TOP 10

Agenda

- What is OWASP Top 10 Project?
- What have been changed since first release in 2023?
- What have been changed since last reales this November?
- Brief overview of all current findings

What is OWASP

- A global nonprofit focused on improving software security.
- Founded in 2001 by security practitioners.
- Known for open, free security resources: Top 10, ASVS, Cheat Sheets, ZAP, etc.
- Community-driven: thousands of volunteers, local chapters, and projects.
- Goal: make security transparent, practical, and built into how software is made.

What is OWASP Top 10

- A community-driven list of the **most critical web application security risks**.
- Updated every few years based on real-world data and industry feedback.
- Helps organizations **prioritize what actually gets attacked**, not theoretical flaws.
- Used globally as a **baseline standard** for secure development and assessments.
- Not a compliance checklist, but a **minimum bar** for application security.

Comparison 2003 until 2013

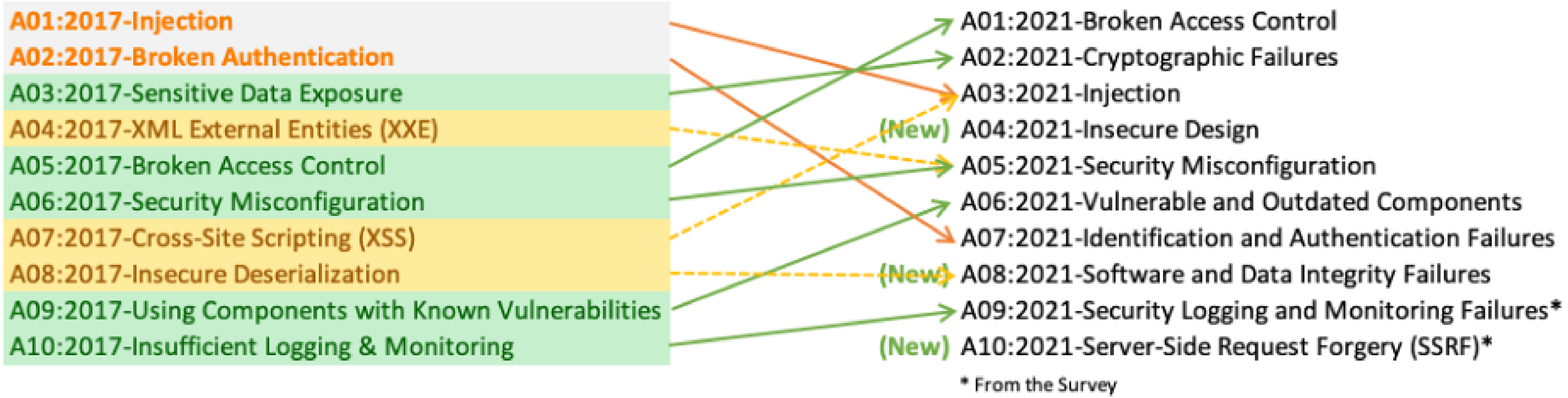
OWASP Top Ten Entries (Unordered)	Releases				
	2003	2004	2007	2010	2013
Unvalidated Input	A1	A1[9]	×	×	×
Buffer Overflows	A5	A5	×	×	×
Denial of Service	×	A9[2]	×	×	×
Injection	A6	A6[3]	A2	A1[10]	A1
Cross Site Scripting (XSS)	A4	A4	A1	A2	A3
Broken Authentication and Session Management	A3	A3	A7	A3	A2
Insecure Direct Object Reference	×	A2	A4[11]	A4	A4
Cross Site Request Forgery (CSRF)	×	×	A5	A5	A8
Security Misconfiguration	A10	A10[3][5]	×	A6	A5
Missing Functional Level Access Control	A2	A2[1]	A10[13]	A8	A7[16]
Unvalidated Redirects and Forwards	×	×	×	A10	A10
Information Leakage and Improper Error Handling	A7	A7[14][4]	A6	A6[8]	×
Malicious File Execution	×	×	A3	A6[8]	×
Sensitive Data Exposure	A8	A8[6][5]	A8	A7	A6[17]
Insecure Communications	×	A10	A9[7]	A9	×
Remote Administration Flaws	A9	×	×	×	×
Using Known Vulnerable Components	×	×	×	×	A9 [18][19]

