Principles of Computer Security

Alvin Lin

January 2018 - May 2018

Database Auditing

A security audit is an independent review or examination of system records and activities for the purpose of:

- Determining the adequacy of system controls
- Ensuring compliance with established security policy and procedures
- Detecting breaches in security services
- Recommending any changes needed for countermeasures

The basic object of an audit is to establish accountability for system entities that initiate or participate in security-relevant events and actions. Means are needed to generate and record a security audit trail and to review and analyze the audit trail to discover and investigate attacks. Usually this is done by having a chronological record of system activities enabling the reconstruction and examination of a sequence of environments and activities surrounding or leading to an operation, procedure, or event in a security relevant transaction.

Events Subject to Audits

In order to gain useful insights, we need to define the set of events that are subject to audit:

- Introduction of objects
- Deletion of objects

- Distribution or revocation of access rights or capabilities
- Changes to subject or object security attributes
- Policy checks performed by the security software
- Use of access rights to bypass a policy check
- Use of identification and authentication functions
- Security related actions taken by an operator or user
- Import or export of data from/to removable media

Event Detection

Appropriate hooks must be available in the application and system software to enable event detection. Monitoring software needs to be added to the system and to appropriate places to capture relevant activity. An event recording function is needed, which includes the need to provide for a secure storage resistant to tampering or deletion. Event and audit trail analysis software, tools, and interfaces may be used to analyze collected data as well as for investigating data trends and anomalies. Any auditing system should have a minimal effect on functionality.

Implementation Guidelines

- Audit requirements should be agreed upon with appropriate management.
- The scope of technical audit tests should be agreed upon and controlled.
- Audio tests should be limited to read-only access to software and data.
- Access other than read-only should only be allowed for isolated copies of system files.
- Any requirements for special or additional processing should be identified and agreed upon.
- Audit tests that could affect system availability should be run outside business hours.
- All access should be monitored and logged to produce a reference trail.

Database Audit Trail

A mechanism should be provided for the complete reconstruction of every action taken on the database:

- Who: Identify the person viewing/modifying the data
- What: Full listing of data viewed or modified
- When: Reliable datetime-stamp
- Where: Specific application used for access
- Why: Context information on data disclosure

We are particularly interested in who made and approved the transaction, what they did, and what the result was.

Practical Auditing Requirements

In reality, auditing should be robust, comprehensive, efficient, and customizable. Database activity should be audited by statement, use of system privilege, object, and the user. Table creation, database connections, and successful or unsuccessful connections should be monitored for auditing. Auditing should also be implemented efficiently, where statements are parsed once for both execution and auditing, not separately. It should be implemented within the server itself, but stored separately in case of failure. The audit log should be able to filter objects of interest.

It is possible to use triggers to further customize auditing conditions. This allows administrators to define specific audit policies to detect the misues of legitimate data access. This can further help reconstruct audited events to determine whether access rights have been violated and prevents users from bypassing auditing. Categories:

- Accesses, login, sources of usage
- Errors of any kind, which may allow you to determine the intent of the user
- Usage outside of normal hours
- Changes to sensitive data
- Changes to stored procedures and triggers
- Changes to privilieges, user/login definitions, and security attributes

- Creation, changes to, and usage of database links
- Changes to the definition of what to audit

Non-Repudiation

Non-repudiation is the assurance that a party to a contract cannot deny authenticity of their signature on a document. This holds principals accountable for their actions and provides irrefutable evidence about their events or actions. This is increasingly needed due to increased distributed and mobile computing. Typical evidence for non-repudiation involves proof of message creation and proof of the receipt of a message, which also needs proper logging and auditing.

Baselining and DDL Audits

A baseline usage pattern should be developed to monitor unusual changes from the baseline. Changes to the schema as well as changes to the data should be monitored. This can be done through system triggers or other mechanisms.

Auditing Architectures

Auditing architectures should not create a false sense of security. They should provide an independent audit trail. The audit system should be archived, secure, and also itself be auditable. Ideally, it should be sustainably automatable. The data stored from auditing can be massive, and this can present other logistical issues since it needs to be stored to generate reports and alerts in addition to being capable of archival and restoration. This usually implies the need for data warehouse capable of handling dynamic changes as demanded by auditors.

You can find all my notes at `http://omgimanerd.tech/notes`. If you have any questions, comments, or concerns, please contact me at alvin@omgimanerd.tech