Events Classification in Log Audit

Sabah Al-Fedaghi[1] and Fahad Mahdi [2]

Computer Engineering Department, Kuwait University, Kuwait
[1]sabah@alfedaghi.com, [2]fahad.a.mahdi@gmail.com

ABSTRACT

Information security audit is a monitoring/logging mechanism to ensure compliance with regulations and to detect abnormalities, security breaches, and privacy violations; however, auditing too many events causes overwhelming use of system resources and impacts performance. Consequently, a classification of events is used to prioritize events and configure the log system. Rules can be applied according to this classification to make decisions about events to be archived and types of actions invoked by events. Current classification methodologies are fixed to specific types of incident occurrences and applied in terms of system-dependent description. In this paper, we propose a conceptual model that produces an implementation-independent logging scheme to monitor events.

KEYWORDS

Information security, event classification, audit system, log analysis.

1. INTRODUCTION

An event log or audit trail is an ordered sequence of occurrences containing evidence of the execution of a "process" by users, systems, or other entities. Various sources and entities in the system send messages regarding their processes (e.g., who, what operations, time, etc.) that are kept in several logs, including logs about the following:

Application events: Events describe operations of various application programs.

Security-related events: Events report on successful or failed operations, e.g., attempts to access critical servers.

System events: Events that include setup events and system warnings, e.g., bad disk, attempt to tamper with a system file.

Event logs are becoming increasingly valuable tools for monitoring the security and performance of computer systems and networks. Their rationalization is to be alert since prevention is better than cure. Their benefits and usages include
- complying with security policies and regulation
- developing systems for troubleshooting and problem diagnostics
- providing evidence of accidental or deliberate security breaches for forensic investigations
- detecting a problem in the system, hence providing an opportunity to respond quickly to security incidents (e.g., a firewall failure)
- providing information to help in monitoring, retention, and reporting of operations
- use in software development to support debugging of the application.

Logs of events by servers, network devices, diagnostic tools, and security-specific devices have increased tremendously in size. Increasing the number of audited events consumes more system resources, impacting system performance. Accordingly, the production of auditing reports involves *selectively* collecting logged data and aggregating events. Most recorded events are not valuable and create difficulties for allocating important messages. Analyzing log data involves sorting events according to priorities and categorizing audited events to cover various systems' functions.

Log analysis may involve a human expert who defines message patterns and responses to occurrences [11, 22]. Tools can be used to graphically visualize log data to help in the analysis [21]. Detecting message patterns may focus on frequency of occurrences [21]. Another method is to analyze time correlation and utilize text analysis algorithms [15, 20].

According to recent reports, logging and event monitoring solutions such as analysis of audit trail have not measurably improved the overall state of information security [19]. "There are several reasons. First, logging solutions are commonly underutilized. . . . Similarly, event monitoring and management tools . . . are rarely used to their full potential" [19]. In addition, "70% of information in logs is irrelevant," and "Logs are written by developers, not managers or system administrators" [14]. The problem is that logging results in vast quantities of collected data and hence does not lead to effective analysis of auditing. A basic concern is that of controlling the quantity of data collected. Additionally, according to Anderson [7], for audit logging "it is necessary to provide a data reduction program to summarize significant security events." Additionally, a partial solution has been proposed to index and classify log event data and to look "for a log management tool that classifies and trends log data it hasn't handled previously" [14].

In terms of theoretical aspects, a computer system-dependent theory to categorize events based on its source (location) is lacking. For example, an event in "Event Viewer - Windows XP" log is classified as Information, Warning, Error, Success Audit (Security log), and Failure Audit (Security log) [16]. Of course, these classes are attached to their "place of occurrence," such as resources and function, e.g., an Information event is logged when a network driver loads successfully. In this case the "place of occurrence (source of event)" (e.g., a network driver) is taken as a black box that omits messages related to its interior and/or its input and output.

We claim that this black-box approach blocks valuable information that could be used in analyzing event logs. This paper is concerned with finding a "generic" classification of log events according to stages of information flow (e.g., creation stage, receiving stage). "Generic" here means conceptually independent categories. Computers, sub-systems, operating systems, programs, servers, etc. can be viewed as "information machines" with five main states (stages) of flow of information, supplemented by many substages. Even in a network-oriented environment where resources are distributed, connection channels can be viewed as an information system that receives, processes, creates (e.g., noise), releases, and transfers data [5]. Thus, network traffic logging complements individual system logs by recognizing chains of events across systems. The five stages are "functional" in the sense that they are system-impendent and describe tasks such as receiving, releasing, processing, creating, and transferring. Thus, the source of an event can be qualified with its functional place ("upon receiving," "upon transferring"), which creates a meaningful interpretation of the event (e.g., system *received* information from an outsider, and such an event, somehow, penetrated [flowed to] the processing stage, resulting in retrieval, release, and transfer of a critical file, e.g., SQL injection attack). In this case, the pattern of the sequence of interior events identifies the type of attack.

This model with five stages of information flow will be introduced in section 5. In preparation for utilizing such a model in events classification, section 2 reviews the general notion of logging. Since we concentrate on security-based events, section 3 discusses classifying of threats as an illustration of our approach to be introduced later. Section 4 discusses the general issue of event classification. Following the background introduced in the first five sections, sections 6, 7, and 8 introduce our methodology in terms of a flow-based model. Section 9 describes experimental results demonstrating application of the model to classifying of events.

