OWASP Top 10:2025

The Most Critical Web Application Security Risks

Danny Vacar

InfoSec Hamilton, Jan 2026

http://localhost/whoami

- application security engineer
  - security consultant, network and system administrator
  - SANS/GIAC GPEN, GWAPT | Check Point CCSA, CCSE
- private pilot, aviation enthusiast, amateur volleyball player

Mastodon: @FlyingDan [on https://infosec.exchange/@FlyingDan]
GitHub: https://github.com/FlyingDan/presentations [later tonight]

Opinions and thoughts represent me and not any of my employers (past, current, future, parallel universe, etc.)

Agenda

- What is OWASP?
- Notable OWASP Projects
- History of the OWASP Top 10
- What's New in 2025
- The 2025 Top 10 Deep Dive
- Key Takeaways & Practical Actions

What is OWASP?

Open Worldwide Application Security Project

- Founded in 2001 as Open **Web** Application Security Project
- World's largest non-profit focused on software security
- Global community-driven organization
- 250+ local chapters worldwide (Ontario chapters in Toronto, Waterloo, and Ottawa)

Mission: Make software security visible so individuals and organizations can make informed decisions about software security risks

Notable OWASP Projects

Flagship Projects - Strategic value to application security

- **OWASP Top 10** - Reference standard for critical web app risks
  - OWASP Mobile Top 10
  - OWASP Top 10 for Large Language Model Applications
- **ZAP (Zed Attack Proxy)** - Penetration testing tool (free, open-source alternative to PortSwigger's Burp)
- **Dependency-Check** - Software Composition Analysis (SCA) tool
- **Juice Shop** - Intentionally insecure app for training
- **ModSecurity Core Rule Set** - WAF rules

More OWASP Projects

Application Security Verification Standard (ASVS)
Security verification requirements for applications

Web Security Testing Guide (WSTG)
Premier testing resource for web application security

SAMM (Software Assurance Maturity Model)
Framework to improve software security posture

Mobile Application Security (MAS)
Security standards and testing guides for mobile apps

The OWASP Top 10: History

First Published: 2003

Update Cycle: Every 3-4 years

- 2004, 2007, 2010, 2013, 2017, 2021, **2025**

Data-Driven Approach:

- Analyzes real vulnerability data from organizations worldwide
- Over 2.8 million applications analyzed for 2025 edition
- Community survey captures emerging threats not yet in data

Why the Top 10 Matters

Industry Standard

Referenced by MITRE, PCI DSS, DISA-STIG, FTC

First Step Toward Security

Most effective starting point for secure coding culture

Training Foundation

Baseline for security awareness programs

Compliance & Procurement

Used in vendor questionnaires and security audits

Top 10 Methodology

Data Analysis

- 589 Common Weakness Enumerations (CWEs) analyzed
- 175,000+ CVE records mapped

Community Survey

- Captures emerging risks not yet detectable at scale
- Front-line practitioner insights

What's New in 2025?

Two New Categories:

- A03: Software Supply Chain Failures (expanded scope)
- A10: Mishandling of Exceptional Conditions (new)

One Consolidation:

- Server-Side Request Forgery (SSRF) merged into A01

Philosophical Shift:
Focus on **root causes** over symptoms

OWASP Top 10:2025 List

1. **Broken Access Control**
2. **Security Misconfiguration** ⬆️
3. **Software Supply Chain Failures** 🆕
4. **Cryptographic Failures** ⬇️
5. **Injection** ⬇️
6. **Insecure Design** ⬇️
7. **Authentication Failures**
8. **Software or Data Integrity Failures**
9. **Security Logging & Alerting Failures**
10. **Mishandling of Exceptional Conditions** 🆕

A01: Broken Access Control

The Risk: Staying at #1 for a reason

- 40 CWEs mapped (most of any category)
- Now includes SSRF

What Goes Wrong:
Users accessing data or functions they shouldn't be able to

A01: Real-World Examples

Horizontal Privilege Escalation

Change `/profile/677` to `/profile/678` - see another user's data

Vertical Privilege Escalation

Regular user performs admin functions

IDOR (Insecure Direct Object Reference)

Modify IDs in URLs or parameters to access others' resources

Missing Function-Level Access Control

Hidden admin panel accessible to regular users

LLM Agents having super-user access

End-user session is restricted properly but agents' access is not

A01: Practical Takeaways

✅ **Default Deny** - Deny by default, permit explicitly
✅ **Enforce Server-Side** - Never trust client-side controls
✅ **Single Reusable Mechanism** - Centralized access control
✅ **Log Failures** - Monitor and alert on repeated failures
✅ **Rate Limit APIs** - Prevent automated attacks

