OWASP Top 10:2025 – A06 to A10

Web Hacking

Fall Term 2026/27

Emmanuel Benoist | BFH-TI

Table of Contents

- Introduction
- A06 – Insecure Design
- A07 – Authentication Failures
- A08 – Software or Data Integrity Failures
- A09 – Security Logging and Alerting Failures
- A10 – Mishandling of Exceptional Conditions
- Conclusion

Introduction

Where we Are in the Top 10:2025

Covered in previous chapters

1. A01 Broken Access Control
2. A02 Security Misconfiguration
3. A03 Software Supply Chain Failures
4. A04 Cryptographic Failures
5. A05 Injection

This chapter

6. A06 Insecure Design
7. A07 Authentication Failures
8. A08 Software or Data Integrity Failures
9. A09 Security Logging and Alerting Failures
10. A10 Mishandling of Exceptional Conditions

- These five complete the picture: **design**, **identity**, **integrity**, **visibility** and **robustness**
- **A10 is brand new** in 2025; A07 and A09 were renamed

The Five at a Glance

Cat.	Name	CWEs	CVEs
A06	Insecure Design	39	7 647
A07	Authentication Failures	36	7 147
A08	Software or Data Integrity Failures	14	3 331
A09	Security Logging & Alerting Failures	5	723
A10	Mishandling of Exceptional Conditions	24	3 416

- **A06** and **A10** are about *whole classes* of flaws, hence many CWEs
- **A09** has few CVEs – it is rarely a “vulnerability” per se, but it decides whether you see the others
- Together they shift focus from single bugs to **how software is designed, operated and observed**

A06 – Insecure Design

A06:2025 – Insecure Design

- **Flaws in the *design*, not the code**
  - A missing or ineffective security control that was never designed in
- **Key distinction**
  - “A secure design can still have implementation defects”
  - “An **insecure design cannot be fixed by a perfect implementation**” – the control simply isn’t there
- Dropped from #4 (2021) to #6

CWEs mapped	39
Avg incidence	1.86%
Max coverage	88.76%
Occurrences	729 882
Total CVEs	7 647

- **Causes**: no business-risk profiling, no threat modeling, missing secure-design patterns, business logic that ignores abuse cases

A06 – Examples and Prevention

- **Example: business-logic abuse (OWASP)**
  - A cinema lets you reserve many seats across locations without a deposit; an attacker locks thousands of seats ⇒ revenue loss
  - No code bug – the *workflow* was never designed against abuse
- **Example: weak recovery by design**
  - "Security questions" as proof of identity (prohibited by NIST 800-63b): answers are guessable or shared – the mechanism is wrong by design
- **Prevention: shift security *left***
  - A secure development lifecycle with AppSec involvement
  - **Threat modeling** for auth, access control and business logic
  - Security in user stories; abuse/misuse cases; tests against the threat model; segregate tiers by exposure

A07 – Authentication Failures

A07:2025 – Authentication Failures

- **Confirming identity goes wrong**
  - Weak/default passwords, no MFA, brute-force & credential stuffing, broken session management
- **Stable at #7**; renamed (was “Identification and Authentication Failures”)
- Closely tied to A01 (authZ) and A04 (password hashing)

CWEs mapped	36
Avg incidence	2.92%
Max incidence	15.80%
Occurrences	1 120 673
Total CVEs	7 147

- **Symptoms**: hard-coded passwords, session fixation, identifiers reused or not invalidated at logout, sessions that never time out

A07 – Examples and Prevention

- **Example: hybrid credential stuffing**
  - Attackers take breached passwords and tweak them (`Winter2025` → `Winter2026`)
  - With no automated defenses, the app becomes a **password oracle** that confirms which guesses work
- **Example: session not invalidated**
  - A user on a shared PC closes the browser without logging out; the next person resumes the authenticated session
- **Prevention**
  - **Multi-factor authentication**; no default admin credentials
  - Check new passwords against **breach lists** (haveibeenpwned)
  - Rate limiting / lockout / bot defenses against automated attacks
  - Server-side session manager with high-entropy IDs; invalidate on logout; sensible timeouts; follow NIST 800-63b (no forced rotation)

A08 – Software or Data Integrity Failures

A08:2025 – Software or Data Integrity Failures

- **Trusting code or data without verifying integrity**
  - Updates, plugins, serialized objects accepted without a signature check
- **Relation to A03**
  - A03 is the *whole supply chain*; A08 is the **verification step** – did we check *this* artifact or data before trusting it?

