Study on Audit for Distributed Applications: a Proposal for Simplified Audit Module Architecture

Anca APOSTU
Bucharest University of Economic Studies
ROMANIA
ancaiapostu@gmail.com

Abstract: The tracking and audit services of an informatic system provide the possibility to record the actions performed within the system and the states of the objects, modules and of the application itself. In order to assure a scalable and secure tracking for the auditing service it is important to define the basis of the record process itself, to determine efficient ways to store and process the audit data, to improve the performance and functions that enable a better understanding and use of the event data recorded. The paper presents the main requirements during auditing process in the context of informatic systems and proposes a simplified solution for implementing them in software distributed systems.

Key-words: distributed systems, software audit, software quality, audit module.

1. Introduction

Considering nowadays economic challenges, the economic agents using financial and accounting systems are obliged to perform a periodical audit of the mentioned systems. The internal audit includes also the informatic audit(the audit of information technology and communications) i.e. the audit of systems(applications).

Current activity leads to giving up on the basic meaning of audit for the financial-accounting side of the economic activity and raising the importance of audit through including the function of internal audit under the direct coordination of the economic agent becoming a strong instrument for discovering and monitoring the risks. In the same idea, the audit of informatic systems should be regarded as the audit of the integrated software infrastructure[5] of the economic agent which also contains the financial subsystem.

The internal audit, including also the informatic audit, is monitored by a large number of national organisations, based on geographical or linguistic criterias such as The Francophone Union of Internal Audit and The European Confederation of Internal Audit Institutes[6].

According to The European Confederation of Internal Audit Institutes, Internal Auditing is defined as an independent, objective assurance and consulting activity designed to add value and improve an organisation's operations. “It helps an organisation accomplish its objectives by bringing a systematic, disciplined approach to evaluate and improve the effectiveness of risk management, control, and governance processes.”[6] “Independence and objectivity are two critical components of an effective internal audit activity.”[6]

Distributed networks are unique in that they bring together concepts of communication, engineering, and computing [4]. From an audit standpoint, the complexities involved in control design and testing are challenging. The auditor needs to have strong knowledge concerning these complexities in order to apply the proper audit tools, particularly since some of these tools are in need of improvement or development. This paper provides a summary of important areas of audit concerning distributed processing systems.

Processes are said to be concurrent if they run at the same time, and concurrency gives rise to a number of well-studied problems. Processes may use old data; they can make inconsistent updates; the order of updates may or may not matter; the system might
deadlock; the data in different systems might never converge to consistent values. Programming distributed systems is a difficult problem in general; and, unfortunately, most of the examples come from the relatively rarefied world of operating system internals and thread management. But concurrency control is also a security issue; like access control, it exists in order to prevent users interfering with each other, whether accidentally or on purpose. Also, concurrency problems can occur at a number of levels in a system, from the hardware right up to the business environment.

2. Motivation for implementing audit for informatics systems

Many secure distributed systems have incurred huge costs or developed serious vulnerabilities, because their designers ignored the basic lessons of how to build (and how not to build) distributed systems. Most of these lessons are still valid, and there are more to add. A large number of security breaches are concurrency failures of one kind or another; systems use old data, make updates inconsistently or in the wrong order, or assume that data are consistent when they aren't and can't be. Knowing the right time is harder than it seems. Fault tolerance and failure recovery are critical. Providing the ability to recover from security failures, and random physical disasters, is the main purpose of the protection budget for many organizations. At a more technical level, there are significant interactions between protection and resilience mechanisms. We need to protect not just against failures and attempted manipulation, but also against deliberate attempts to deny service, which may often be part of larger attack plans. Many problems also arise from trying to make a name do too much, or making assumptions about it which don’t hold outside of one particular system, or culture, or jurisdiction. For example, it should be possible to revoke a user’s access to a system by cancelling their user name without getting sued on account of other functions being revoked. The simplest solution is often to assign each principal a unique identifier used for no other purpose, such as a bank account number or a system logon name. But many problems arise when merging two systems that use naming schemes that are incompatible for some reason. Sometimes this merging can even happen by accident—an example being when two systems use a common combination such as “name plus date of birth” to track individuals.

3. Main requirements for software audit

In the research community, secure distributed systems tend to have been discussed as a side issue by experts on communications protocols and operating systems, rather than as a discipline in its own right. There are many technical issues to be mentioned such as how the design of secure time protocols is done and the complexities of naming. But perhaps the most important research problem is to work out how to design systems that are resilient in the face of malice, that degrade gracefully, and whose security can be recovered simply once the attack is past. This may mean revisiting the definition of convergent applications.