2013 vs 2017

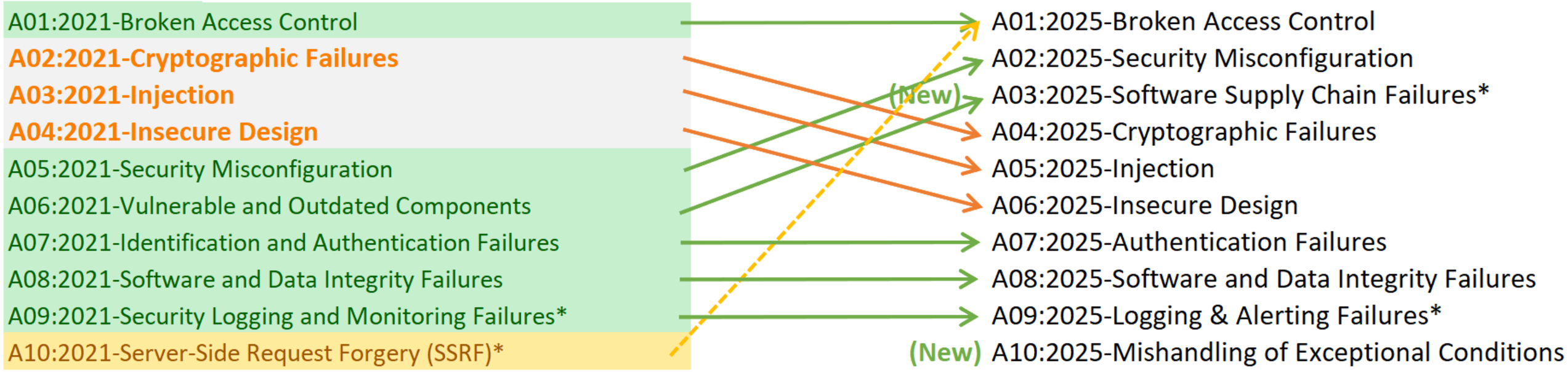
2013	2017
A1 – Injection	A1 – Injection
A2 – Broken Authentication and Session Management	A2 – Broken Authentication
A3 – Cross-Site Scripting (XSS)	A3 – Sensitive Data Exposure
A4 – Insecure Direct Object References **[Merged + A7]**	A4 – XML External Entities (XXE) **[NEW]**
A5 – Security Misconfiguration	A5 – Broken Access Control **[MERGED]**
A6 – Sensitive Data Exposure	A6 – Security Misconfiguration
A7 – Missing Function Level Access Control **[Merged + A4]**	A7 – Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	A8 – Insecure Deserialization **[NEW, COMMUNITY]**
A9 – Using Components with Known Vulnerabilities	A9 – Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	A10 – Insufficient Logging & Monitoring **[NEW, COMMUNITY]**

Source: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf

2017 vs 2021


A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring
A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*
* From the Survey

2021 vs 2025



* From the Survey

* From the Survey

A01:2025 - Broken Access Control

- Violation of the principle of least privilege.
- Bypassing access control checks by modifying the URL.
- Permitting viewing or editing someone else's account.
- Accessing API with missing access controls for POST, PUT and DELETE.
- Elevation of privilege. Acting as a user without being logged in or acting as an admin when logged in as a user.
- Metadata manipulation, such as replaying or tampering with a JSON Web Token (JWT).

A01:2025 - Prevention

- Except for public resources, deny by default.
- Implement access control mechanisms once and re-use them.
- Model access controls should enforce record ownership.
- Unique application business limit requirements should be enforced by domain models.
- Disable web server directory listing.
- Log access control failures, alert admins.
- Rate limit API and controller access to minimize the harm.

A02:2025 - Security Misconfiguration

- Missing appropriate security hardening.
- Unnecessary features are enabled or installed.
- Default accounts and their passwords are still used.
- Error handling reveals stack traces.
- For upgraded systems, the latest security features are disabled.
- The security settings in the servers and application frameworks are not set to secure values.
- The server does not send security headers or directives.
- The software is out of date or vulnerable

A02:2025 - Prevention

- A repeatable hardening process makes it fast and easy to deploy another environment that is appropriately locked down.
- Remove or do not install unused features and frameworks.
- A task to review and update the configurations appropriate to all security notes, updates, and patches.
- A segmented application architecture provides effective and secure separation between components or tenants.
- Sending security directives to clients, e.g., Security Headers.
- An automated process to verify the effectiveness of the configurations and settings in all environments.

A03:2025 - Software Supply Chain Failures

- If you do not know the versions of all components you use.
- If the software is vulnerable, unsupported, or out of date.
- If you do not scan for vulnerabilities regularly.
- If you do not fix or upgrade the underlying platform, frameworks, and dependencies in a risk-based, timely fashion.
- If software developers do not test the compatibility of updated, upgraded, or patched libraries.
- If you do not secure the components configurations.