Key Concept: Authorization is hard - every request needs verification

A02: Security Misconfiguration

The Risk:

- Every tested application had some misconfiguration
- 16 CWEs, including XXE and improper configuration

Why It's Rising:

Modern applications have complex, extensive configuration options

A02: Common Misconfigurations

Default Accounts & Passwords

"admin/admin" still in production (`hunter2`)

Unnecessary Features Enabled

Unused services, pages, accounts

Verbose Error Messages

Stack traces revealing system details

Missing Security Headers

No HSTS, CSP, X-Frame-Options

Outdated Software

Unpatched systems and components

A02: Practical Takeaways

✅ **Hardening Process** - Repeatable process for all environments
✅ **Minimal Platform** - Remove unused features and frameworks
✅ **Security Headers** - Implement comprehensive header policies
✅ **Automated Verification** - Regular configuration scans
✅ **Segmented Architecture** - Separate components with security controls

Key Concept: Secure by default, not secure by configuration

A03: Software Supply Chain Failures

The Risk:

- Highest exploit and impact scores
- Includes malicious packages and compromised builds

Modern Reality:

Your application includes thousands of dependencies you didn't write

A03: Attack Vectors

Malicious Packages

Typosquatting, compromised maintainers

Dependency Confusion

Internal package names matching public repositories

Compromised Build Pipeline

Injecting malware during CI/CD

Transitive Dependencies

Vulnerability several layers deep

Outdated Components

Known CVEs in old library versions

A03: Practical Takeaways

✅ **Inventory Everything** - Software Bill of Materials (SBOM)
✅ **Source Verification** - Only use official repositories
✅ **Automated Scanning** - Continuous dependency monitoring
✅ **Patch Management** - Regular updates with testing
✅ **Vendor Security Assessment** - Evaluate third-party security

Key Concept: The supply chain is your new perimeter

A04: Cryptographic Failures

The Risk:

- 32 CWEs mapped
- Often leads to sensitive data exposure

What Fails:

Protecting data in transit and at rest

A04: Common Failures

Plaintext Storage

Passwords, credit cards, health records unencrypted

Weak Algorithms

MD5, SHA1, deprecated TLS versions

Weak Keys

Predictable encryption keys, insufficient key length

Improper Certificate Validation

Accepting self-signed certificates in production

Hardcoded Keys

Encryption keys in source code

A04: Practical Takeaways

✅ **Classify Data** - Know what needs protection
✅ **Encrypt Everything Sensitive** - Data at rest and in transit
✅ **Strong Algorithms** - AES-256, SHA-256, TLS 1.3+
✅ **Proper Key Management** - Use key management systems
✅ **Salt and Hash Passwords** - Use bcrypt, Argon2, or PBKDF2

Key Concept: Don't invent crypto - use proven standards

A05: Injection

The Risk:

- Still one of the most dangerous
- 38 CWEs mapped
- Ranges from XSS (high frequency) to SQLi (high impact)

What It Is:
Untrusted data interpreted as commands or code

A05: Injection Types

SQL Injection

```
SELECT * FROM users WHERE id = '1' OR '1'='1'
```

Cross-Site Scripting (XSS)

```
<script>steal_cookies()</script>
```

Command Injection

User input executed as system commands

LDAP, XML, NoSQL Injection

Similar concepts, different contexts

A05: Practical Takeaways

✅ **Parameterized Queries** - Never concatenate user input into queries
✅ **Input Validation** - Allowlist validation on all user input
✅ **Output Encoding** - Context-appropriate encoding
✅ **ORM/Framework Protection** - Use built-in protections
✅ **Least Privilege** - Database accounts with minimal permissions

Key Concept: Never trust user input - validate, sanitize, encode

⚠️ ***Input*** is anything your stack processes: headers, cookies, local storage, URL parameters, logs, and so on ⚠️

A06: Insecure Design

The Risk:

- Focus on design and architectural flaws
- Cannot be fixed by perfect implementation
- Requires threat modeling and secure design patterns

The Difference:

Insecure design vs insecure implementation

A06: Design Flaws

Missing Security Controls

No rate limiting on password reset

Inadequate Threat Modeling

Not considering attack scenarios during design

Business Logic Vulnerabilities

Discount codes applied multiple times

Insecure Architecture

Single-tier application with no defense in depth

Insufficient Resource Limits

No protection against resource exhaustion or letting LLMs run wild 💰💰💰💰

A06: Practical Takeaways

✅ **Threat Modeling** - Identify threats early in design phase
✅ **Secure Design Patterns** - Use proven security patterns
✅ **Defense in Depth** - Multiple layers of security controls
✅ **Security Requirements** - Define security from the start
✅ **User Story Integration** - Security in every user story