CWEs mapped	14
Avg incidence	2.75%
Max coverage	78.52%
Avg weighted exploit	7.11
Occurrences	501 327
Total CVEs	3 331

- **Hotspots**: insecure deserialization, auto-update without signature verification, CI/CD without integrity checks, unsigned firmware

A08 – Examples and Prevention

- **Example: unsigned updates (OWASP)**
  - Home routers / IoT accept firmware without verifying a signature
  - An attacker ships **malicious firmware** – no way to remediate
- **Example: insecure deserialization**

```
# Dangerous: rebuilding objects from attacker-controlled bytes
obj = pickle.loads(request.body)        # can execute code during unpickling
```

- A tampered serialized object can alter logic or trigger code execution
- **Prevention**
  - **Digital signatures** for updates, artifacts and critical data
  - Trusted repositories only; code review; CI/CD segregation & access control
  - Avoid native deserialization of untrusted data; use plain data formats (JSON) with strict schemas; verify integrity before use

A09 – Security Logging and Alerting Failures

A09:2025 – Security Logging and Alerting Failures

- **If you cannot see it, you cannot stop it**
  - Missing logs, no monitoring, no alerting, no response
- Renamed to stress **alerting**, not just logging
- **Few CVEs** – it is rarely the entry point, but it determines **detection and forensics**

CWEs mapped	5
Avg incidence	3.91%
Max coverage	85.96%
Occurrences	260 288
Total CVEs	723

- **Symptoms**: logins/failures not logged, logs without integrity, no alert thresholds, logs holding sensitive data or vulnerable to log injection

A09 – Examples and Prevention

- **Example: breaches undetected for years (OWASP)**
  - A health-plan provider: 3.5 M records exposed, unnoticed for **seven years** for lack of monitoring and logging
  - An airline: ten years of passenger data breached at a cloud provider, with no timely detection
- **Prevention**
  - Log auth, access-control and validation **failures and successes** with enough context – and no sensitive data
  - **Encode log data** to prevent log injection (ties to A05)
  - **Append-only, integrity-protected** audit trails
  - Centralise in a format your SIEM can consume; define **alerting use cases**, thresholds and **incident-response playbooks**
  - Detect the attacks from earlier chapters: enumeration, stuffing, injection probes

A10 – Mishandling of Exceptional Conditions

A10:2025 – Mishandling of Exceptional Conditions

- **Brand-new category in 2025**
- **What happens when things go wrong?**
  - Poor error handling, logical errors, and above all **failing open**
- Affects **confidentiality, integrity *and* availability**

CWEs mapped	24
Avg incidence	2.95%
Max incidence	20.67%
Occurrences	769 581
Total CVEs	3 416

- **Root causes**: missing input validation, incomplete exception handling, unexpected environmental states, NULL dereference, over-informative error messages

A10 – Fail Closed, not Open

- **The core idea: on error, deny**

```
# Failing OPEN: an exception in the check grants access
try:
    allowed = authz.check(user, resource)
except Exception:
    allowed = True            # <-- disaster: errors become access

# Failing CLOSED: any error denies and is logged
try:
    allowed = authz.check(user, resource)
except Exception:
    log.error("authz check failed"); allowed = False
```

- **Examples**: a caught upload exception that never frees resources ⇒ **DoS**; an interrupted multi-step payment without rollback ⇒ **double transfer / drained account**
- **Prevention**: handle exceptions where they occur with real recovery; **centralised error handling**; **always fail closed** (full rollback, no partial state); rate limits & resource quotas; threat modeling, reviews, pen testing

Conclusion

Conclusion: the Remaining Five

- **A06 Insecure Design** – build the right controls *before* coding; threat-model; some flaws cannot be patched away
- **A07 Authentication Failures** – MFA, breach-checked passwords, bot defenses, solid session management
- **A08 Integrity Failures** – verify signatures on code, updates and data; beware insecure deserialization
- **A09 Logging & Alerting Failures** – log, protect, monitor, alert and *respond* – so you can see the other nine
- **A10 Exceptional Conditions** – handle errors deliberately and **fail closed**

Conclusion: the Whole Top 10 Together

- **The categories reinforce each other**
  - A06 designs the controls; A01/A07 enforce them; A04/A08 protect data and code; A02/A03 keep the platform and chain sound; A05 guards the interpreters; A09 watches; A10 keeps it safe under stress
- **Recurring principles**
  - Deny / fail **closed** by default; least privilege everywhere
  - **Never trust input** – or the client, the component, the update
  - Keep **data separate from code**; verify **integrity** and **identity**
  - Make the **secure path the default**, and **automate** the checks
  - **Assume breach**: log, detect, limit the blast radius, respond
- **Security is a property of the whole lifecycle**, not a feature

References

- OWASP Top 10:2025 overview `https://top10.owasp.org/2025/`
- A06 Insecure Design
  `https://top10.owasp.org/2025/A06_2025-Insecure_Design/`
- A07 Authentication Failures
  `https://top10.owasp.org/2025/A07_2025-Authentication_Failures/`
- A08 Software or Data Integrity Failures
  `https://top10.owasp.org/2025/A08_2025-Software_or_Data_Integrity_Failures/`
- A09 Security Logging and Alerting Failures
  `https://top10.owasp.org/2025/A09_2025-Security_Logging_and_Alerting_Failures/`
- A10 Mishandling of Exceptional Conditions
  `https://top10.owasp.org/2025/A10_2025-Mishandling_of_Exceptional_Conditions/`
- OWASP Cheat Sheet Series, SAMM, ASVS; NIST SP 800-63b