Under what conditions can we recover neatly from corrupt security state? Do we have to rework recovery (which explores how to rebuild databases from backup tapes)? What interactions are there between recovery mechanisms and particular protection technologies? In what respects should protection mechanisms be separated from resilience mechanisms, and in what respects should they be separated? Generally, the audit applications should consider two types of requirements: the functionality requirements and security related requirements[1] and should implement the following minimum requirements: tracking the flow of activities performed within the system; recording object states; providing means
to retrieve audit data; providing methods for tracking the history of an object; allowing the detection of any external change of data; storing audit data in a secure manner; assuring continuous audit activities; providing compliancy and integration with standards.

The implementation of audit modules has known two feasible solutions:
- The integration with an already existing application, also known as an external audit system;
- A set of interfaces and rules implemented by the application for invoking audit methods, i.e. an internal audit system.

4. Modeling an architecture for an audit system

In this section the main directions regarding the development of high-level architecture of an audit system are presented. The structuring process starts with the identification of a set of criteria. The audit will involve the following criteria:

1. **Maintenance strategy.** It analyses the existence of a defined maintenance strategy in accordance with the global strategy of the organization. The maintenance strategy and the level of application of each element must be established. The estimated influence of the maintenance area over the organization must also be established. Therefore, objectives from the maintenance department must be recorded.

2. **Attitude and perception.** It evaluates the behavior amongst the maintenance department personnel and between them and personnel of other areas of the entity, safety and quality departments.

3. **Resources/Facilities management.** It evaluates the level of integration of the maintenance information system with the Computer Maintenance Management System (CMMS) or Enterprise Resource Planning (ERP). It evaluates other hardware required to transfer the maintenance data.

4. **Human resources management.** It includes the job roles, responsibilities, incentives, and training aspects. It evaluates the organizational structure.

Paper [1] presents a detailed architecture containing interfaces of an audit module and the relationship between it sub-modules. On a general basis, an audit module must record and process two types of data:

- **Action execution stage (Action execution):** starting from the action executed by an user at certain time, together with the basic information, the system must identify a record containing the type of the action relevant for the audit process and also other details such as: the context of the execution (moment of execution, network configuration, application status), the reason/cause of the action, the result(s) of the action and the status (successful execution, partial successful execution, failure);

- **Degree of object status change (Object modification):** for the case in which the status of an object is modified at the same time regardless of the source of the modification (e.g. user, third-party application, processing routine), the audit module must identify the stages of the action which determined the change (the action execution stages defined at the previous point: success, partial success, failure).
Starting from source [1], this study proposes a simplified architecture (as shown in diagram 1) for an audit module to be applied and integrated in current software development. Based on the mentioned simplified diagram, we will introduce the functionality of each sub-module within the audit module:

- **Audit and reporting interface** – this interface combines methods for recording the degree of an action execution or of a change in an object’s status with audit reporting functionality. This interface is to be mainly used by the application user. Based on the fact that an audit module is to be developed as a part of a software system, it is necessary to integrate such a module under a user friendly interface. This interface must ensure the explicit call of the audit module and also be flexible and stable enough to fulfill all requests depending on the type and number of the actions and of the existing objects. In productive systems, the quantity of data stored and processed is of big dimensions, therefore the main challenge is to filter relevant data and to give the possibility to generate reports based on multiple criteria: complete activity of one or more users, analysis upon different states of the same object at given moments of time, the complete history of one object etc.

- **Action audit sub-module** – this sub-module implements the audit interface by providing recording methods of relevant actions related, adding also the user information and also the result of the action;

- **Sub-module for object auditing (Object audit sub-module)** – performs the recording of the object status before and after the modification and also the possibility to encrypt/decrypt stored data regarding objects. The implementation of the current sub-module can affect the general performance of the audit module and also the one of the system and this must be considered in the phase of module design as it can influence the reporting functionality as well.

- **Sub-module for tracking of changes (Tracking sub-module)** – this module implements standard interrogations (queries) for retrieving relevant information; it also provides the possibility to generate reports
and user-defined queries or customized queries;

- **difference sub-module** – the sub-module identifies and reports the differences between two states of an object at given moments of time;

- **sub-module for initialization and configuration** – configuration and initialization are important steps in the life cycle of an informatic system, therefore they come as an important section for an audit module as well. Therefore it is necessary to design this sub-module following the integration with the main system(host) in order to ensure optimal performance and compliance with audit standards. Current module is also responsible with the management of credentials for secured connections to storage system(s) of audit data; i.e. it will implement high-end features of audit sub-module. As only a limited set of activities and objects is relevant for audit process, the configuration of this module will provide filtering features from this point of view. Config sub-module is responsible also for initializing the database structure for auditing process using configuration files and metadata.