2. ABOUT LOGGING

Logging has been the traditional means for recording activities in computer systems. Log entries are produced by the activities in network devices, operating systems, applications, hardware- and software-based firewalls, Web servers, authentication servers, management information system servers, Web portals, database, query and index servers, mail servers, and Web mail servers.

Most computer systems perform the logging of important events during running of the system for different purposes, e.g., accounting and monitoring. We concentrate on a security audit trail concerned with "who was doing what in a . . . system . . . [to] detect patterns of abnormal activities . . . [and] provide a sufficient record to determine penetration that may be discovered by external means" [7]. In general, the concern in security-related events can be generalized as who did what to which object, when, and on which system. This process aims at finding defects and taking corrective action.

From the security point of view, monitoring for security breaches can be accomplished by means of network level TCP/IP, server, and application, and through process-specific monitoring [8]. A record of monitoring typically shows the identity of any entity that has accessed a system and types of operations executed. It may include attempted accesses and services, a sequence of events that have resulted in modifying data. The purpose is mostly to provide evidence for reconstruction of the sequence of events that led to a certain effect or change.

To accomplish its function, an auditing system runs in a privileged mode to oversee and monitor all operations. Key information in such a system includes information format, type of activity, identity, storage, location, time, cause, tools and mechanisms used, and so forth [18].

Logs are typically created for such activities as
- Creating, reading, updating, or deleting confidential information
- Initiating or accepting a network connection
- User authentication and authorization such as user log-in and log-out
- Granting, modifying, or revoking access rights, including adding a new user or group, changing user privilege levels, changing file permissions, changing database object permissions, changing firewall rules, and changing user password
- System, network, or services configuration changes, including the auditing system itself
- Application process startup, shutdown, or restart
- Application process abort, failure, or abnormal end
- Detection of suspicious/malicious activity [18]

At the implementation stage, the "First step to implement an audit logging infrastructure is listing critical systems . . . and determining what logging is turned on" [8]. At the analysis stage, raw log data are analyzed and interpreted to consolidate logs. Sophisticated tools are needed to spot security problems. A number of software products are available to help collect and analyze audit and log data.

3. THREATS CLASSIFICATION

Monitoring and logging events are designed according to potential threats. Consider the threat called "content spoofing," which is used to trap a user into thinking that content appearing on a Web site is legitimate. This can be accomplished through *gaining access* to a server and altering its content. For example, in a case where serving Web pages requires dynamically created HTML frames, the source location (e.g., "frame_src") of a frame is replaced with another value controlled by the attacker. Consequently, a user may be lured to a Web page designated by the attacker and believe it to be the authentic *content* of a trusted Web site. Spoofed sites may include fictitious design forms and *log-in* applications that require sensitive data such as passwords and credit card numbers.

This problem can be analyzed in terms of failure to adequately protect the file against modification, and an implementation-specific solution in terms of building dynamically created HTML frames is discussed. Assume a "content spoofing" event where the attacker modifies information and links in an established Web site by accessing and altering content on the server. Typically, a solution to the problem is developed for the server side to fix the frame-spoof vulnerability. No systematic framework is introduced to identify "weak points" in the system that should be monitored and protected.

In our approach, we view this class of attack as an event that embeds considerable vulnerability into different subsystems. Consider attacker requests for systems services illustrated in the following:
Attacker: transmits to the system (illegitimate) log-in information.
System: (1) *receives* and (2) *processes* such information and accepts the attacker as a user.
Attacker: requests some content.
System: (3) *retrieves*, (4) *releases*, and (5) *transmits* the content.
Attacker: alters the content.
Attacker: sends new content.
System: *receives* new content.
Attacker: requests replacing old content with new content.
System: (6) *replaces* content with new version.

In such a scenario, the vulnerable points in the system that require close monitoring are illustrated in Figure 1.

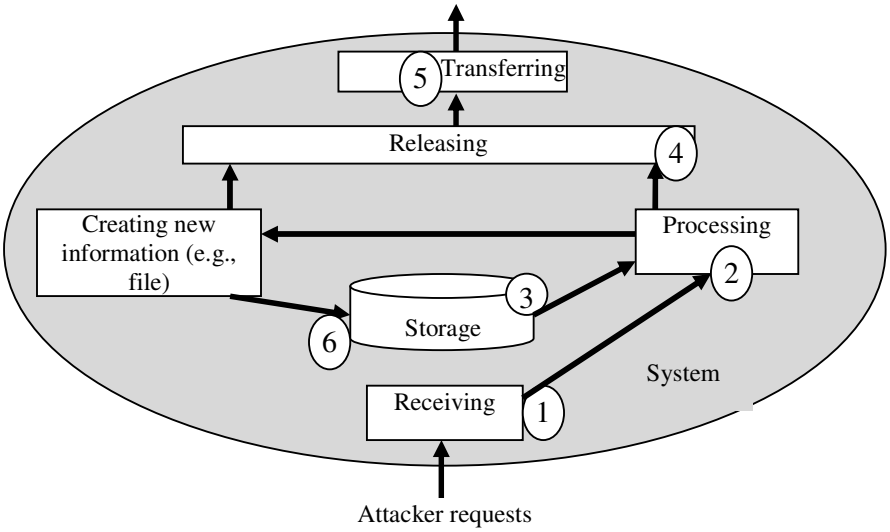


Figure 1. Illustration of our generic approach showing locations in the system critical to "spoofing content."

