the product's installation files, and the end-user installation package is shipped to the server side of the audit service.

USES OF THE AUDIT SERVICE

IBM Tivoli Access Manager for e-business Version 6.0 uses the C and Java interfaces to audit the following security events using the audit service described in this paper: authentication, configuration, authorization, runtime operations for security servers, resource access events, user self-care password-change operations, and management operations for resources, security policies, users, and groups.

IBM Tivoli Access Manager for Operating Systems Version 6.0 uses the C interfaces to audit the same set of security events with the addition of credential modification operations, and is limited to management operations for resources, security policies, and users.

IBM Tivoli Federated Identity Manager Version 6.1 users the Embeddable Java Client to audit security

events for authentication, federation, trust, signing, encryption and management operations for security policies using this audit service.

In the following section, we provide examples to illustrate how reports generated for the security events of one of these products can be used for compliance management.

Compliance management

IBM Tivoli Access Manager for e-business Version 6.0 provides the following reports for audit data logged by its components using the audit service:

- *General Audit Event Details Report*—Displays all information about a single auditable event denoted by the event reference ID parameter. Typically, a user runs this report after running other reports and deciding that an event "drill down" (i.e., detailed analysis) is desired.
- *General Audit Event History*—Displays the total number of auditable events for each event type during a specified time period. It also shows all events of a specified event type and product name, sorted by a selectable sort criterion and time stamp. This report can be used for incident investigation and assuring compliance.
- *Audit Event History by User*—Displays the total number of events for a specified user during a specified time period. It also presents a list of all events of the specified event type and product name, sorted by time stamp and grouped by session ID during the time period. The purpose of this report is to investigate activity of a particular user during a specified time period.
- *Failed Authentication History*—Presents a list of all failed authentication events over a time period, sorted by selectable sort criteria such as time stamps. This report can be used by an administrator to investigate security incidents.
- *Failed Authorization History*—Lists all of the failed authorization events during a specified time period.
- *Locked Account History*—Displays all of the accounts that have been locked during a specified time period.
- *User Password Change History*—Displays events related to password changes performed by the users themselves during a specified time period.
- *Administrator and Self-Care Password-Change History*—Displays events related to password changes performed by the user and the administrator during a specified time period.
- *Server Availability Report*—Shows the availability status of security servers on a specific machine. The user can display data for all protected machines in the report or limit the report by entering a single host name as the subject of the report.
- *Certificate Expiration Report*—Allows detection of soon-to-expire certificates and highlights the need to replace the certificate to ensure 24/7 operability. It shows the number of clients that have server/SSL certificates which expire in a certain number of days. It also shows a table of client host names, the days until their certificates expire, and the server they are associated with.
- *Most-Active-Accessors Report*—Shows a list of users who are the most active in the system. This can lead administrators to investigate improper use of resources.
- *General Authorization Event History*—Displays the total number of authorization events, failed authorization events, successful authorization events, and unauthenticated events during the specified time period. Additionally, it shows a list of all authorization events, sorted by a selectable sort criterion (time stamp, resource, or user name) during the time period. The purpose of this report is to analyze the authorization event history for incident investigation and assuring compliance.
- *Authorization Event History by Action*—Displays a list of all authorization events that contain the specified action, sorted by resource and then time stamp during the time period specified.
- *General Administration Event History*—Shows the history of general management actions done over a specified time interval. The administrator can use the report to track the actions of a user for administrative events.
- *User Administration Event History*—Can be used to investigate security incidents and to track changes made by administrators which affect users.
- *Group Administration Event History*—Can be used to investigate security incidents and to track changes to groups by administrators.
- *Security Server Audit Event History*—Presents a list of auditable events related to security servers that occurred during the specified time period.
- *Resource-Access-by-Accessor Report*—Shows the top resources in terms of access or authorization events during a time period for each machine

