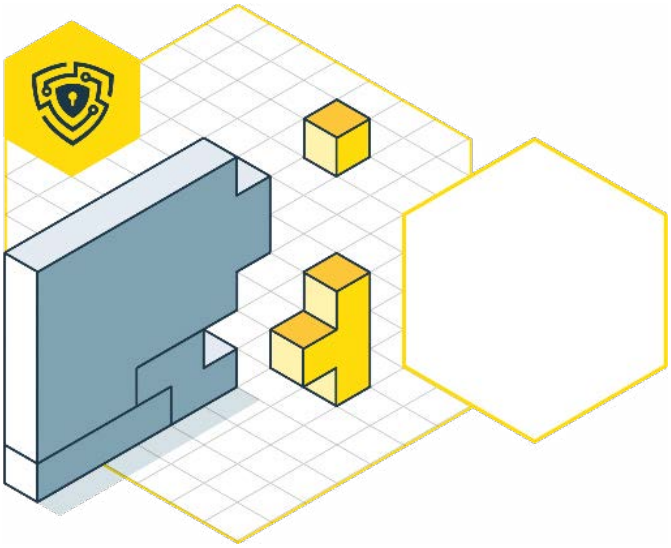
CRASHTEST **SECURITY**

GUIDE FOR PREVENTING SOFTWARE & DATA INTEGRITY FAILURE

WHAT ARE THE STEPS TO KEEP YOUR WEB APP OR API SAFE FROM SUCH VULNERABILITY

GUIDE TO
SOFTWARE & DATA INTEGRITY FAILURE PREVENTION

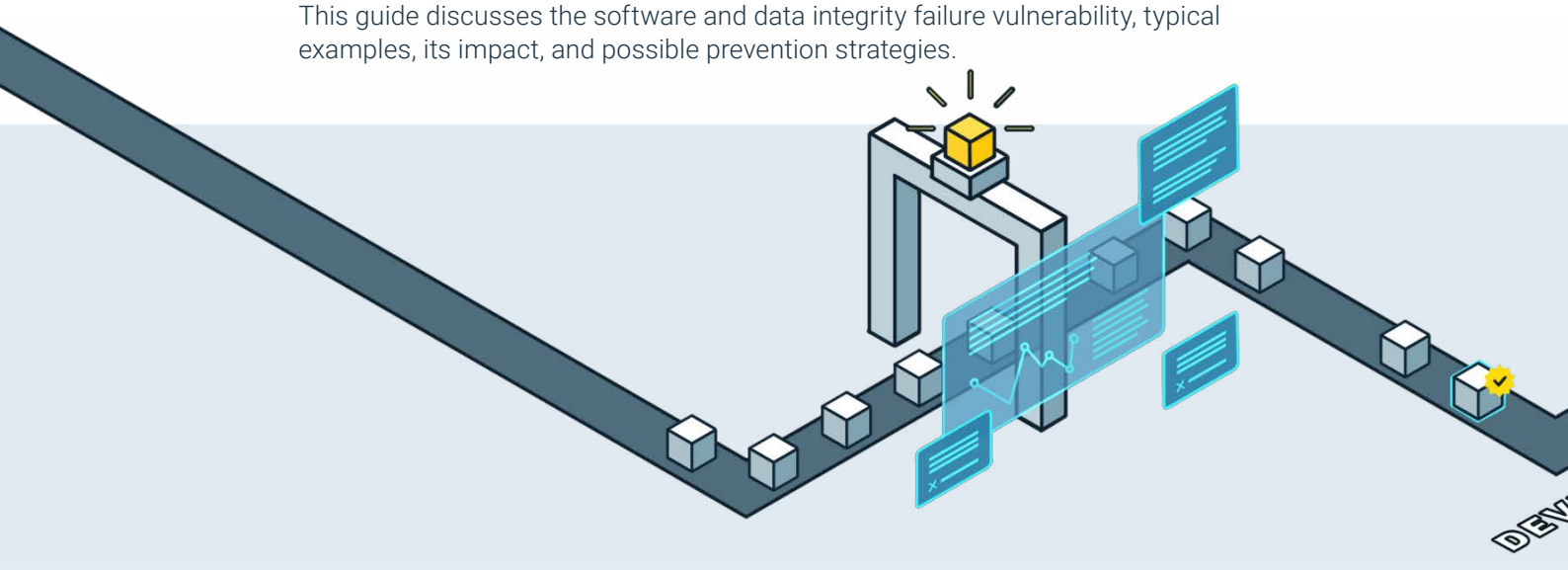
Table of Contents



INTRODUCTION TO THIS GUIDE

Software integrity represents a critical aspect of application security that ensures code and data are protected from unauthorized alterations while keeping them error-free and reliable. Applications with high levels of integrity are often considered accurate, complete, consistent, and secure throughout the deployment and development life cycle. Software and data integrity failures encompass a broad category of application security threats that occur when the application's code and infrastructure are exposed to unauthorized changes that lead to a system-wide compromise.