The circled numbers refer to points that ought to be monitored and actions that ought to be performed immediately to abort content spoofing. System operations are described in terms of generic stages such as receiving, releasing, transmitting, processing, and creation of information, as described later. In such an approach, the classification of threats and resultant logging and monitoring scheme are not fixed to specific types of incident occurrences or applied in terms of system-specific description (e.g., dynamically created HTML frames).

4. EVENTS CLASSIFICATION

A basic question we raise in regard to logging is the nature of "events" recorded. The notion of "event" is a central concept in logging. It might indicate a state change, failure (e.g., log-in), or occurrence of an activity (e.g., availability of a service). In the operation system, it is an entry in the audit trail such as "request for file access." According to Becta [8],

> All of this information [collected by logging system] does not necessarily indicate the specific events to log, or *how to classify* the events. For example, is a user log-on and log-off *a single event with two states, or two separate events*? The audit events recognized by a system depend entirely on the capabilities and sophistication of the system's components and security mechanisms.

Event classification refers to defining the characteristics of a system's various sets of events, grouping them into logical categories to facilitate security objectives. In addition, auditable events are selected to *cover* (a) different types of events at (b) different security-relevant points in the system. Accordingly, the level of detail of auditable event definition is next decided. It is possible that a given event may be recorded at various points.

Many security-oriented event classifications have been given. We review a few methods as examples.

AIX [6] uses the following classifications (some subcategories are deleted for the sake of brevity):

Security Policy Events

- Subject Events: process creation, process deletion, setting subject security attributes
- Object Events: object creation, object deletion, object open, object close
- Import/Export Events: importing or exporting an object
- Accountability Events: adding a user, changing user attributes in the password *database, user log-in,* and *user log-off*
- General System Administration Events: use of privilege, file system configuration, device definition and configuration, normal system shutdown
- Security Violations (potential): access permission refusals, privilege failures, diagnostically detected faults and system errors

In such a long list of events, grouping is performed according to subject/object, then import/export, then violations. In this case, a list seems to be built by examining various rules (subject/object), functions (e.g., accountability), organizational units (system administration), and status (violations).

In the area of network intrusion events, Kazienko and Dorosz [13] list monitoring of the following events groups:

- Network traffic (packets) attempting to access the host
- Log-in activity on the networking layer
- Actions of a super-user (root)
- File system integrity, e.g., any changes to the files
- State of system registers
- State of key operating system files and streams

We can observe no systematic grouping of event types—activities, actions, or changes to files, states, or streams—extracted from the network operating environment.

Windows event logs contain the following types of events [9, 10]: error events, warning events, information events (successful operation of an application, driver, or service), success audit events, failure audit events. Again, these classes of events seem to lack systematization and are specified according to the hosting system.

HP recommends basic monitored events: admin event, log-in event, moddac self-auditing event, execv, execve, and pset event [9, 10]. "Admin," "log-in," "exec" . . . seem to be targeted events according to the specifics of the system.

Web Services Architecture [23] contains an "audit guard" that monitors agents, resources and services, and actions relative to one or more services. We again note the heterogeneous categorization of monitoring types.

It can be concluded that a general theory of event classification for monitoring purposes is lacking. The required classification must be generic in the sense that it is not tied to any specific system activity. In the next section, we introduce the foundation of such an approach to event logging.

5. FLOWTHING MODEL

The flowthing model (FM) has been used in several applications. This section provides a review of the basic model [1, 2, 3, 4].

FM is a uniform method for representing things that "flow," i.e., things that are exchanged, processed, created, released, and transferred. "Things that flow," denoted as *flowthings*, include information, materials (e.g., manufacturing), money, etc. In economics, "goods" can be viewed as flowthings that are received, processed, manufactured (created), released, and transported. Information is another type of flowthing that is received, processed, created, released, and communicated.

The state transition diagram of flowthings is shown in Figure 2. The model uses *flow of flowthings* as a fundamental notion. Different types of flow can trigger each other.

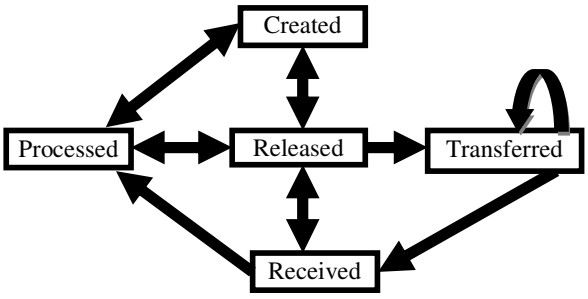


Figure 2. Flowthing states in FM.

6. FLOW-BASED LOGGING MODEL

FM provides a conceptual foundation for the classification of information-related events. Information-related events are those that deal specifically with the *handling* of information as follows:

- Receiving information events
- Processing information events
- Creating information events
- Releasing information events
- Transferring information events

To illustrate such a classification of events and its effect on the logging trail mechanism, we compare it with sample methodologies utilized by the Joint NEMA/COCIR/JIRA Security and Privacy Committee (SPC) [12] that identify audit entries pertaining to access to personal health information (PHI). The SPC methodology "describes how auditing in a medical IT environment, including electronic medical devices, can effectively meet legal mandates and provide the individual accountability and anomaly detection called for in privacy and security regulations" [12]. According to SPC, "the following events should be entered in the audit trail at the discretion of the Security Administrator."