Table 1 Use of reports in compliance testing

Compliance Category	IBM Tivoli Access Manager for e-business Report
Authenticating all users	General Audit Event History Failed Authentication History
Maintaining effective access and authentication	User Administration Event History Group Administration Event History Server Availability Report General Authorization Event History Locked Account History User Password Change History Administrator and Self-Care Password-Change History
Defining user account management procedures	User Password Change History Administrator and Self-Care Password-Change History User Administration Event History Group Administration Event History Security Server Audit Event History
Following appropriate segregation of duties	General Administration Event History
Monitoring and logging security activities	General Audit Event Details Report General Audit Event History Audit Event History by User Failed Authentication History Failed Authorization History Locked Account History User Password Change History Administrator and Self-Care Password-Change History Certificate Expiration Report Most Active Accessors Report Authorization Event History by Action General Administration Event History User Administration Event History Group Administration Event History Security Server Audit Event History Resource-Access-by-Accessor Report Resource-Access-by-Resource Report

name identified. The report identifies who is repeatedly accessing resources and what resources are being accessed.

- *Resource-Access-by-Resource Report*—Shows the top accessors in terms of access authorization events during a time period for each machine name identified. The report identifies which resources are most heavily accessed and which users are accessing those resources.

Table 1 shows how some of the reports described in the preceding subsections can be used to support control objective testing in the specified compliance categories.

Sample scenarios

In this subsection, we present three sample scenarios which illustrate the use of reports to support various compliance categories.

Figure 4 shows how the Audit-Event-History-by-User report can be used to look into violations of policies related to segregation of duties. "Segregation of duties" refers to policies wherein a user assigned to perform a particular task is not allowed to perform another task. For example, a user who accesses a Web application cannot be the one deciding who can access a given Web application. In *Figure 5*, the Resource-Access-by-Resource report monitors accesses to a critical resource, such as a critical application. In *Figure 6*, the Resource-Access-by-Resource report is used to review whether only authorized entities have access to a critical file.

CONCLUSION

In this paper, we have discussed the key role an audit service plays in compliance management. We illustrated this role by showing how an audit service

