

UNIT 8

Security Auditing

(7 Hours) — Information Security (CACS459)

Security auditing is the systematic, independent examination of an information system's records, activities, and controls to determine whether they conform to established security policies, operational procedures, and regulatory requirements. It is both a technical activity — inspecting logs, configurations, and access records — and a managerial one, since its findings inform risk decisions, compliance posture, and continuous improvement of an organization's security programme. This unit covers the audit process itself, the architecture that supports it, how audit trails are generated and stored, how logging functions are implemented in practice, and the techniques used to analyze audit trails once they have been collected.

8.1 Security Audit

A security audit is a structured, evidence-based evaluation of an organization's information system, its policies, and its operational practices, carried out to verify that security controls are correctly designed, properly implemented, and operating effectively. Unlike an informal review, an audit follows a defined methodology, produces documented evidence, and results in a formal report with findings and recommendations.

8.1.1 Objectives of a Security Audit

- Verify compliance with internal security policies, industry standards (e.g. ISO/IEC 27001), and legal or regulatory requirements.
- Identify vulnerabilities, misconfigurations, and weaknesses in technical controls before they are exploited.
- Assess whether access controls correctly enforce the principle of least privilege.
- Evaluate the effectiveness of existing security policies and procedures in practice, not just on paper.
- Provide management with an independent, objective basis for risk-related decisions.
- Establish accountability by creating a documented record of who did what, and when.

8.1.2 Types of Security Audits

Type	Focus	Typical Performer
Internal Audit	Ongoing self-assessment of controls and compliance	Internal audit team / IT security staff
External Audit	Independent verification for certification or regulatory purposes	Third-party audit firm
Compliance Audit	Conformance to a specific standard (ISO 27001, PCI-DSS, HIPAA)	Certified external auditor
Technical / Penetration Audit	Hands-on testing of systems, networks, and applications	Security engineers / ethical hackers
Operational Audit	Day-to-day procedures — backups, patching, access reviews	IT operations / internal audit

8.1.3 The Security Audit Process

Although details vary by organization and standard, a security audit typically follows six broad phases, forming a repeatable cycle rather than a one-time event.

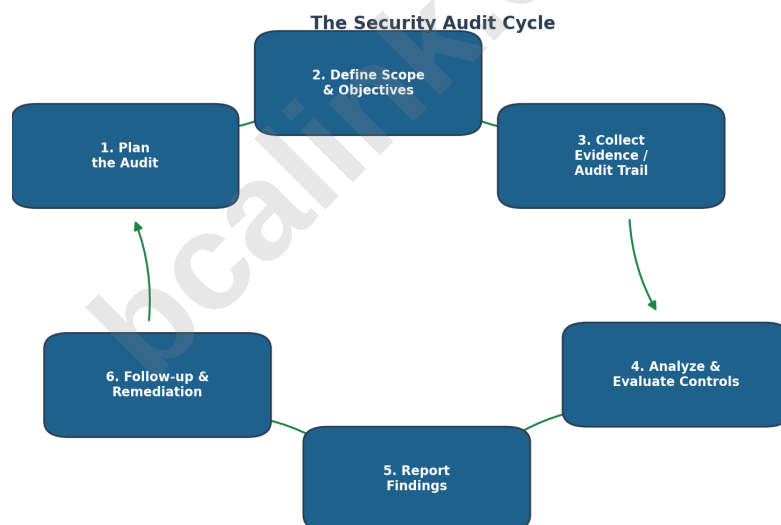


Figure 8.1 — The Security Audit Cycle: planning through remediation.

- Plan the Audit — establish objectives, timeline, resources, and the audit team.
- Define Scope & Objectives — identify which systems, processes, and controls fall within the audit boundary.
- Collect Evidence / Audit Trail — gather logs, configuration files, interview notes, and policy documents.
- Analyze & Evaluate Controls — compare observed practice against the stated policy or standard, looking for gaps.