6.1. Create Events

PHI Create events cover system or user actions resulting in the creation of PHI, such as:

- Creation of records that contain PHI (e.g., images, input of data records, patient histories, billing and insurance data).
- Import of records that contain PHI. [12]

This description of events that "should be entered in the audit trail" refers to the event of inputting information from outside the system. Viewed from FM perspective, input of data records to the system is the event of *receiving* information, even though the data are received from staff or health workers (each of whom has his/her own five-stage schema). For staff and data workers, the event is *creating/releasing/transferring* information, but for the system it is *receiving* information, as shown in Figure 3.

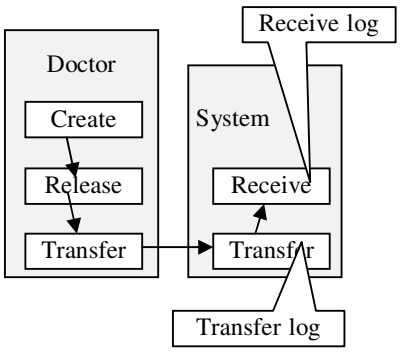


Figure 3. What is visualized as "Create event" by SPC
is a complex of events in FM.

The figure shows two possible events conducted by a doctor as an example of a supplier of information: The event of *transferring* information, and the event of *receiving* information. As examples of points of monitoring these operations, consider the server/system side. After the

doctor logs in, his or her *transferring* of information (e.g., a file) does not necessarily imply receiving that information. Many circumstances (e.g., communication errors, attached file size) may cause nonreceipt of the content of a message. In certain situations, the act of communication is more important than the content of the message. The SPC description of events in the audit trail does not make such a fine scrutiny of information flow because it is not based on a systematic account of the sequence of involved events.

"Creating events" also includes actions resulting in the *creation* of new information, such as information generated by data mining programs (e.g., *John is a risk*), or information generated internally, such as data fed directly from measurement devices, e.g., a thermostat. Suppose the system *creates* a diagnostic record, as shown in Figure 4. This event involves processing (previously) stored information (circle 1). In this case, events of *released* information are different from events of *transferred* information. It is possible that information is released to be transferred outside the system; however, the channel of communication is down; thus data are stored in the buffer waiting to be transferred by the channel.
We can see that the SPC description "Creating events" is a complex of events that can be unentwined systematically by utilizing FM.

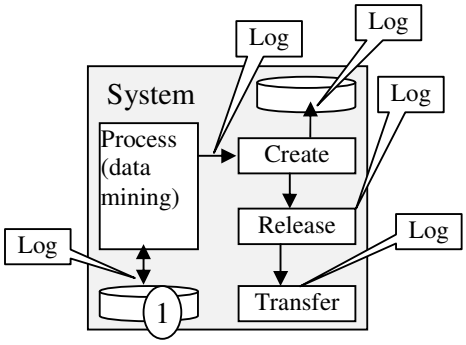


Figure 4. Monitoring created information from the point of retrieving stored information, storing, releasing, and transferring it.

6.2. Modify Events

The Joint NEMA/COCIR/JIRA Security and Privacy Committee (SPC) [12] identifies second audit entries dealing specifically with access to personal health information (PHI) as follows.

> Since PHI contains the history of a patient during medical treatment it is rarely static. New data are added or specific records have to be updated by physicians, nurses, and authorized others; thus the original data are modified during routine work. In case they are modified in an unauthorized or accidental manner, audit files could be the only tools available to reconstruct the original data. Modify events can include:
> - Editing of data (e.g., appending, merging, modifying).
> - Re-association of data.
> - De-identifying of PHI.

As in the previous discussion, "Modify events" can be viewed as a sequence of FM-based events that requires logging in at different stages of the system. "Editing of data," described above, may involve a mere processing of data, as in merging (e.g., merging *John is diabetic* and *John has high blood pressure* creates *John is diabetic and has high blood pressure*), or it may involve creation of new data, as in modifying information. The SPC description "Modify events" does not distinguish between such cases.

6.3. View Events

The third type of event mentioned in the SPC report [12] is defined as follows:

> The simple act of viewing PHI can lead to a compromise of its confidentiality if viewed by any inappropriate person. To be able to reconstruct a record of which individuals accessed what PHI if an investigation becomes necessary, many events should be logged, falling into the following three categories:
> - Access to PHI by any user.
> - Export of PHI to digital media or network.
> - Print or fax of PHI.

Thus "View events" implies retrieval of data records to be viewed or transferred outside the system, as shown in Figure 5.

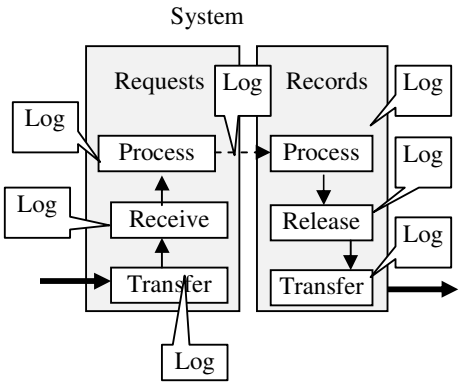


Figure 5. Access to information is a complex event that involves a trail of events that can be logged.