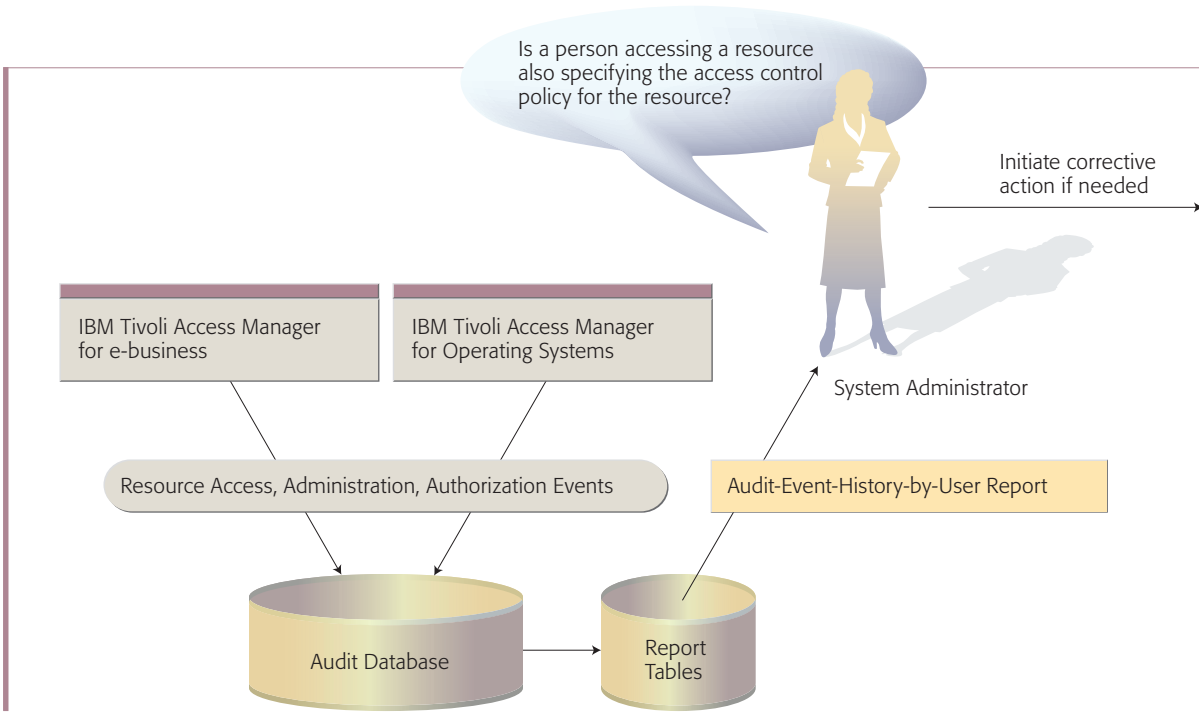

Figure 4
Segregation of duties

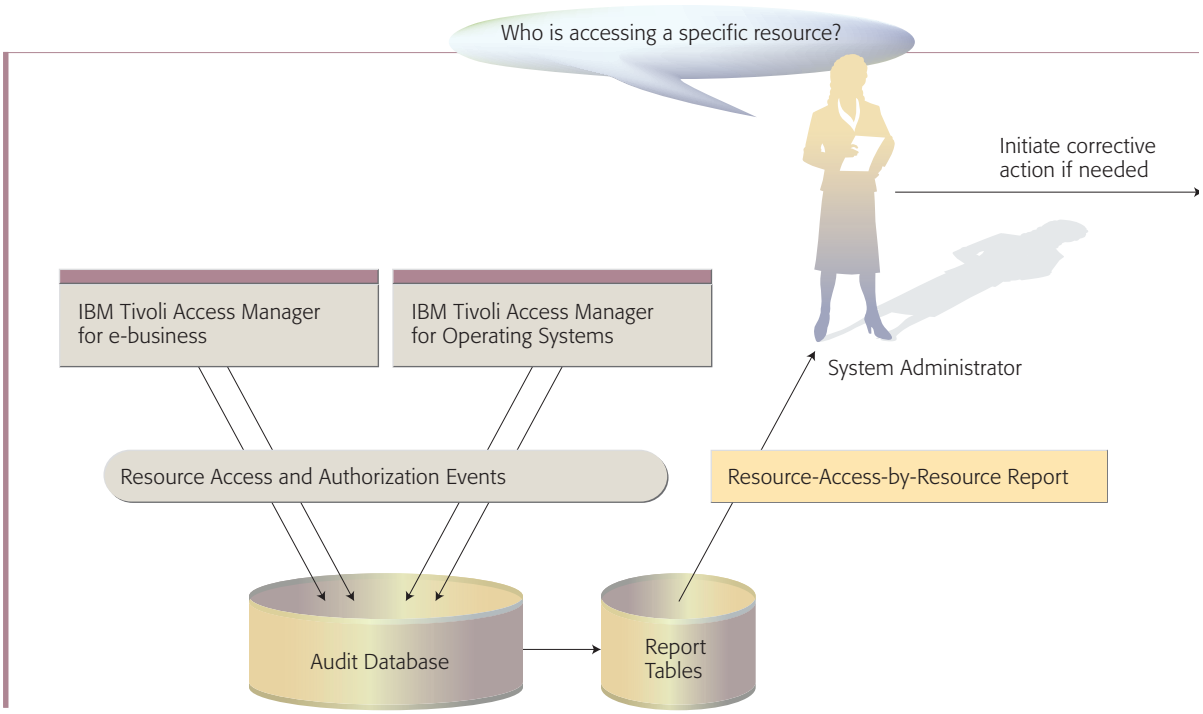

Figure 5
Monitoring critical resources

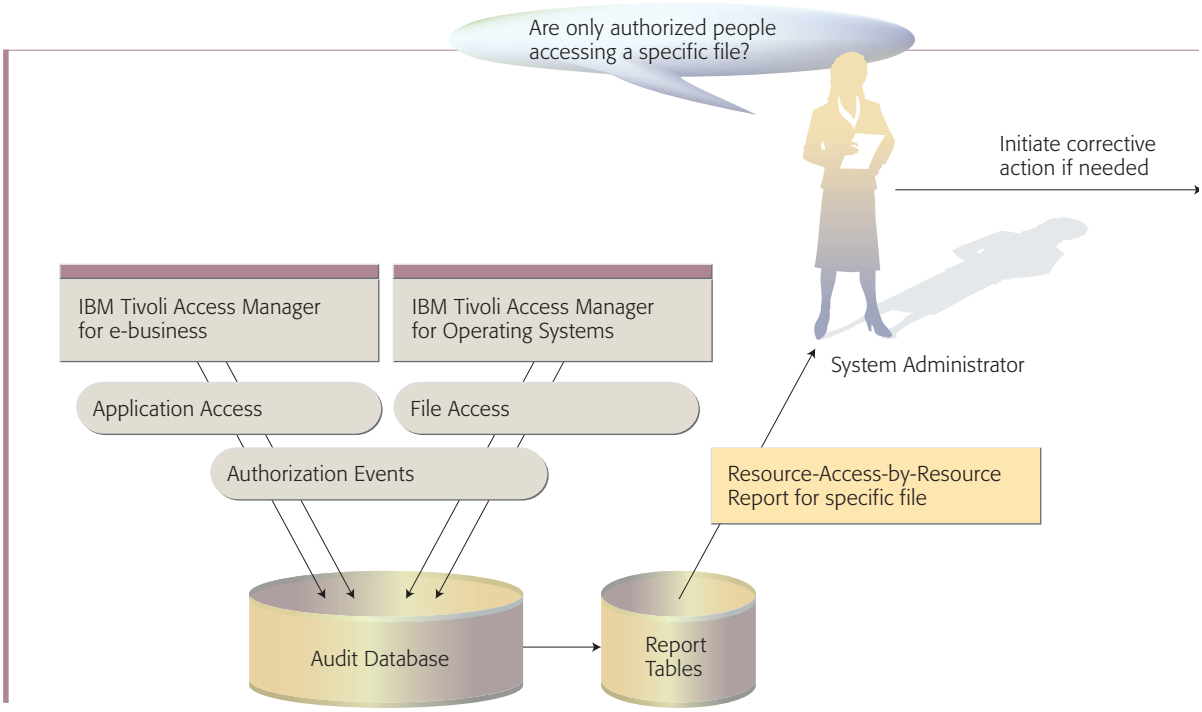

Figure 6
Monitoring specific file access

can be used for compliance management by discussing its architecture, how products can use it, and how a particular product that uses this audit service can generate reports from its audit logs that can be used in support of compliance management.

This paper focused on only the compliance-management-related scenarios for usage of audit events. Other solutions using audit events include those related to risk analysis (e.g., IBM Identity and Risk and Investigation Solution[13]), security information and event management (e.g., IBM Tivoli Security Operations Manager), and incident investigation. These solutions represent potential areas for further research.

ACKNOWLEDGMENTS

The authors would like to thank Anthony Nadalin, Nataraj Nagaratnam, Robert High, Randy Forlenza, Timothy Hahn, Jim Fletcher, and John Dinger for providing architecture direction for the work discussed in this paper. The authors would also like to thank Arvind Krishna, Steve Wojtowecz, Brian Turner, Jody Hasten, Ann Graham, Lisa Zinna, Lee Hagy, Vincent Abbosh, Winton Campbell, Patricia Griffin, Chris Lita, Phil Klickman, and Kellie Lecompte for their contributions to the development of the audit service discussed in this paper.

CITED REFERENCES

1. *Public Law 107-204, Sarbanes Oxley Act of 2002*, 107th Congress of the United States of America (2002), http://www.sec.gov/about/laws/soa2002.pdf.
2. *H. R. 3103, Health Insurance Portability and Accountability Act of 1996*, 104th Congress of the United States of America (1996), http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?dbname=104_cong_bills&docid=f:h3103enr.txt.pdf.
3. *International Convergence of Capital Measurement and Capital Standards—A Revised Framework*, Basel Committee on Banking Supervision (2004), http://www.federalreserve.gov/boarddocs/press/bcreg/2004/20040626/attachment.pdf.
4. *Payment Card Industry (PCI) Data Security Standard, Version 1.1*, PCI Security Standards Council (2006), https://www.pcisecuritystandards.org/pdfs/pci_dss_v1-1.pdf.

