A10:2025 — Mishandling of Exceptional Conditions

January 6, 2026

Document Summary

This document consolidates the provided content for *A10:2025 — Mishandling of Exceptional Conditions* into a structured, print-ready reference, including category background, scoring metrics, a definition of exceptional-condition mishandling and its security implications, prevention guidance emphasizing fail-closed transactional behavior, localized exception handling with global fallback, consistent centralized patterns, rate limiting and resource controls, and integration with logging/monitoring/alerting (A09). It also includes illustrative attack scenarios, references, and the mapped CWE list.

Contents

1 Background

Mishandling of Exceptional Conditions is a new category for 2025. This category contains 24 CWEs and focuses on improper error handling, logical errors, failing open, and other scenarios stemming from abnormal conditions that systems may encounter.

Some CWEs in this category were previously associated with "poor code quality," which is overly broad. This category is intended to provide more specific guidance.

Notable CWEs include:

- CWE-209: Generation of Error Message Containing Sensitive Information
- CWE-234: Failure to Handle Missing Parameter
- CWE-274: Improper Handling of Insufficient Privileges
- CWE-476: NULL Pointer Dereference
- CWE-636: Not Failing Securely ("Failing Open")

2 Score Table

Metric	Value
CWEs Mapped	24
Max Incidence Rate	20.67%
Avg Incidence Rate	2.95%
Max Coverage	100.00%
Avg Coverage	37.95%
Avg Weighted Exploit	7.11
Avg Weighted Impact	3.81
Total Occurrences	769,581
Total CVEs	3,416

Table 1: Provided scoring summary for Mishandling of Exceptional Conditions.

3 Description

Mishandling exceptional conditions in software occurs when programs fail to prevent, detect, and respond to unusual or unpredictable situations, leading to crashes, unexpected behavior, and sometimes vulnerabilities.

This can involve one or more of the following three failings:

1. The application does not prevent an unusual situation from happening.
2. The application does not identify the situation as it is happening.
3. The application responds poorly (or not at all) to the situation afterwards.

Exceptional conditions can be caused by:

- Missing, poor, or incomplete input validation
- Late or overly high-level error handling rather than handling at the functions where errors occur
- Unexpected environmental states (e.g., memory exhaustion, privilege issues, network faults)
- Inconsistent exception handling, or exceptions not handled at all
- System entering an unknown, unpredictable state when it is unsure of the next instruction

Hard-to-find errors and exceptions can threaten the security posture of an application for a long time.

3.1 Security Impact

Many security vulnerabilities can arise from mishandled exceptional conditions, such as:

- Logic bugs and flawed business rules
- Overflows and memory/state/resource issues
- Race conditions and timing issues
- Fraudulent transactions
- Authentication and authorization weaknesses

These can affect the confidentiality, availability, and/or integrity of systems and data. Attackers may manipulate flawed error handling to exploit these weaknesses.

4 How to Prevent

Core Principles

To handle exceptional conditions properly, you must plan for abnormal situations and "expect the worst." Catch errors at the point of occurrence, handle them meaningfully, and ensure recovery. Handling should include user-visible errors (understandable messages), logging of the event, and alerting where justified. Use a global exception handler as a final safety net.

4.1 Catch and Handle at the Source, with a Global Fallback

- Catch every possible system error directly where it occurs and handle it with a meaningful recovery strategy.
- Include consistent user-facing error behavior (avoid leaking sensitive details), logging, and escalation/alerting when warranted.
- Implement a global exception handler as a fallback for unanticipated failures.

4.2 Fail Closed for Transactions

Catching and handling exceptional conditions prevents the underlying infrastructure from dealing with unpredictable states.

- If an error occurs part-way through a transaction, **roll back the entire transaction** and start again (fail closed).

- Avoid "partial recovery" for multi-step transactions, as this commonly creates unrecoverable mistakes.

4.3 Prevent Exceptional Conditions via Limits and Controls

Whenever possible:

- Add rate limiting, resource quotas, throttling, and other limits to prevent exceptional conditions.
- Avoid "limitless" design, which can result in reduced resilience, denial-of-service risk, brute force success, and excessive cloud costs.
- Consider whether identical repeated errors above a threshold should be emitted as aggregated statistics (count and time window), appended to the original message so as not to interfere with automated logging/monitoring (see A09:2025 — Security Logging & Alerting Failures).

4.4 Consistency, Centralization, and Secure Engineering Practices

- Implement strict input validation (plus sanitization/escaping for hazardous characters that must be accepted).
- Centralize error handling, logging, monitoring, and alerting, and apply a consistent global exception strategy.
- Avoid multiple divergent error-handling approaches across the same application; handle exceptional conditions in one place, the same way each time.
- Create project security requirements from the guidance in this category; perform threat modeling and/or secure design reviews during design.
- Perform code review and static analysis, and execute stress/performance/penetration testing of the final system.
- Where possible, standardize exceptional-condition handling across the organization to improve reviewability and auditability.

5 Example Attack Scenarios

Scenario #1: Resource Exhaustion via Poor Cleanup (Denial of Service)

If an application catches exceptions during file uploads but does not release resources after exceptions occur, each new exception can leave resources locked or otherwise unavailable until resources are exhausted.

Scenario #2: Sensitive Data Exposure via Error Messages

Improper error handling (including database errors) may reveal full system errors to the user. An attacker can force repeated errors to collect sensitive system information, using it as reconnaissance to craft stronger SQL injection or other attacks.

Scenario #3: State Corruption in Financial Transactions (Failure to Roll Back)

An attacker interrupts a multi-step transaction via network disruptions. Example transaction order:

1. Debit user account
2. Credit destination account
3. Log transaction

If the system does not properly roll back the entire transaction (fail closed) when an error occurs mid-flow, an attacker could potentially drain funds or exploit race conditions that cause multiple credits.

6 References

- OWASP MASVS–RESILIENCE
- OWASP Cheat Sheet: Logging
- OWASP Cheat Sheet: Error Handling
- OWASP Application Security Verification Standard (ASVS): V16.5 Error Handling
- OWASP Testing Guide: 4.8.1 Testing for Error Handling
- Best practices for exceptions (Microsoft, .NET)
- Clean Code and the Art of Exception Handling (Toptal)
- General error handling rules (Google for Developers)
- Example of real-world mishandling of an exceptional condition

7 List of Mapped CWEs

CWE	Title
CWE-209	Generation of Error Message Containing Sensitive Information
CWE-215	Insertion of Sensitive Information Into Debugging Code
CWE-234	Failure to Handle Missing Parameter
CWE-235	Improper Handling of Extra Parameters
CWE-248	Uncaught Exception
CWE-252	Unchecked Return Value
CWE-274	Improper Handling of Insufficient Privileges
CWE-280	Improper Handling of Insufficient Permissions or Privileges
CWE-369	Divide By Zero
CWE-390	Detection of Error Condition Without Action
CWE-391	Unchecked Error Condition
CWE-394	Unexpected Status Code or Return Value
CWE-396	Declaration of Catch for Generic Exception
CWE-397	Declaration of Throws for Generic Exception
CWE-460	Improper Cleanup on Thrown Exception
CWE-476	NULL Pointer Dereference
CWE-478	Missing Default Case in Multiple Condition Expression
CWE-484	Omitted Break Statement in Switch
CWE-550	Server-generated Error Message Containing Sensitive Information
CWE-636	Not Failing Securely (“Failing Open”)
CWE-703	Improper Check or Handling of Exceptional Conditions
CWE-754	Improper Check for Unusual or Exceptional Conditions
CWE-755	Improper Handling of Exceptional Conditions
CWE-756	Missing Custom Error Page