In the figure, there is first a request (for a data record) that flows in the system (left solid arrow), is received, and processed. Assuming the request is approved, the processing stage *triggers* (dashed arrow) the flow of a record, starting with processing of the requested record (retrieving it from storage), releasing it from the system, and transferring it through a communication channel. The FM-based description of events in these cases is a more refined picture of the sequence of events. Rules can be different at different stages. For example, privacy enhancing systems include various rules for restrictions on the handling of personal identifiable information for purposes of collecting, disclosing, or releasing of information; therefore, many events are logged according to these rules. The rule for gathered received information from outside may permit disclosure to its proprietor (person to whom the information refers) so he/she can verify its correctness, while created information (e.g., internal evaluation of the person) may not be disclosed for security reasons.

7. GENERAL MODEL

It is now possible to outline a log processing scheme according to event classification. At this level of conceptual description we do not have to specify format or event processing rules;

however, every log type requires different configuration parameters. We can configure log parameters for any information flow system. Classification here is based on information flow in different stages of the flow stream. In addition, event rules can be classified according to the flow stream. For example, the flow model allows drawing rules with regard to such issues as classifying the severity/sensitivity of processed events. In privacy enhancing systems, creating personal information is generally a more sensitive event than processing it. In addition, releasing such information to outsiders is a more privacy-significant event than receiving it. In such cases it is possible to configure a type of filtering mechanism that would apply to these types of events; e.g., the rule: the event of releasing personal identifiable information to an outsider triggers sending an e-mail to alert the proprietor of that personal identifiable information, as required by some privacy regulations.

Figure 6 illustrates log processing, classification, and actions.

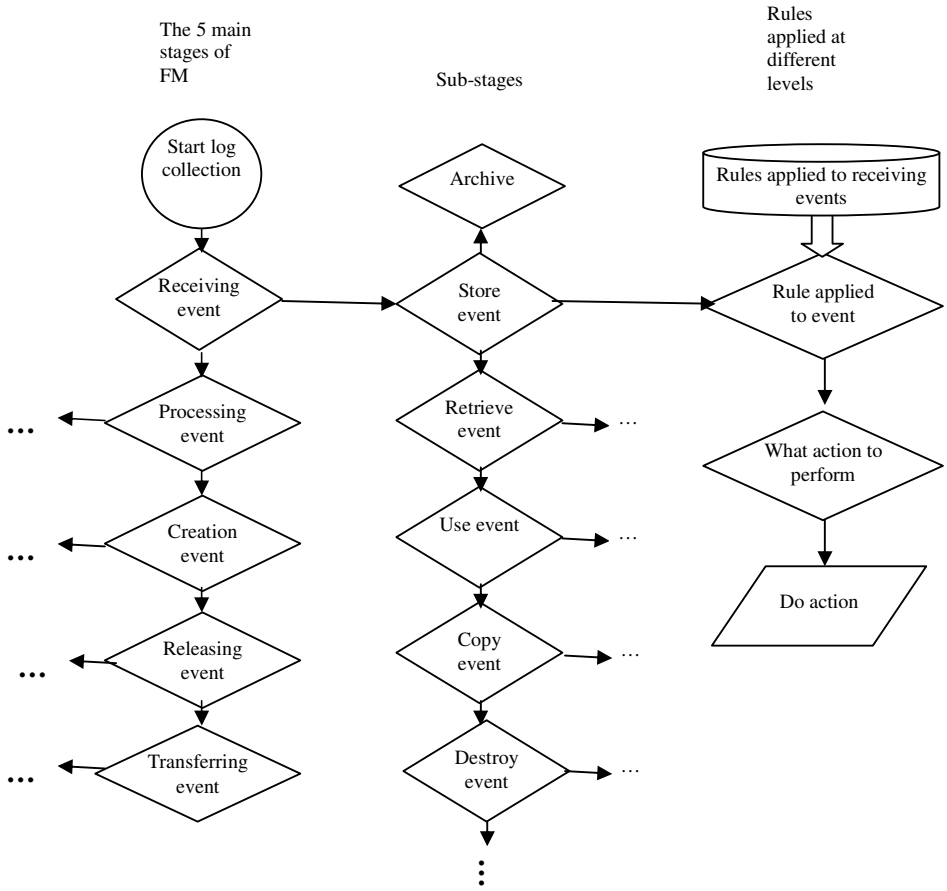


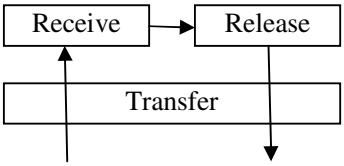
Figure 6. Log processing, classification, and actions.

The figure does not present the details of the processing. From "start of log collection," the event is classified according to stages of FM. The *receiving* stage is expanded, where the event is categorized according to the substages at the receiving stage, such as substages involved in storing, retrieval, use, copy, and discarding or destruction of received information. A sample stream of flow is: personal information is *received* by the front desk of a company, passed to the processing department, after a *copy* is made that is *stored* in the reception desk area.

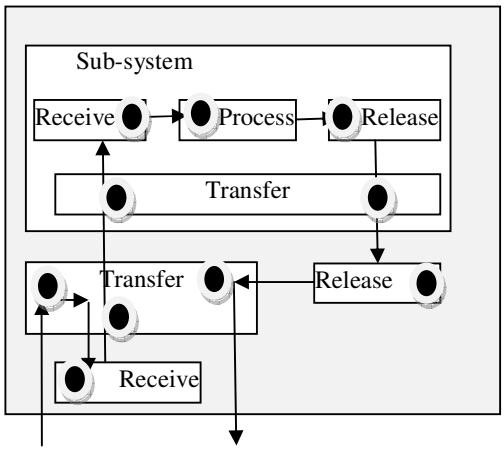
The figure also expands the "Store event" substage by searching for applied rules for "stored received information." For example, in case of received personal identifiable information, the rule can be: if information is to be stored, then the action involves appending a retention period to its record, as required by some privacy regulations.

