

Two decades of secure software development: shifting left, right, and down

Erik Poll

Radboud University Nijmegen

Early 2000s: cybersecurity becoming a problem

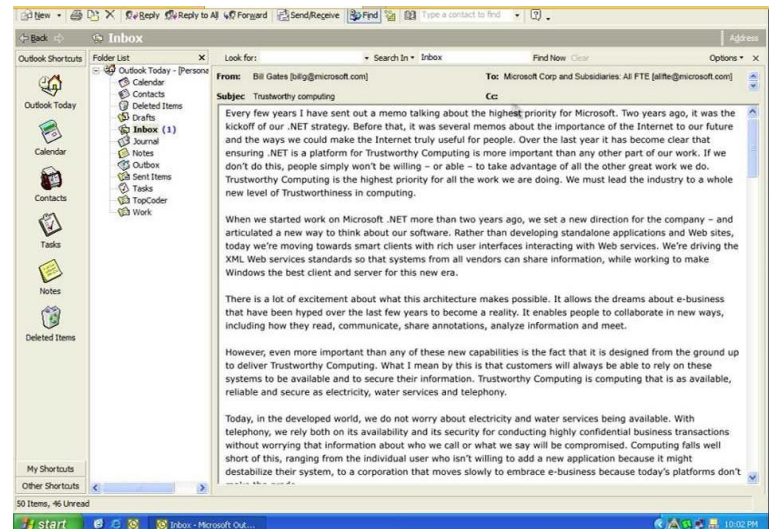
- OWASP founded (2001)



- Bill Gates makes security a top priority for Microsoft (2002)

Highest priority for Microsoft:
... trustworthiness ...

- Availability
- Security
- Privacy

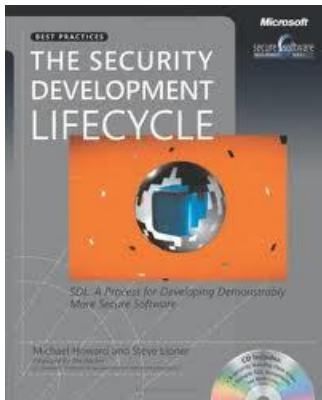


- NWO project **SAMASC** started
(Security Analysis for Multi-Applet Smart Cards)

First efforts to tackle software security (aka application security)



OWASP Top 10 (2003)



SDL by Microsoft (2004)

20 years later

Legislation for software security
with e.g. EU **Radio Equipment Directive (RED)**
and **Cyber Resilience Act (CRA)**
to complement **GDPR, NIS2,**



CRA requires

products with digital elements shall:

- (a) be made available on the market without known exploitable vulnerabilities;

20 years later: hard to see the forests for the trees

- lots of **vulnerabilities** (CVEs) and **vulnerability categories** (CWEs)

E.g. buffer overflows, SQL injection, XSS, CRSF, SSRF, integer overflow, TOCTOU, XXE, LDAP injection, Xpath injection, CRFL injection, PHP remote file inclusion, HTTP response splitting, EAR, type confusion, insecure deserialization, missing authentication, use-after-free, double-free, NULL pointer dereference, use of default password, use of default cryptographic key,

...

Insecure Setting of Generative AI/ML Model Inference Parameters (CWE-1434)