- **Report Findings** — document non-conformities, risks, and recommended corrective actions.
- **Follow-up & Remediation** — track whether identified issues are actually resolved, closing the loop before the next cycle.

Example 8.1 — Scope Definition

A bank plans an annual security audit of its online banking platform. The scope is defined to include the web application, the authentication server, and the transaction database, but explicitly excludes the branch-office LAN, which is audited separately under a different schedule.

Note: An audit that lacks a clearly defined scope tends to either miss critical systems or consume excessive time and budget reviewing irrelevant ones.

8.2 Security Auditing Architecture

A security auditing architecture is the set of components — collectors, storage, analysis engines, and reporting interfaces — that together enable an organization to capture security-relevant events, preserve them reliably, and turn them into actionable information. A well-designed architecture separates the concerns of collection, storage, and analysis, so that a failure or compromise in one layer does not automatically compromise the others.

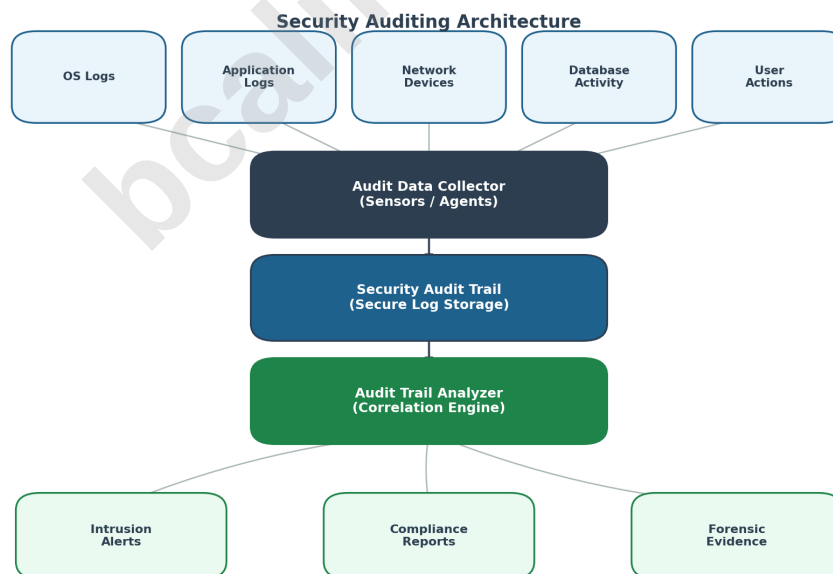


Figure 8.2 — Layered security auditing architecture: from raw event sources to actionable output.

8.2.1 Core Architectural Components

Component	Role
Event Sources	Operating systems, applications, network devices, and databases that generate raw events.
Audit Data Collector	Agents or sensors that capture, normalize, and forward events from each source.
Audit Trail / Log Store	Centralized, tamper-evident repository where normalized records are retained.
Audit Trail Analyzer	Correlation and analysis engine that searches for patterns, anomalies, and policy violations.
Reporting Layer	Dashboards, alerts, and formal reports consumed by security staff and management.

8.2.2 Design Principles

- **Centralization** — collecting logs from distributed sources into one place makes correlation and review far more efficient than checking each system individually.
- **Separation of duties** — the people or systems that generate audit data should not be the same ones that can freely modify or delete it.
- **Redundancy** — critical audit data should be replicated so that a single point of failure cannot erase evidence of an incident.
- **Scalability** — the architecture must handle growing volumes of log data as the organization and its systems expand.
- **Minimal performance impact** — collection agents should not noticeably degrade the performance of the systems they monitor.

Note: A common architectural pattern is the Security Information and Event Management (SIEM) system, which combines centralized collection, correlation, and alerting in a single platform.

8.3 Security Audit Trail

A security audit trail (or audit log) is a chronological, tamper-evident record of system and user activity that allows an analyst to reconstruct exactly what happened, who did it, when it occurred, and — as far as possible — why. It is the primary evidence base for detecting intrusions, investigating incidents, and demonstrating compliance.