5. K. Kent and M. Souppaya, *Guide to Computer Security Log Management*, National Institute of Standards and Technology, Special Publication 800-92 (2006), http://csrc.nist.gov/publications/nistpubs/800-92/SP800-92.pdf.
6. CobIT—Control Objectives for Information and Related Technology, IT Governance Institute http://www.itgovernance.co.uk/page.cobit.
7. DB2 Alphablox, IBM Corporation, https://www-306.ibm.com/software/data/db2/alphablox/.
8. D. Ogle, et al., *Canonical Situation Data Format: The Common Base Event V1.0.1*, http://www.eclipse.org/tptp/platform/documents/resources/cbe101spec/CommonBaseEvent_SituationData_V1.0.1.pdf.
9. Tivoli Access Manager for e-business, Tivoli Information Center, IBM Corporation, http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.itame.doc/welcome.htm.
10. Tivoli Access Manager for Operating Systems, Tivoli Information Center, IBM Corporation, http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.itamos.doc/welcome.htm.
11. Tivoli Federated Identity Manager, Tivoli Information Center, IBM Corporation, http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.tivoli.fim.doc/welcome.htm.
12. Tivoli Netcool Security Operations Manager Version 3.1, Tivoli Information Center, IBM Corporation, http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/index.jsp?toc=/com.ibm.netcool_som.doc/toc.xml.
13. *IBM Identity Risk and Investigation Solution* (2006), http://www-935.ibm.com/services/us/bcs/pdf/g510-6527-ibm-identity-risk.pdf.

Accepted for publication November 21, 2006.
Published online April 11, 2007.

Jayashree Ramanathan
IBM Software Division, Tivoli, 11501 Burnet Road, Austin, Texas 78758-3400 (jramanat@us.ibm.com). Dr. Ramanathan is a security architect in the IBM Austin Development Laboratory. She received an M. Tech. degree in computer science from the Indian Institute of Technology in Mumbai, India in 1985 and a Ph.D. degree in computer science from Michigan State University in 1992. She joined IBM in 1992, and has worked in the areas of clustering, access control, auditing, security events, and compliance. She is the lead architect for the audit service discussed in this paper.

Richard J. Cohen
IBM Software Division, Tivoli, 11501 Burnet Road, Austin, Texas 78758-3400 (rcohen@us.ibm.com). Mr. Cohen is a compliance and security architect for Tivoli Security products at the IBM Austin Development Laboratory. He received a B.A. degree in computer science and a B.S. degree in computer science from the University of Texas at Austin in 1980 and 1987, respectively. Mr. Cohen joined IBM at the Austin Development Laboratory and started his career working in various areas related to compilers and operating systems. From 1991 to 1992, he was on assignment at the Information Technology Center at Carnegie Mellon University. For the last 14 years, Mr. Cohen has worked on distributed systems and security products, focusing in the last three years on the compliance area. As part of his distributed systems work, he was the primary author of the DCE Event Management System (EMS) specification.

Ernst Plassmann
IBM Software Division, Tivoli, 11501 Burnet Road, Austin, Texas 78758-3400 (eplassma@us.ibm.com). Mr. Plassmann is a software engineer at the IBM Austin Development Laboratory. He is the development technical lead for the audit service discussed in this paper. Prior to this, Mr. Plassmann worked as a developer on the IBM Tivoli Access Manager for e-Business product. He received a B.S. degree in computer science from the University of Houston in 1987.

Karthikeyan Ramamoorthy
IBM Software Division, Tivoli, 11501 Burnet Road, Austin, Texas 78758-3400 (kramamoo@us.ibm.com). Mr. Ramamoorthy has been a software engineer with IBM since 1998 and a member of the audit-service development team since 2004. He received an M.S degree in computer science from the University of Michigan in 1993. Prior to joining IBM, Mr. Ramamoorthy worked as a software engineer for the University of Michigan Digital Library Project (JSTOR). He also worked at FAME Information Services on a proprietary database product to store time-series data and delivery mechanisms for daily stock market data. ■