8. APPLICATION TO MULTI-SYSTEMS

IT infrastructure systems in network, operating system, applications, databases, and even hardware appliances can be modeled as FM schemes, or systems that receive, process, create, release, and transfer information. Examples of information flows are *requests* for log-in, log-out, update database, run software (process that invokes another process), communicate information, and so forth. As samples of such modeling, Figure 7(a) shows a "dump terminal" function in terms of FM. Figure 7(b) shows flow in a subsystem. The black circles indicate different events that can be logged.



(a)



(b)

Figure 7. Information flow in a system and between a system and subsystem.

Consequently, it is possible to design an audit trail of events that provides a uniform, structured view of all the events captured by the system. A closer look at contents of log files at each record level (i.e., entries of audit trail) produces a generic mapping of the five stages of FM. The log contents can be grouped and categorized into the stages of the information flow model, as follows:

Creating Stage: Each event that triggers creation of new information is classified as "CREATE" event, e.g., creating a new account or new file, launching a new application, establishing a connection to a host server or new device, updating of different types of files.

Receiving Stage: Each event that triggers receipt of information is classified as a "RECEIVE" event, e.g., user requests, SQL statements to be executed, data packets in network buffer.

Processing Stage: Each event that triggers a process that changes the form of information is classified as a "PROCESS" event, e.g., operations that compress, translate, encrypt existing information, change the format of the file, or any operation that changes the format without producing new information.
Released Stage: Each event that triggers the release of information to outsiders is considered a "RELEASED" event, e.g., print reports, send files to communication module.
Transfer Stage: Each event that triggers the flow of information from the current system boundary to/from an external boundary is considered a "TRANSFER" event, e.g., when file is transmitted through communication channel, arrival of ad-hoc queries through channel.

9. EXPERIMENTATION

To demonstrate application of the FM-based classification of events, a sample log file was extracted from the university logging system of Microsoft SQL Server 2008 that aims to construct a backup database. The resultant logged events are parsed for entries of actual execution to classify them according to the five stages of FM. Table 1 shows a view of the output screen, including tracked events in the SQL Server environment of the server level activities and also database activities.

Event Date	Event Time	Server Instance Name	Action ID	Class Type	Succeeded	Database Name	Object Name
31/05/2009	10:02:17	USER-RI9NV80QQZ	ALTER	DATABASE	TRUE	CPE512	CPE512
28/05/2009	9:41:01	USER-RI9NV80QQZ	AUDIT SESSION CHANGED	SERVER AUDIT	TRUE		
27/05/2009	10:54:40	USER-RI9NV80QQZ	BACKUP	DATABASE	TRUE	Northwind	Northwind
26/05/2009	11:10:50	USER-RI9NV80QQZ	CREATE	SQL LOGIN	TRUE	master	cpe512
26/05/2009	10:04:02	USER-RI9NV80QQZ	CREATE	USER	TRUE	CPE512	
26/05/2009	10:02:17	USER-RI9NV80QQZ	CREATE	DATABASE	TRUE	master	CPE512
25/05/2009	10:59:00	USER-RI9NV80QQZ	DELETE	TABLE	TRUE	Northwind	Customers
24/05/2009	11:04:55	USER-RI9NV80QQZ	DROP	TABLE	TRUE	Northwind	Employees
21/05/2009	11:08:46	USER-RI9NV80QQZ	DROP MEMBER	SERVER ROLE	TRUE	master	dbcreator
21/05/2009	11:04:55	USER-RI9NV80QQZ	GRANT WITH GRANT	TABLE	TRUE	Northwind	Employees
21/05/2009	11:00:46	USER-RI9NV80QQZ	LOGIN FAILED	LOGIN	FALSE		
20/05/2009	11:10:54	USER-RI9NV80QQZ	LOGIN SUCCEEDED	LOGIN	TRUE		
20/05/2009	11:10:54	USER-RI9NV80QQZ	LOGOUT	LOGIN	TRUE		
20/05/2009	11:07:30	USER-RI9NV80QQZ	RESET PASSWORD	SQL LOGIN	TRUE	master	fahad
20/05/2009	11:00:05	USER-RI9NV80QQZ	RESTORE	DATABASE	TRUE	Northwind	Northwind
20/05/2009	10:59:18	USER-RI9NV80QQZ	SELECT	TABLE	TRUE	Northwind	Employees
20/05/2009	10:54:51	USER-RI9NV80QQZ	SERVER SHUTDOWN	SERVER	TRUE	master	
19/05/2009	10:55:03	USER-RI9NV80QQZ	SERVER STARTED	SERVER	TRUE	master	
18/05/2009	10:59:41	USER-RI9NV80QQZ	UPDATE	TABLE	TRUE	Northwind	Employees

Table 1. Sample logged entries.

SQL Server Audit provides centralized storage of audit logs and also facilitates integration with the system. It permits fine-grained auditing that targets an audit to specific actions by a “principal” against a particular object. Audit events can also be written to the Windows

Application log, making it easier to access audit information and allowing consolidation through management applications. The Windows Security log can be specified as a target for audit [17]. The SQL Server Audit can be used to specify auditing (e.g., activity of users, roles, or groups on database objects) down to the table level. The audit log view includes parameters like date and time stamp, database name, type of action, the class of that action, status of the logged action, user credential details, etc., while each row of the audit view includes an instance of an event captured by the audit log system.