A03:2025 - Prevention

- Remove unused dependencies, unnecessary features, components, files, and documentation.
- Continuously inventory the versions of both client-side and server-side components (e.g., frameworks, libraries) and their dependencies using tools like versions,
- Only obtain components from official sources over secure links.
- Monitor for libraries and components that are unmaintained or do not create security patches for older versions.

A04:2025 - Cryptographic Failures

- Is any data transmitted in clear text?
- Are any old or weak cryptographic algorithms or protocols used either by default or in older code?
- Are default crypto keys in use?
- Is encryption not enforced, e.g., are any HTTP headers (browser) security directives or headers missing?
- Is the received server certificate and the trust chain properly validated?
- Are deprecated hash functions such as MD5 or SHA1 in use, or are non-cryptographic hash functions used when cryptographic hash functions are needed?
- Is randomness used for cryptographic purposes that was not designed to meet cryptographic requirements?

A04:2025 - Prevention

- Classify data processed, stored, or transmitted by an application. Identify which data is sensitive according to privacy laws, regulatory requirements, or business needs.
- Don't store sensitive data unnecessarily.
- Make sure to encrypt all sensitive data at rest & transit.
- Disable caching for response that contain sensitive data.
- Do not use legacy protocols such as FTP and SMTP for transporting sensitive data.
- Store passwords using strong adaptive and salted hashing functions with a work factor (delay factor), such as Argon2, scrypt, bcrypt or PBKDF2.
- Avoid deprecated cryptographic functions and padding schemes, such as MD5, SHA1, PKCS number 1 v1.5 .

A05:2025 - Injection

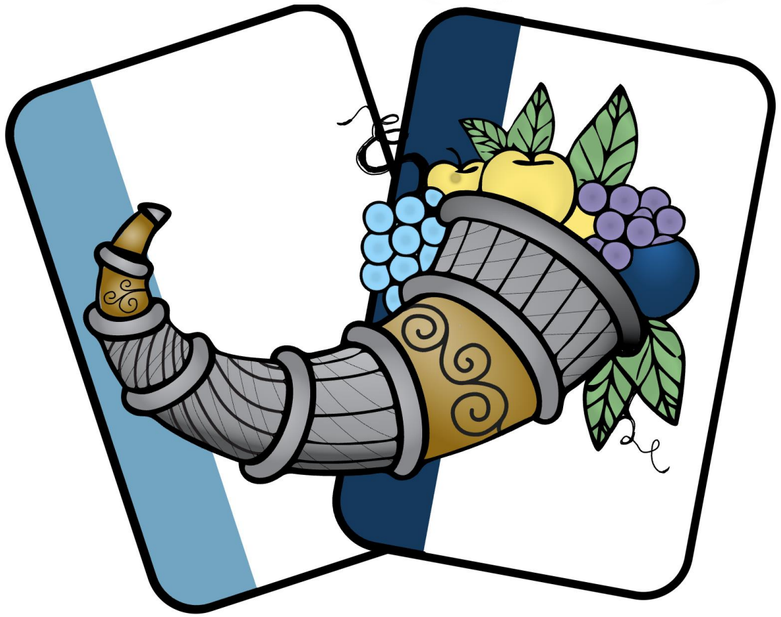
- User-supplied data is not validated, filtered, or sanitized by the application.
- Dynamic queries or non-parameterized calls without context-aware escaping are used directly in the interpreter.
- Hostile data is used within object-relational mapping (ORM) search parameters to extract additional, sensitive records.
- Hostile data is directly used or concatenated. The SQL or command contains the structure and malicious data in dynamic queries, commands, or stored procedures.

A05:2025 - Prevention

- User-supplied data is not validated, filtered, or sanitized by the application.
- Dynamic queries or non-parameterized calls without context-aware escaping are used directly in the interpreter.
- Hostile data is used within object-relational mapping (ORM) search parameters to extract additional, sensitive records.
- Hostile data is directly used or concatenated. The SQL or command contains the structure and malicious data in dynamic queries, commands, or stored procedures.