8.3.1 What an Audit Trail Should Capture

Field	Purpose
Timestamp	Precise date and time of the event, ideally synchronized via NTP across all systems.
Subject / User ID	The account or process that initiated the action.
Event Type	Login, file access, privilege change, configuration change, etc.
Object Affected	The resource acted upon — a file, record, device, or account.
Outcome	Success or failure of the action, with an error code if applicable.
Source Location	IP address, terminal, or host from which the action originated.

8.3.2 Properties of a Trustworthy Audit Trail

- **Completeness** — all security-relevant events are captured; nothing significant is silently dropped.
- **Integrity** — records cannot be altered after the fact without detection, often enforced with cryptographic hashing or write-once storage.
- **Accountability** — every entry can be traced to a specific subject, supporting non-repudiation.
- **Confidentiality** — access to the trail itself is restricted, since logs often contain sensitive information.
- **Availability** — the trail must be retrievable when needed, including during or after an active incident.

Example 8.2 — Reconstructing an Incident

An administrator notices a customer record was modified without authorization. By querying the audit trail for that record's object ID, the investigator finds a single entry: user 'svc_backup' updated the record at 02:14 AM from an internal IP that does not match the service account's usual host — evidence pointing to a compromised credential rather than a legitimate backup job.

Analysis

The timestamp (unusual hour), the subject (a service account acting outside its normal pattern), and the source location (unexpected host) together form the evidentiary chain that lets the analyst conclude the account was likely compromised, well before a customer complaint might have revealed the tampering.

*Note: An audit trail that records ***what*** happened but not ***who*** or ***from where*** is of limited forensic value — accountability depends on all three being present together.*

8.4 Implementing Logging Function

The logging function is the mechanism that actually converts a raw system event into a durable audit trail entry. Implementing it correctly is as much about disciplined engineering as it is about security policy: a poorly implemented logging function can leave gaps that undermine an otherwise well-designed audit architecture.

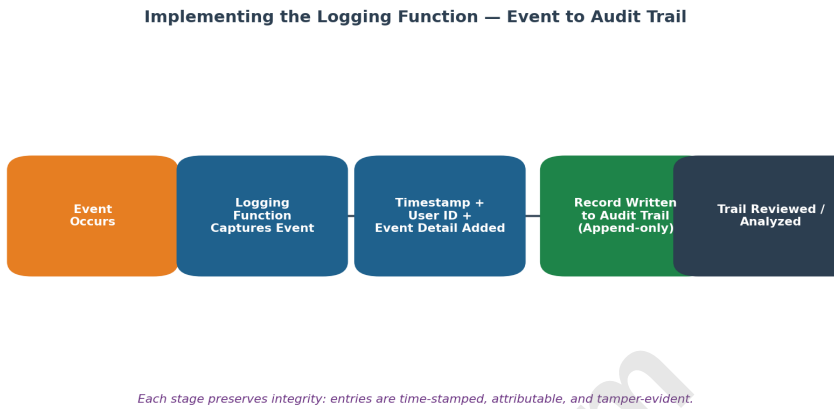


Figure 8.3 — From event occurrence to a reviewable audit trail entry.

8.4.1 Steps in Implementing Logging

- Identify what to log — authentication attempts, privilege changes, access to sensitive data, configuration changes, and administrative actions are typical priorities.
- Standardize the log format — a consistent structure (e.g. syslog, JSON, CEF) makes later correlation and analysis far easier.
- Attach context — every entry should be enriched with a synchronized timestamp, user identity, and source before it is written.
- Protect the log path — logs should be transmitted and stored securely, typically over an encrypted channel to a centralized, access-controlled store.
- Enforce retention — define and follow a retention period that satisfies both operational needs and regulatory obligations.
- Test the logging function itself — periodically verify that expected events are actually being captured, not just assumed to be.

8.4.2 Common Implementation Pitfalls

Pitfall	Consequence
Logging too little	Investigators cannot reconstruct what happened during an incident.
Logging too much / unfiltered	Critical events are buried in noise, and storage costs rise sharply.