In this experiment, we have identified FM classification of logged events according to each event type column (created, processed, received, disclosed, communicated) using an asterisk (*), as shown in Table 2.

Date Time	Action ID	Class Type	Database Name	Created	Processe	Received	Disclosed	Communicated
5/31/2008 10:49	LOGIN SUCCEEDED	LOGIN			*	*		
5/31/2008 10:49	LOGOUT	LOGIN			*	*		
5/31/2008 10:49	VIEW SERVER STATE	SERVER	master		*	*		
5/31/2008 10:49	LOGIN SUCCEEDED	LOGIN			*	*		
5/31/2008 10:47	SERVER STARTED	SERVER	master	*	*			*
5/31/2008 10:47	AUDIT SESSION CHANGED	SERVER AUDIT			*	*		
5/18/2008 11:19	SERVER SHUTDOWN	SERVER	master					
5/17/2008 11:10	CREATE	SQL LOGIN	master	*	*	*		
5/17/2008 11:08	DROP MEMBER	SERVER ROLE	master		*			
5/17/2008 11:07	RESET PASSWORD	SQL LOGIN	master	*	*	*		
5/17/2008 11:04	DENY	TABLE	Northwind		*	*		
5/17/2008 11:04	GRANT WITH GRANT	TABLE	Northwind		*	*		
5/17/2008 11:01	CREATE	SQL USER	Northwind					
5/17/2008 11:00	LOGIN FAILED	LOGIN			*	*		
5/17/2008 11:00	RESTORE	DATABASE	Northwind	*	*	*		*
5/17/2008 10:59	UPDATE	TABLE	Northwind	*	*	*		*
5/17/2008 10:59	SELECT	TABLE	Northwind		*	*	*	*
5/17/2008 10:59	DELETE	TABLE	Northwind	*	*	*	*	*
5/17/2008 10:54	BACKUP	DATABASE	Northwind	*	*	*	*	*
5/17/2008 10:02	ALTER	DATABASE	CPE512	*		*	*	
5/17/2008 10:02	CREATE	DATABASE	master	*	*	*		*
5/17/2008 9:50	DROP	TABLE	Northwind		*	*		*

Table 2. Detailed, customized view of log table.

For example,
Row Number 4 [LOGIN SUCCEEDED] is of type *received* information, i.e., a user request to access the system is accepted according to credentials, e.g., user name and password.
Row Number 5 [SERVER STARTED] is of type *created* information, i.e., creating a database session.
Row Number 6 [AUDIT SESSION CHANGED] is of type *processed* information, i.e., whenever a change occurs in the state of the audit, like create, modify, or delete, it is captured. In addition, the audit includes any changes in "audit specification"; i.e., any processing of the audit is also audited.

Row Number 17 [SELECT] is of type *disclosed* information; i.e., the execution of SQL SELECT statement results in populating a dataset released to user. For example, the statement SELECT * FROM EMPLOYEES discloses all employees' data, including sensitive data.
Row Number 19 [BACKUP] results in *transferred* information, i.e., transferring database files from their original location to another location like media storage. This would establish a communication channel between source (say, disk storage) and destination (say, tape storage)
Table 2 shows a derived version of the original audit entries, where columns show the date and time stamp of the audited event, the "action ID" (event name), class type (event group), database name of logged events, and five stages flow model.
This shows that FM-based classification of events can be feasibly applied in current systems. Categories can be compared and assigned priorities.

SQL Server 2008 logged "complex" events can be examined to determine basic FM stages embedded in these events. In addition, each FM stage can be subclassified into secondary events such as *storing received* information, *releasing received* information, *copying received* information, etc. A mapping between locations of logical FM-based events and actual events in server's components can be devised.

The implication here is that it is possible to construct a version of an auditing system in SQL Server 2008 that monitors and logs events in the more systematic FM way. Such a venture would result in a system design mirroring the logical implementation-free FM description. The benefits of such a mapping are enormous, including—to mention just one—providing a complete audit-based language to nontechnical people. Thus, instead of SELECT event, the event is described in terms of retrieve created (or previously acquired) data, release it (from system), and transfer it somewhere.

10. CONCLUSIONS

Audit logging is an important component in protection of information systems from unauthorized access, process, destruction, or disclosure. In this paper, we have shown that the information flow model can be applied to an audit log system, where audited events are categorized according to different implementation-independent stages. The log system of Microsoft SQL Server 2008 is used as a case study to show the feasibility of such a categorization of events. The resulting classification of events suggests embedding stages of the flow model in the design of the audit/log system.

REFERENCES

[1] S. Al-Fedaghi, (2010) "Threat risk modeling", 2010 International Conference on Communication Software and Networks (ICCSN 2010), Singapore, 26-28 February.

[2] S. Al-Fedaghi, (2009) "On developing service-oriented Web applications", The 2009 AAAI (Advancement of Artificial Intelligence)/IJCAI (International Joint Conferences on Artificial Intelligence) Workshop on Information Integration on the Web (IIWeb:09), 11 July, Pasadena, California, USA. Paper: **http://research.ihost.com/iiweb09/notes/9-P4-ALFEDAGHI.pdf**

[3] S. Al-Fedaghi, (2009) "Flow-based description of conceptual and design levels", IEEE International Conference on Computer Engineering and Technology, Singapore , 22-24 January.

