A08:2025 — Software or Data Integrity Failures

January 6, 2026

Document Summary

This document consolidates the provided content for *A08:2025 — Software or Data Integrity Failures* into a structured, print-ready reference, including background context, scoring metrics, description of integrity and trust-boundary failures (untrusted code/data treated as trusted), prevention guidance (signing, trusted repositories, review and segregation in CI/CD, and deserialization integrity protections), attack 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 3

6 References 4

7 List of Mapped CWEs 5

1 Background

Software or Data Integrity Failures continues at #8, with a slight, clarifying name change from "Software and Data Integrity Failures." This category focuses on failures to maintain trust boundaries and verify the integrity of software, code, and data artifacts at a lower level than Software Supply Chain Failures.

This category emphasizes making assumptions related to software updates and critical data without verifying integrity.

Notable Common Weakness Enumerations (CWEs) include:

- CWE-829: Inclusion of Functionality from Untrusted Control Sphere
- CWE-915: Improperly Controlled Modification of Dynamically-Determined Object Attributes
- CWE-502: Deserialization of Untrusted Data

2 Score Table

Metric	Value
CWEs Mapped	14
Max Incidence Rate	8.98%
Avg Incidence Rate	2.75%
Max Coverage	78.52%
Avg Coverage	45.49%
Avg Weighted Exploit	7.11
Avg Weighted Impact	4.79
Total Occurrences	501,327
Total CVEs	3,331

Table 1: Provided scoring summary for Software or Data Integrity Failures.

3 Description

Software and data integrity failures relate to code and infrastructure that do not protect against invalid or untrusted code or data being treated as trusted and valid.

Examples include:

- Applications that rely upon plugins, libraries, or modules from untrusted sources, repositories, and content delivery networks (CDNs).
- Insecure CI/CD pipelines that do not consume and provide software integrity checks, introducing the potential for unauthorized access, insecure or malicious code, or system compromise.
- CI/CD processes that pull code or artifacts from untrusted locations and/or do not verify them before use (e.g., by checking signatures or similar mechanisms).
- Auto-update functionality where updates are downloaded and applied without sufficient integrity verification; attackers could potentially upload malicious updates to be distributed widely.

- Insecure deserialization, where objects or state are serialized into attacker-visible and modifiable structures and then used without appropriate integrity and trust validation.

4 How to Prevent

Integrity and Trust Controls

1. Use digital signatures or similar mechanisms to verify software or data is from the expected source and has not been altered.
2. Ensure libraries and dependencies (e.g., npm, Maven) are obtained only from trusted repositories. For higher risk profiles, consider hosting an internal known-good repository that is vetted.
3. Ensure there is a review process for code and configuration changes to reduce the likelihood that malicious code or configuration is introduced into the software pipeline.
4. Ensure the CI/CD pipeline has proper segregation, configuration, and access control to preserve integrity of the code flowing through build and deploy processes.
5. Ensure unsigned or unencrypted serialized data is not received from untrusted clients and used without integrity checks or digital signatures to detect tampering or replay.

5 Example Attack Scenarios

Scenario #1: Inclusion of Web Functionality from an Untrusted Source

A company uses an external service provider to provide support functionality. For convenience, it sets a DNS mapping for `myCompany.SupportProvider.com` to `support.myCompany.com`. This causes all cookies set on the `myCompany.com` domain (including authentication cookies) to be sent to the support provider.
Anyone with access to the support provider's infrastructure can steal cookies for users who visit `support.myCompany.com` and perform session hijacking.

Scenario #2: Updates Without Signing

Many home routers, set-top boxes, device firmware, and similar systems do not verify updates via signed firmware. Unsigned firmware is a growing target for attackers. This is a major concern because there may be no remediation path other than fixing a future version and waiting for previous versions to age out.

Scenario #3: Package from an Untrusted Source

A developer cannot find an updated version of a package in the usual trusted package manager, so they download it from a random website. The package is not signed, leaving no mechanism to verify integrity. The package contains malicious code.

Scenario #4: Insecure Deserialization

A React application calls a set of Spring Boot microservices. To enforce immutability, the system serializes user state and passes it back and forth with each request. An attacker notices the `rO0` Java object signature (in base64) and uses a Java Deserialization Scanner to gain remote code execution on the application server.

6 References

- OWASP Cheat Sheet: Software Supply Chain Security
- OWASP Cheat Sheet: Infrastructure as Code
- OWASP Cheat Sheet: Deserialization
- SAFECode Software Integrity Controls
- A "Worst Nightmare" Cyberattack: The Untold Story Of The SolarWinds Hack
- CodeCov Bash Uploader Compromise
- *Securing DevOps* by Julien Vehent
- *Insecure Deserialization* by Tenendo

7 List of Mapped CWEs

CWE	Title
CWE-345	Insufficient Verification of Data Authenticity
CWE-353	Missing Support for Integrity Check
CWE-426	Untrusted Search Path
CWE-427	Uncontrolled Search Path Element
CWE-494	Download of Code Without Integrity Check
CWE-502	Deserialization of Untrusted Data
CWE-506	Embedded Malicious Code
CWE-509	Replicating Malicious Code (Virus or Worm)
CWE-565	Reliance on Cookies without Validation and Integrity Checking
CWE-784	Reliance on Cookies without Validation and Integrity Checking in a Security Decision
CWE-829	Inclusion of Functionality from Untrusted Control Sphere
CWE-830	Inclusion of Web Functionality from an Untrusted Source
CWE-915	Improperly Controlled Modification of Dynamically-Determined Object Attributes
CWE-926	Improper Export of Android Application Components