A09:2025 — Security Logging & Alerting Failures

January 6, 2026

Document Summary

This document consolidates the provided content for *A09:2025 — Security Logging & Alerting Failures* into a structured, print-ready reference, including background context, scoring metrics, the operational impact of insufficient logging/monitoring/alerting, common failure modes (coverage gaps, tampering risk, lack of monitoring, poor escalation, alert fatigue, and missing playbooks), prevention guidance (context-rich logs, integrity controls, encoding, audit trails, SOC use cases and playbooks, honeytokens, and incident response planning), illustrative real-world scenarios, references, and the mapped CWE list.

Contents

1 Background 2

2 Score Table 2

3 Description 2

4 How to Prevent 3

5 Example Attack Scenarios 4

6 References 4

7 List of Mapped CWEs 6

1 Background

Security Logging & Alerting Failures retains its position at #9. This category includes a slight name change to emphasize the alerting function needed to induce action on relevant logging events.

This category will always be underrepresented in the data and, for the third time, was voted into its position via the community survey. It is difficult to test for and has minimal representation in CVE/CVSS data (only 723 CVEs), but can be highly impactful for visibility, incident alerting, and forensics.

This category includes issues such as:

- CWE-117: Improper output encoding/neutralization for logs
- CWE-532: Inserting sensitive data into log files
- CWE-778: Insufficient logging

2 Score Table

Metric	Value
CWEs Mapped	5
Max Incidence Rate	11.33%
Avg Incidence Rate	3.91%
Max Coverage	85.96%
Avg Coverage	46.48%
Avg Weighted Exploit	7.19
Avg Weighted Impact	2.65
Total Occurrences	260,288
Total CVEs	723

Table 1: Provided scoring summary for Security Logging & Alerting Failures.

3 Description

Without logging and monitoring, attacks and breaches cannot be detected. Without alerting, it is difficult to respond quickly and effectively during a security incident.

Insufficient logging, continuous monitoring, detection, and alerting to initiate active responses occur any time:

- Auditable events (e.g., logins, failed logins, high-value transactions) are not logged or are logged inconsistently (e.g., logging only successful logins but not failures).
- Warnings and errors generate no, inadequate, or unclear log messages.
- Log integrity is not properly protected from tampering.
- Application and API logs are not monitored for suspicious activity.
- Logs are only stored locally and are not properly backed up.

- Alerting thresholds and escalation processes are missing or ineffective; alerts are not received or reviewed in a reasonable amount of time.
- Penetration testing and dynamic application security testing (DAST) scans (e.g., Burp or ZAP) do not trigger alerts.
- The application cannot detect, escalate, or alert for active attacks in real-time or near real-time.
- Logging and alerting events are made visible to users or attackers (see A01:2025 — Broken Access Control), or sensitive information is logged that should not be logged (e.g., PII or PHI).
- Log data is not correctly encoded, creating risk of injections or attacks on logging/monitoring systems.
- Errors and exceptional conditions are missing or mishandled such that the system is unaware an error occurred and therefore cannot log the problem.
- Alert "use cases" are missing or outdated, preventing recognition of special situations.
- Excessive false positives create alert fatigue, making it impossible to distinguish important alerts from unimportant ones.
- Detected alerts cannot be processed correctly because the playbook is incomplete, outdated, or missing.

4 How to Prevent

Developers should implement some or all of the following controls, depending on application risk:

Developer and System Controls

1. Ensure all login, access control, and server-side input validation failures can be logged with sufficient user context to identify suspicious or malicious accounts, and retained long enough to support delayed forensic analysis.
2. Ensure every part of the application containing a security control is logged, whether the control succeeds or fails.
3. Ensure logs are generated in a format that log management solutions can easily consume.
4. Ensure log data is encoded correctly to prevent injections or attacks on logging or monitoring systems.
5. Ensure all transactions have an audit trail with integrity controls to prevent tampering or deletion (e.g., append-only database tables or similar mechanisms).
6. Ensure transactions that throw an error are rolled back and restarted. Always fail closed.
7. If application behavior is suspicious, issue an alert. Provide developers guidance on suspicious patterns, or procure a system that supports this capability.

DevSecOps, SOC, and Operational Controls

1. Establish effective monitoring and alerting use cases including playbooks so suspicious activity is detected and responded to quickly by the Security Operations Center (SOC).
2. Add “honeytokens” as attacker traps (e.g., in databases, data, or as real/technical user identities). Because they are not used in normal operations, any access produces alertable events with low false positives.
3. Optionally apply behavior analysis and AI support to reduce false positives and improve alert quality.
4. Establish or adopt an incident response and recovery plan, such as NIST 800-61r2 (or later). Train developers to recognize application attacks and incidents and to report them appropriately.
5. Consider commercial and open-source application protection and log correlation products (e.g., OWASP ModSecurity Core Rule Set; Elasticsearch/Logstash/Kibana (ELK) stack), as well as commercial observability tools supporting near real-time response and blocking.

5 Example Attack Scenarios

Scenario #1: Undetected Breach Due to Lack of Logging and Monitoring

A children’s health plan provider could not detect a breach due to lack of monitoring and logging. An external party informed the provider that an attacker had accessed and modified thousands of sensitive health records of more than 3.5 million children. A post-incident review found significant vulnerabilities were not addressed. Without logging or monitoring, the breach may have been ongoing since 2013 (over seven years).

Scenario #2: Third-party Cloud Provider Breach Notification Lag

A major Indian airline experienced a data breach involving more than ten years of personal data of millions of passengers, including passport and credit card data. The breach occurred at a third-party cloud hosting provider, who notified the airline only after some time.

Scenario #3: GDPR-reportable Breach from Payment App Vulnerabilities

A major European airline suffered a GDPR-reportable breach reportedly caused by payment application security vulnerabilities exploited by attackers, who harvested more than 400,000 customer payment records. The airline was fined 20 million pounds by the privacy regulator.

6 References

- OWASP Proactive Controls: C9: Implement Logging and Monitoring
- OWASP Application Security Verification Standard: V16 Security Logging and Error Handling
- OWASP Cheat Sheet: Application Logging Vocabulary

- OWASP Cheat Sheet: Logging
- Data Integrity: Recovering from Ransomware and Other Destructive Events
- Data Integrity: Identifying and Protecting Assets Against Ransomware and Other Destructive Events
- Data Integrity: Detecting and Responding to Ransomware and Other Destructive Events
- Real world example of such failures in Snowflake Breach

7 List of Mapped CWEs

CWE	Title
CWE-117	Improper Output Neutralization for Logs
CWE-221	Information Loss of Omission
CWE-223	Omission of Security-relevant Information
CWE-532	Insertion of Sensitive Information into Log File
CWE-778	Insufficient Logging