This guide discusses the software and data integrity failure vulnerability, typical examples, its impact, and possible prevention strategies.



WHAT ARE SOFTWARE AND DATA INTEGRITY FAILURES?

Most modern applications are built using plugins, libraries, and modules from official sources and unknown repositories. A compromise in one repository allows attackers to introduce unauthorized files into the software delivery cycle as an advanced persistent threat. Such models often include vulnerable components in an existing stack without an established review process and extensive integrity checks for software components.

Software and Data Integrity Failures are commonly found in application installations that lack protections against integrity violations. These application security weaknesses typically arise when developers fail to verify the source of objects from untrusted repositories, eventually inheriting platform-level misconfigurations into an existing stack.

The focus on automation also creates a broad attack surface, as modern update mechanisms often download code and execute it without checking for security misconfigurations. Threat actors can exploit such flaws to introduce corrupted payloads into the deployment pipelines, allowing them to run malicious programs or relay unwanted commands to the application server.

Integrity failures are classified into three primary categories:

- **Human error** - These occur when the application's users unknowingly enable integrity violations through abusive use of in-built functionalities
- **Transmission errors** - These result from the alteration of data and application code while in transit
- **Malware and viruses** - malicious code and executable programs that introduce unwanted functionality into the application

TYPES OF SOFTWARE AND DATA INTEGRITY FAILURES

Types of data and software integrity risks include:

DESERIALIZATION OF UNTRUSTED DATA

Deserialization is reconstructing a data structure into its original form from a sequence of byte streams to instantiate the object for consumption. Insecure deserialization occurs when the application deserializes user-supplied objects without adequate validation for the data supplied.

With insecure deserialization, attackers can manipulate serialized objects with the sole intention of passing malicious inputs into the application logic. Attackers can even replace the original object with an object from an entirely different class. Insecure deserialization of untrusted data allows threat actors to reuse the existing source code, enabling the addition of various other exploits, such as remote code execution and object injection attacks.

AUTO-UPDATE FUNCTIONALITY

Most development frameworks ship with the ability to check for new updates and download and install them without human intervention. While the automatic update process improves developer productivity by eliminating manual download and installation procedures, it allows attackers to include malicious updates in the deployment pipeline. Most software update mechanisms lack an update authentication functionality, allowing the inclusion of components from untrusted sites. This opens up the pipeline to exploits such as malicious code injection and a man-in-the-middle attack.

INCLUSION OF FUNCTIONALITY FROM AN UNTRUSTED CONTROL SPHERE

This flaw occurs when the application imports executable software or functionality from an external domain without checking for Subresource Integrity. In such cases, the application downloads functionality it does not have direct control over, such as libraries and web widgets, but only performs insufficient integrity validation.

The entire application remains susceptible to security threats if the included functionality contains security vulnerabilities. The included functionality could consist of outdated components and be spoofed or altered while transmitted from the source. Depending on the injected functionality, consequences of this weakness include unauthorized information disclosure, open redirects to malicious programs, and stealing of user cookies, among others.

RELIANCE ON COOKIES WITHOUT VALIDATION AND INTEGRITY CHECKING

This flaw occurs in applications that rely on cookies for critical security controls but fail to ensure that the cookie data is valid for the associated user. In most modern applications, developers perform sufficient validation against request and URL parameters while assuming that attackers cannot obtain and alter the values of cookies. Attackers can modify cookies inside the browsers or implement client-side code outside the browser. Modifying cookies allows hackers to bypass access control mechanisms for unauthorized access to critical resources and data.

SOFTWARE AND DATA INTEGRITY FAILURES - SEVERITY LEVEL

The software and data integrity vulnerability ranks **number 8** on OWASP's Top 10 - 2021. Attacks targeting security vulnerability have an average **incidence rate of 2.05%**. The incidence rate and average weighted exploits are attributed to a low severity since exploits rarely work without some modification to the underlying source code.

The software integrity security misconfiguration has a **high average weighted impact of 7.94**, as a lack of integrity checking results in the exposure of an entire software supply chain. Impacts of a successful attack include:

- Unauthorized information disclosure
- Command and object injection attacks
- Man-In-The-middle attacks
- Installation and execution of malicious programs
- Compromise of the entire deployment pipeline

The vulnerability has also been mapped to 1152 common vulnerabilities and exposures and 10 common weakness enumerations. These weaknesses include:

- CWE-345: Insufficient Verification of Data Authenticity
- CWE-353: Missing Support for Integrity Check
- CWE-426: Untrusted Search Path
- CWE-494: Download of Code Without Integrity Check
- CWE-502: Deserialization of Untrusted Data
- 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

IDENTIFY SOFTWARE AND DATA INTEGRITY FAILURES WITH CRASHTEST SECURITY

Crashtest Security Suite integrates seamlessly with most modern frameworks for getting started with automated vulnerability scanning and detecting integrity failures in minutes. Through several automated security scanners, Crashtest Security helps identify different categories of software integrity violations. These include:

- **Command injection scanner** - Scans a web application for security flaws that can be exploited to inject arbitrary commands, ensuring the application server only accepts trusted commands.