Information system audit and control methodologies have come a long way from a small beginning fifty years ago. In a competitive flat world, large non-governmental enterprises affect the society in so many ways that the failure of a large enterprise is not only an issue of interest to its shareholders or its employees but also to the society, at large. The global financial meltdown has shown that ordinary tax-payers have to step in to save such large enterprises by infusing public funds. Today there is no objective metric to measure whether an enterprise has behaved responsibly and whether public funds should be used to save it. It is often proposed that the annual system audit/security function for an enterprise should be expanded to include a focused report on how socially responsible the enterprise has been during the year. For any enterprise, the auditing process today requires three types of audit. The *conventional audit* requires an evaluation of the financial status of the enterprise by defining the gross and net profit of an enterprise. The *second dimension* refers to compliance with various other legal requirements, besides the requirement of working out the financial balance sheet of an enterprise. The *third dimension* is of cyber-security. It is driven by privacy laws for customers’ data and by needs for security of crucial data of an enterprise. To the three dimensions of system audit of Security, Effectiveness, and Compliance we need to add the *fourth dimension of social responsibility*[3]. In order to deliver value, Information systems today need to perform and measure up to evaluate enterprises along this fourth dimension also. This is important for both the enterprises in the relatively poor, backward, less privileged, developing countries as well as for enterprises in the developed world. By using a baseline to measure the impact of actions for socially responsible behavior and audit them in terms of social responsibility norms can make it possible to evaluate enterprises in an objective manner.

The Audit module of applications provides a general image of the full history of the system’s life. This recorded information may be used at any point in time to analyze and prevent system failures, problems or vicious states, to check the reliability of the system and to assure methods for centrally coordinating, collecting and storing audit data in a distributed environment. As audit process provides a strong basis for evaluation of the application and of the activity of an organization, currently, “worldwide companies consider the audit of the systems they use a mission-critical function” [1].

The process of implementing an audit module should consider covering most of the features requested by an informatic system, should continuously improve the usability and integration of the audit
module, should ensure a high level availability and performance of the audit module also considering indicators of system ineffectiveness:
- excessive down time and idle time,
- slow system response time,
- excessive maintenance costs,
- inability to interface with new hardware/software,
- unreliable system outputs,
- slow system response time,
- data loss,
- excessive run costs,
- frequent need for program maintenance and modification,
- user dissatisfaction with output format, content or timeliness.

The financial and internal audit is based on data and information extracted and stored electronically into the information system of the economic agent. The mentioned types of audit do not exclude informatic audit which requires conceptualization, regulation and implementation. Romania’s Chamber of Audit has assimilated the International Standards for Internal Audit, including also the ones for IT systems CIS (Computerized Information Systems).

The need of control and audit for informatic systems, reflected also by law and normatives, is determined by the threats and vulnerabilities for informatic systems used by economic agents for all sorts of activities, that is in the electronic business administration.

All the threats and vulnerabilities of informatic integrated systems of the economic agents generate risks and affect information security and are now the object of controls and informatic audit. Once implemented, an informatic system needs to be periodically audited in order to ensure the requirements at the highest efficiency level. The increase of the organization, the growth of the business, the changes in the business area, the technical evolution and the new informational requests place a growing demand over the informatic system and impose changing and extending it.

Some considerations need to be taken care of for the audit of an informatic system: reevaluation of informational requests; checking of modifications for the existing basis projects; discovering new technical opportunities; improving operation procedures. The audit of informatic systems should be done once in three years or as often as the implemented changes require this.

5. Conclusions

For many applications, access control and other business related information of all user transactions should be kept in secure log files for intrusion and misuse detection or system audit purposes. Because the log files may be stored on or moved to an untrusted machine and may attract attackers because of the large amounts of potentially sensitive information contained in them, it would be safe to guarantee that in the event an attacker gains access to this machine, one can limit his ability to corrupt the log files and is able to detect any compromises afterwards.

An important aspect in audit is continuity of the process during the life cycle of an application. The audit module should ensure no gaps during the time of supervising the system’s activity, giving the possibility to analyze system failures, problems or states, to check the reliability of the system and to assure the application is working in compliance to certain quality standards.

The current approaches should enable a wide-ranging solution for controlling the resources of distributed enterprise applications.

This approach reflects the requirement for evaluations to be tailored to the needs of individual organizations based on their environment, the context of the evaluation, what is to be evaluated and recognition of stakeholders. These will be the subject of further research.

Acknowledgement

Parts of this research have been published in the Proceedings of the 11th International Conference on Informatics in Economy, IE 2012.
References