A06:2025 – Insecure Design

- **Missing or ineffective control design.**
- **Requirements and Resource Management**
- **Secure Design**
- **Secure Development Lifecycle**
- **Threat Modelling – OWASP Cornucopia**



A06:2025 – Prevention

- Establish and use a secure development lifecycle with AppSec .
- Establish and use a library of secure design patterns.
- Use threat modeling for critical authentication, access control, business logic, and key flows.
- Integrate security language and controls into user stories
- Integrate plausibility checks at each tier of your application
- Write unit and integration tests to validate that all critical flows.
- Segregate tier layers on the system and network layers
- Segregate tenants robustly by design throughout all tiers
- Limit resource consumption by user or service

A07:2025 – Authentication Failures

- Permits automated attacks such as credential stuffing, where the attacker has a list of valid usernames and passwords.
- Permits brute force or other automated attacks.
- Permits default, weak, or well-known passwords, such as "Password1" or "admin/admin".
- Uses weak or ineffective credential recovery.
- Uses plain text, encrypted, or weakly hashed passwords data stores.
- Has missing or ineffective multi-factor authentication.
- Exposes session identifier in the URL.
- Reuse session identifier after successful login.
- Does not correctly invalidate Session IDs.

A07:2025 – Prevention

- Where possible, implement multi-factor authentication to prevent automated credential stuffing, brute force, and stolen credential reuse attacks.
- Do not ship or deploy with any default credentials.
- Implement weak password checks, such as testing new or changed passwords against the top 10,000 worst passwords list.
- Align password length, complexity, and rotation policies with National Institute of Standards and Technology (NIST) 800-63b.
- Ensure registration, credential recovery.
- Limit or increasingly delay failed login attempts, but be careful not to create a denial of service scenario.
- Use a server-side, secure, built-in session manager that generates a new random session ID with high entropy after login.

A08:2025 – Integrity Failures

- Code or infrastructure that does not protect against integrity violations.
- Application relies on plugins, libraries and modules from untrusted sources (i.e. public Docker Hub, public Github, CDNs)
- Insecure CI/CD pipeline can introduce the potential for unauthorized access and malicious code.
- Applications might include auto-update or other functionality, such as priority of the library from a package repository that could be overridden.

A08:2025 – Integrity Failures

- Use digital signatures or similar mechanisms to verify the software or data is from the expected source and has not been altered.
- Ensure libraries and dependencies, such as npm or Maven, are consuming trusted repositories. If you have a higher risk profile, consider hosting an internal known-good repository that's vetted.
- Ensure that a software supply chain security tool, is used to verify that components do not contain known vulnerabilities
- Ensure that there is a review process for code and configuration changes to minimize the chance that malicious code or configuration could be introduced into your software pipeline.

A09:2025 – Logging & Alerting Failures

- Auditable events, such as logins, failed logins, and high-value transactions, are not logged.
- Warnings and errors generate no, inadequate, or unclear log messages.
- Logs of applications and APIs are not monitored for suspicious activity.
- Logs are only stored locally.
- Appropriate alerting thresholds and response escalation processes are not in place or effective.
- Penetration testing and scans by dynamic application secur t testing (DAST) tools (such as OWASP ZAP) do not trigger ale
- The application cannot detect, escalate, or alert for active at in real-time or near real-time.

A09:2025 – Prevention

- Log important security events consistently.
- Monitor logs for suspicious behavior.
- Store logs centrally and securely.
- Set alerts that actually trigger and escalate.
- Detect and respond to attacks in real time.

A10:2025 – Mishandling of Exceptions

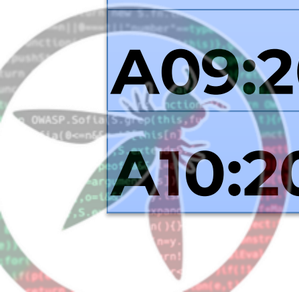
- **Happens when software doesn’t properly prevent, detect, or recover from weird conditions.**
- **Leads to unpredictable behavior: crashes, logic bugs, race conditions, broken auth, data loss.**
- **Often caused by poor input validation, missing error handling, or chaotic/unhandled exceptions.**
- **Attackers exploit messy error handling to bypass checks or manipulate system state.**