Key Concept: Secure design prevents classes of vulnerabilities

A07: Authentication Failures

The Risk:

- 36 CWEs mapped
- Improvement due to standardized frameworks
- Still critical when it fails

What Breaks:
Confirming user identity and maintaining sessions

A07: Common Failures

Credential Stuffing

Automated login attempts with breached credentials

Weak Password Policy

Allowing "password123"

Missing MFA

Single factor authentication only

Session Fixation

Not rotating session IDs after login

Exposed Session IDs

Session tokens in URLs

A07: Practical Takeaways

✅ **Multi-Factor Authentication** - Implement for all users
✅ **Strong Password Requirements** - Length over complexity
✅ **Breach Detection** - Check against known breached passwords
✅ **Rate Limiting** - Limit login attempts
✅ **Secure Session Management** - Random IDs, secure flags, rotation

Key Concept: Identity is the new perimeter

A08: Software or Data Integrity Failures

The Risk:

- Different from supply chain failures (A03)
- Focus on code and data that isn't verified
- Includes insecure deserialization

What It Is:
Trusting code or data without verification

A08: Attack Scenarios

Insecure Deserialization

Manipulating serialized objects to execute code

Unsigned Updates

Accepting updates without signature verification

CI/CD Compromise

Malicious code in build pipeline

Integrity Violations

Modified data not detected

Auto-Update Vulnerabilities

Applications updating from insecure sources

A08: Practical Takeaways

✅ **Digital Signatures** - Sign all software and critical data
✅ **Verify Integrity** - Check signatures and hashes
✅ **Secure CI/CD Pipeline** - Segregated, monitored build environments
✅ **Avoid Native Deserialization** - Use safer data formats (JSON)
✅ **Integrity Monitoring** - Detect unauthorized changes

Key Concept: Trust but verify - especially for code and critical data

A09: Security Logging & Alerting Failures

The Risk:

- Name change from "Logging and Monitoring"
- Emphasizes the need for alerts that drive action
- Chronically underrepresented in testing data

The Problem:

Can't detect or respond to what you can't see

A09: What's Missing

Insufficient Logging

Critical events not logged (logins, access control failures)

Unclear Log Messages

Logs that don't provide actionable information

No Alerting

Great logs but nobody watching

Log Injection

Attackers manipulating log entries

Inadequate Retention

Logs deleted before incident detection

A09: Practical Takeaways

✅ **Log Security Events** - Authentication, access control, input validation failures
✅ **Actionable Alerts** - Automated alerts on suspicious patterns
✅ **Centralized Logging** - Tamper-resistant, centralized collection
✅ **Log Format Standards** - Machine-readable formats for SIEM
✅ **Incident Response Plan** - Define who responds to alerts and how

Key Concept: Logging without alerting is like having cameras with no monitors

A10: Mishandling of Exceptional Conditions

The Risk:

- 24 CWEs focused on error handling
- Improper error handling, logical errors
- Failing open instead of closed

What Happens:
Systems behave insecurely under unexpected conditions

A10: Failure Scenarios

Information Disclosure

Detailed error messages revealing system architecture

Fail Open

Security check fails, grants access anyway

Unhandled Exceptions

Application crashes reveal sensitive information

Logical Errors

Edge cases not considered in code

Race Conditions

Time-of-check to time-of-use vulnerabilities

A10: Practical Takeaways

✅ **Fail Securely** - Default deny on errors
✅ **Generic Error Messages** - Don't reveal system details to users
✅ **Detailed Internal Logs** - Log full details for investigation
✅ **Exception Handling** - Comprehensive error handling
✅ **Edge Case Testing** - Test boundary conditions and error paths

Key Concept: Security must be maintained even when things go wrong

OK, that was a lot

Cross-Cutting Themes

Defense in Depth

Multiple layers of security controls

Shift Left

Security early in development lifecycle

Least Privilege

Minimum necessary access at all times - opt-in to permissions

Secure by Default

Make default configuration secure - opt-out of security controls

Continuous Verification

Ongoing testing and monitoring

Key Takeaways

1. **OWASP Top 10 is your baseline** - Start here but don't stop here
2. **Root causes matter more than symptoms** - Fix the underlying issues
3. **Security is everyone's job** - Developers, operations, management
4. **Modern threats require modern defenses** - Supply chain and design matter
5. **Fail securely** - Security must work even when things break

Thank You

Resources:

- OWASP Top 10: https://owasp.org/Top10
- OWASP Projects: https://owasp.org/projects
- OWASP Cheat Sheets: https://cheatsheetseries.owasp.org
- This presentation: https://github.com/FlyingDan/presentations [later tonight]