[4] S. Al-Fedaghi, Kh. Al-Saqabi, & B. Thalheim, (2008) "Information stream based model for organizing security", Symposium on Requirements Engineering for Information Security (SRIS 2008), Barcelona , Spain, 4-7 March.

[5] S. Al-Fedaghi, Ala'a Alsaqa, & Zahra'a Fadel, (2009) "Conceptual model for communication", *International Journal of Computer Science and Information Security*, Vol. 6, No. 2. http://arxiv1.library.cornell.edu/ftp/arxiv/papers/0912/0912.0599.pdf

[6] AIX, (1999) *System management concepts: Operating system and devices*. Chapter 3, "Auditing Overview". 1st ed. (September). http://www.chm.tu-dresden.de/edv/manuals/aix/aixbman/admnconc/audit.htm

[7] J. Anderson, (1973) "Information security in a multi-user computer environment," *Advances in Computers*, Vol. 12. New York: Academic Press, pp. 1-35. (I A1, SFR) http://books.google.com.kw/books?id=vg_QRDVR7hgC&pg=PA1&lpg=PA1&dq=Anderson,+%22Information+security+in+a+multi-user+computer+environment,%22++Advances+in+Computers&source=bl&ots=5I5-m8a9UE&sig=lXJ6QW_0yHEp9tsj7UcEw6gn2lw&hl=en&ei=bHHUStOVIJySsAay9fTcCw&sa=X&oi=book_result&ct=result&resnum=1&ved=0CAgQ6AEwAA

[8] Becta, (2009, March) "Good practice in information handling: Audit logging and incident handling", Becta, V2. http://schools.becta.org.uk/upload-dir/downloads/audit_logging.doc

[9] GFI, (2008) "Auditing events". http://docstore.mik.ua/manuals/hp-ux/en/5992-3387/ch10s04.html

[10] GFI, (2009, Oct.) *EventsManager*, Manual. GFI Software Ltd. http://www.gfi.com/esm/esm8manual.pdf

[11] S. E. Hansen & E. T. Atkins, (1993) "Automated system monitoring and notification with swatch", *Proceedings of the Seventh Systems Administration Conference (LISA VII)*, p. 145.

[12] Joint NEMA/COCIR/JIRA Security and Privacy Committee, (2001) "Security and privacy auditing in health care information technology". http://www.medicalimaging.org/documents/Security_and_Privacy_Auditing_In_Health_Care_Information_Technology-November_2001.pdf

[13] P. Kazienko & P. Dorosz, (2004) "Intrusion detection systems (IDS) Part 2 - Classification; methods; techniques", WindowsSecurity.com. http://www.windowsecurity.com/articles/IDS-Part2-Classification-methods-techniques.html

[14] W. Kelly, (2006, Dec. 29) "Demystifying event logs: Put event logs to work for better security & compliance", *Processor*, Vol. 28, No. 52. http://www.processor.com/editorial/article.asp?article=articles%2Fp2852%2F31p52%2F31p52.asp

[15] Tao Li, Feng Liang, Sheng Ma, & Wei Peng, (2005) "An integrated framework on mining logs files for computing system management", *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD '05)*, pp. 776–781.

[16] Microsoft Support, (2007) "How to view and manage event logs in Event Viewer in Windows XP", Article ID: 308427 - Last Review: May 7 - Revision: 3.3. http://support.microsoft.com/kb/308427

[17] Il-Sung Lee & Art Rask, (2009) "MSDN SQL Server Developer Center, Auditing in SQL Server". http://msdn.microsoft.com/en-us/library/dd392015.aspx

[18] SANS Consensus Project, (2007) "Information system audit logging requirements", SANS Institute http://www.sans.org/resources/policies/info_sys_audit.pdf

[19] D. Shackleford, (2008, April) "Leveraging event and log data for security and compliance", SANS Whitepaper. http://www.sans.org/reading_room/analysts_program/leveraging_data_April08.pdf

[20] J. Stearley, (2004) "Towards informatic analysis of syslogs", *Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER '04)*, pp. 309–318.

[21] T. Takada & H. Koike, (2002) "Mielog: A highly interactive visual log browser using information visualization and statistical analysis", *Proceedings of the 16th USENIX Conference on System Administration (LISA '02)*, pp. 133–144.

[22] R. Vaarandi, (2002) "Sec - a lightweight event correlation tool", *Proceedings of IEEE International Workshop on IP Operations and Management (IPOM '02)*, Dallas, Texas, USA, 29-31 October.

[23] W3C Working Draft 8, (2003, August) *Web Services Architecture.*

Sabah Al-Fedaghi holds an MS and a PhD in computer science from Northwestern University, Evanston, Illinois, and a BS in computer science from Arizona State University, Tempe. He has published papers in journals and contributed to conferences on topics in database systems, natural language processing, information systems, information privacy, information security, and information ethics. He is an associate professor in the Computer Engineering Department, Kuwait University. He previously worked as a programmer for the Kuwait Oil Company and headed the Electrical and Computer Engineering Department (1991–1994) and the Computer Engineering Department (2000–2007).

Fahad Mahdi holds a BS in computer engineering from Kuwait University (1999). He is currently a graduate student in the same department, and working at the Research Administration of Kuwait University as a database developer/administrator.