A10:2025 – Prevention

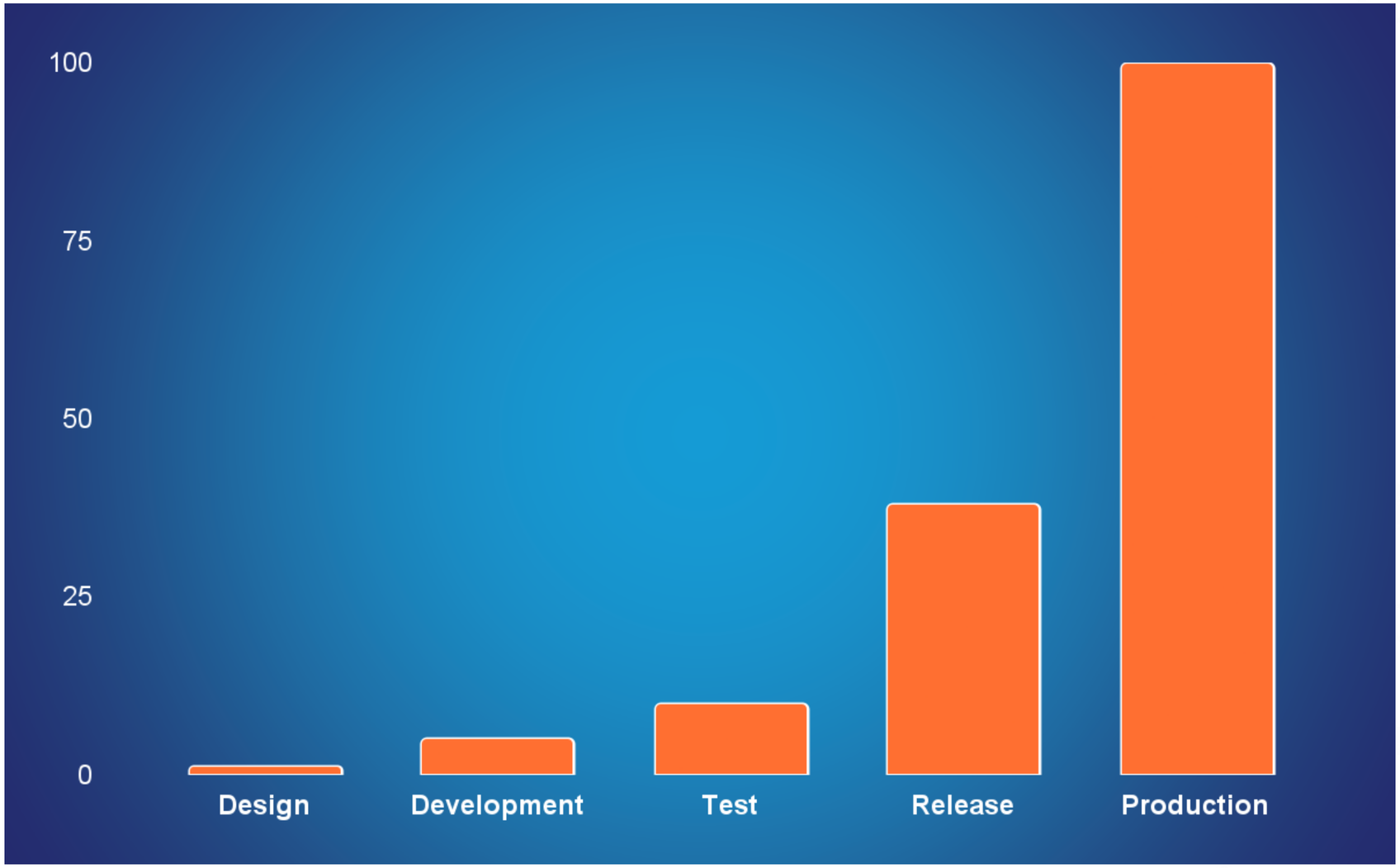
- Catch errors early and fail safely.
- Centralize and standardize exception handling.
- Validate and sanitize all inputs.
- Apply limits on resources and requests.
- Log, monitor, and alert on anomalies.
- Test failure scenarios and secure the design.

Summary

OWASP Top 10 2025	Dev	Ops
A01:2025 - Broken Access Control	X	
A02:2025 - Security Misconfiguration		X
A03:2025 - Software Supply Chain Failures	X	
A04:2025 - Cryptographic Failures	X	
A05:2025 - Injection	X	
A06:2025 - Insecure Design	X	
A07:2025 - Authentication Failures	X	
A08:2025 - Integrity Failures	X	
A09:2025 - Logging & Alerting Failures		X
A10:2025 - Mishandling of Exceptions		X



Cost of fixing a security issue



Summary

- Most risks start in **design and development**. If security isn’t baked in, it breaks later.
- A few show up **only in production**. Weak monitoring and error handling let attacks run unnoticed.
- **Human mistakes** drive many of the issues: misconfigurations, bad access rules, weak auth.
- Dev and Ops **share the blame**. Several problems only disappear when both cooperate.
- Better **design, validation, and visibility** cuts down most of the list.

Thank you!

Онлайн събитие

OWASP Top 10 2025: Какви са най-актуалните рискове в киберсигурността?

Организатор
DEV.BG

Job Board за
IT общността

Дата
14.01.26

Език
Български

Регистрирай се