Pitfall	Consequence
No time synchronization	Correlating events across systems becomes unreliable or impossible.
Writable/deletable logs	An attacker with sufficient access can erase evidence of their own actions.
Logging sensitive data in cleartext	The log itself becomes a new source of data exposure, e.g. passwords in logs.

*Note: A widely used guideline is to log security-relevant *decisions* (was access granted or denied, and why) rather than every low-level operation, which keeps the trail meaningful without overwhelming it with routine noise.*

8.5 Audit Trail Analysis

Collecting an audit trail has limited value unless it is actively analyzed. Audit trail analysis is the process of examining accumulated log data to detect security incidents, confirm policy compliance, and support forensic investigation after the fact.

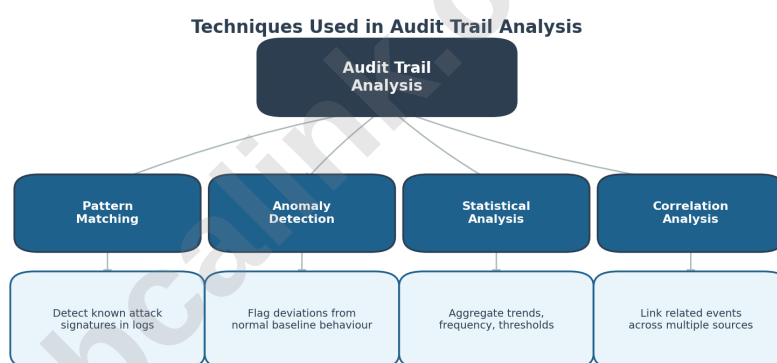


Figure 8.4 — Four complementary techniques for analyzing an audit trail.

8.5.1 Analysis Techniques

- **Pattern Matching** — comparing log entries against known attack signatures, such as a sequence of failed logins followed by a successful one.
- **Anomaly Detection** — establishing a baseline of normal behaviour and flagging significant deviations from it, useful for catching novel attacks that have no known signature.
- **Statistical Analysis** — aggregating events to spot trends, unusual frequencies, or thresholds being crossed, such as a spike in failed access attempts from a single account.

- Correlation Analysis — linking related events across multiple sources (e.g. a firewall log and a database log) to reveal a multi-stage attack that no single log would show on its own.

8.5.2 Manual vs. Automated Analysis

Aspect	Manual Review	Automated Analysis
Speed	Slow; practical only for small volumes	Near real-time, scales to large volumes
Consistency	Varies with analyst experience	Consistent rule application
Novel threats	Human intuition can spot the unusual	Limited to what rules or models are trained on
Cost	Labour-intensive	Higher upfront tooling cost, lower ongoing effort

Example 8.3 — Correlation in Practice

A firewall log shows repeated connection attempts to an internal server from an external IP over several hours. Independently, the server's authentication log shows a successful login using an administrative account at the exact moment the external scanning stopped.

Analysis

Neither log alone proves compromise — failed external connections could be routine scanning noise, and a successful admin login could be legitimate. Correlated together, however, the timing strongly suggests the external attempts eventually succeeded, warranting immediate investigation of that administrative account.

Note: Effective audit trail analysis depends on the quality of the underlying trail: incomplete or unsynchronized logs will produce false conclusions no matter how sophisticated the analysis technique.

Key Takeaways

- ✓ A security audit is a structured, evidence-based evaluation of controls, following a repeatable cycle from planning through remediation.
- ✓ A security auditing architecture separates event collection, trail storage, and analysis into distinct layers, often unified in a SIEM platform.
- ✓ An audit trail must be complete, tamper-evident, and attributable to be trustworthy as evidence.
- ✓ Implementing the logging function well means logging the right events, in a standard format, with reliable context, and protecting the log path itself.
- ✓ Audit trail analysis combines pattern matching, anomaly detection, statistical analysis, and correlation to turn raw logs into actionable security intelligence.

— *End of Unit 8* —

bcalink.com