- **Privilege escalation scanner** - Prevents threat actors from acquiring elevated privileges that allow them to access and modify critical resources and protected data.

- **HTTP header scanner** - Checks whether the host header parameter has been altered while in transit, which forces the application server to include malicious content in responses.

- **SQL injection scanner** - Checks whether the application's databases are protected from threat actors looking to obtain unauthorized data by issuing malicious commands to the database.

SOFTWARE AND DATA INTEGRITY FAILURE PREVENTION TECHNIQUES

Security measures to prevent software and data integrity failures include:

MULTI-FACTOR AUTHENTICATION

Multi-factor authentication (MFA) requires the fulfillment of multiple authentication parameters using different credential categories to verify a user's identity. This helps administer a layered defense mechanism that makes it difficult for threat actors to access a protected resource. In the eventuality of the breach of one of the critical security controls, attackers remain restricted from accessing or modifying the target resource since they still have one or more barriers to breach.

Authentication controls used in MFA typically fall into three categories:

- **Knowledge factors** - This category of authentication factors typically requires the entity to answer a personal security question. Knowledge-based factors include passwords, personal identification numbers, and one-time passwords, among others.

- **Possession factors** - These factors require having something in their possession before they can be authenticated. Possession factors include key fobs, badges, tokens, subscriber identity module (SIM) cards, etc.

- **Inherence factors** - These factors check the user's biological features before logging them into the system. Inherence factors consist of biometric verification methods, such as fingerprint scans, retina scans, voice authentication, digital signature scans, and facial recognition.

SUPPLY CHAIN SECURITY MANAGEMENT PROGRAMS

Modern software delivery pipelines promote the combination of loosely-coupled components that contain open-source software. Software supply chains require implementing security and risk management best practices to help protect the applications from potential security risks. Given the size and scale of modern application deployments, a supply chain security tool can help implement adequate supply chain security. Popular supply chain security management solutions include:

- Crashtest Security Suite
- Snyk
- OWASP Dependency-Check Project
- OWASP CycloneDX
- OpenVAS
- OpenIAM
- OWASP ZAP
- AlienVault OSSIM
- Metasploit

SECURE DESIGN PATTERNS

Integrity failures are often a result of insecure design practices. Secure design patterns help reduce the chances of design flaws in the code and help development teams prepare a mitigation plan in case the vulnerabilities are identified in production. Secure design patterns also include provisions such as the principle of least privilege, strong security protocols, and incident response plans, which help avoid exploits and reduce the impacts of an attack.

BEST PRACTICES IN PREVENTING SOFTWARE AND DATA INTEGRITY FAILURES

Best practices to avoid integrity failures in modern applications include:

USE DIGITAL SIGNATURES TO VERIFY SOFTWARE COMPONENTS

Code signing enables software users to check the integrity of software components using hash functions. This allows developers to verify if specific application components are certified by an electronic signature of a trusted source. These signatures follow public key infrastructure patterns that facilitate smoother component verification, eventually enabling integrity checks without compromising the agility of the delivery pipeline.

ENFORCE CI/CD PIPELINE SEGREGATION AND ACCESS CONTROLCOOKIES

Security professionals should decouple all development and deployment pipeline functions to ensure a successful exploit has minimal effect on the overall application infrastructure. When configuring access rules, it is also recommended to administer robust access roles and leverage the law of least privileges to ensure users can only access the functionalities needed for their tasks and are not able to trick the system into escalating privileges.

ONLY USE COMPONENTS FROM TRUSTED REPOSITORIES

Ensure that the application‘s dependencies and libraries are pulled only from trusted repositories. Where possible, use self-hosted repositories configured with stringent security controls. When using public repositories, only consume resources that have verified digital signatures to ensure they are from trusted sources.

ENFORCE SECURE CODE REVIEWS

a All core components used in the development pipeline should undergo a vigorous, independent, automated application-level review to ensure it is free of any security vulnerability that impacts its integrity. Security professionals should also inspect every line of code for cyber threats, ensuring only those components that meet strict integrity checks are eventually pushed to production.

MAINTAINING SOFTWARE AND DATA INTEGRITY WITH CRASHTEST SECURITY

Crashtest Security Suite is an online SaaS offering that enforces automated black box penetration testing while simulating scenarios of integrity failures to aid threat modeling. The platform integrates with the most significant frameworks seamlessly, which helps spot security defects before threat actors do.

With its quick security assessments, actionable security reports, and low false positives, Crashtest Security enables proactive prevention of attacks targeting integrity failures.

To know more about how Crashtest Security can help maintain your application stack‘s software and data integrity, try a free 14-day trial here.

Start 2-Week Trial for Free

CRASHTEST **SECURITY**

WWW.CRASHTEST-SECURITY.COM