- lots of **'standards'**

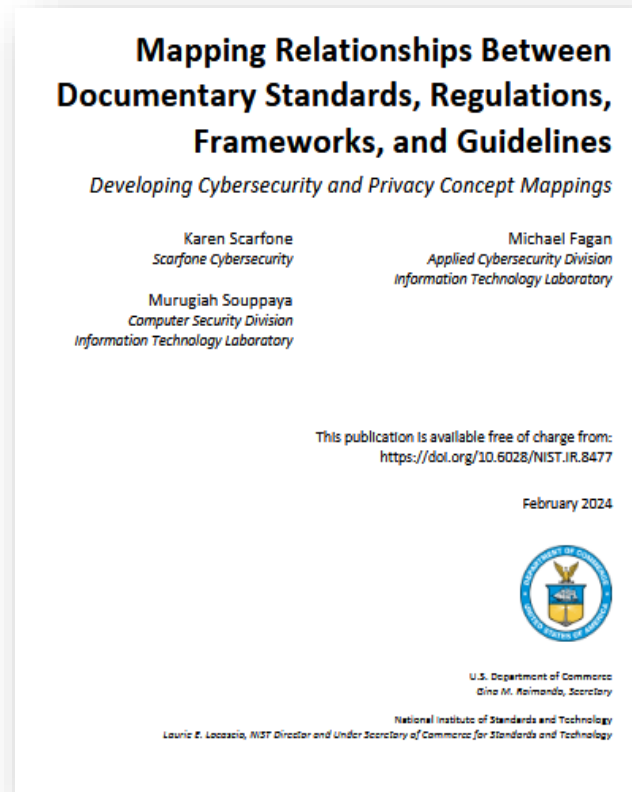
Microsoft SDL, Touchpoints, BSIMM12, PCI SSLLC, OWASP CLASP, OWASP SAMM, OWASP ASVS, OWAPS SCVS, BSA Framework for Secure Software, IDA SOAR, ISA/IEC 62443, SafeCode Fundamental Practices For Secure Software Development, SafeCode SIC, SafeCode TPC, CNCF FSSCP, EO 14028, NIST SSDF, IR8397, SP800-52, SP800-160, SP800-161, NIST CSF, NIST LAB,

...

EN 18031 (2025, for **RED**), **EN 40000** (2026, for **CRA**)

How to cope with ever more security standards?

In 2024 NIST released a methodology for mapping relations between security standards (IR 8477)



How to cope with ever more security standards?

Providing explicit cross-references to other standards helps

NIST Special Publication 800-218

Secure Software Development Framework (SSDF) Version 1.1:

*Recommendations for Mitigating
the Risk of Software Vulnerabilities*

Recent example: **NIST SSDF (2022)**

combines detailed ‘activities’ and ‘practices’ from 25 other standards with explicit cross-references, incl.

Microsoft SDL, BSIMM12, BSA Framework for Secure Software, OWASP SAMM, OWASP ASVS, OWAPS SCVS, SafeCode Fundamental Practices For Secure Software Development, SafeCode SIC, SafeCode TPC, Payment Card Industry SSLC, NIST IR8397, SP800-52, SP800-160, SP800-161, NIST CSF, NIST LAB, CNCF FSSCP, EO14028, IDA SOAR, ISA/IEC 62443, ...

How to cope with ever more security standards?

Simply refer to other standards for the details,
as **EN 18031** does to define security-by-design:

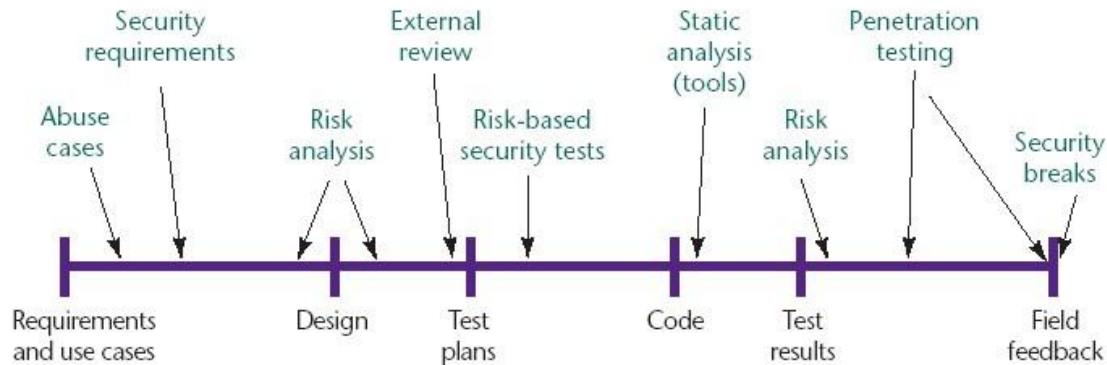
A.2.2 Security by design

Effective security management requires established security by design processes, which is not covered by this document which defines common security requirements for the equipment. Examples of security by design process standards which would aid in the ability to satisfy the security requirements include:

- IEC 62443-4-1[1]: Security for industrial automation and control systems – Part 4-1: Secure product development lifecycle requirements
- NIST 800-160[16]: Systems Security Engineering; Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems
- NIST 800-218[17]: Secure Software Development Framework (SSDF)
- Microsoft Security Development Lifecycle (SDL)
- SAFECODE Fundamental Practices for Secure Software Development
- GSMA FS.16 NESAS Development and Lifecycle Security Requirements

Broad distinction: process vs product

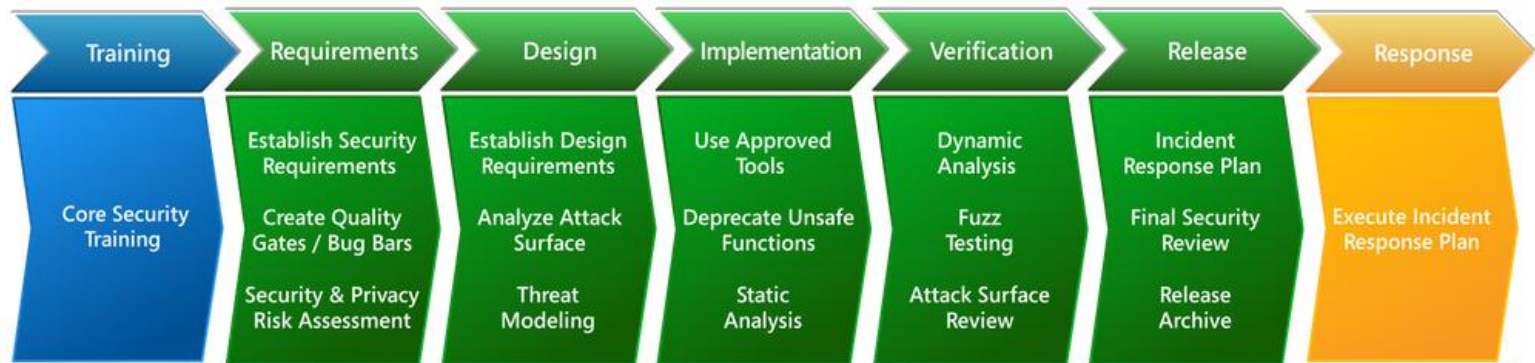
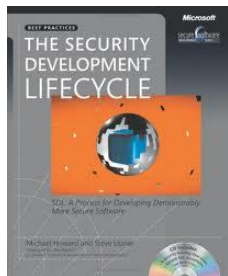
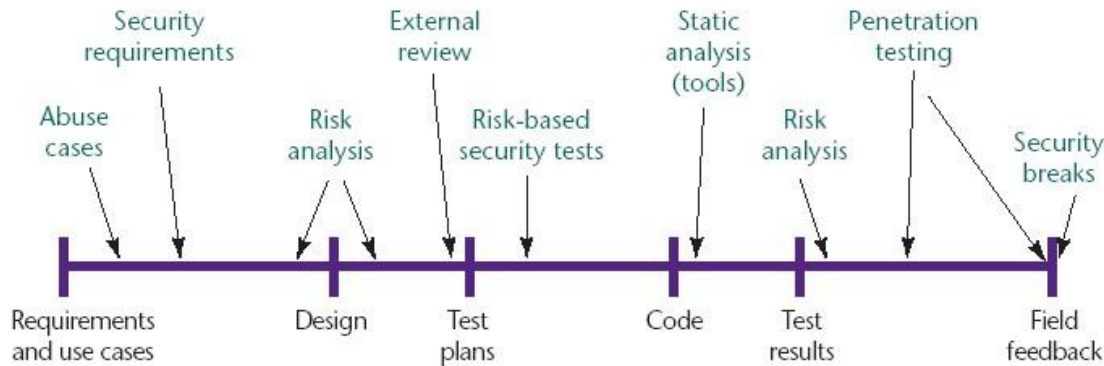
- Guidance for the software development *process*



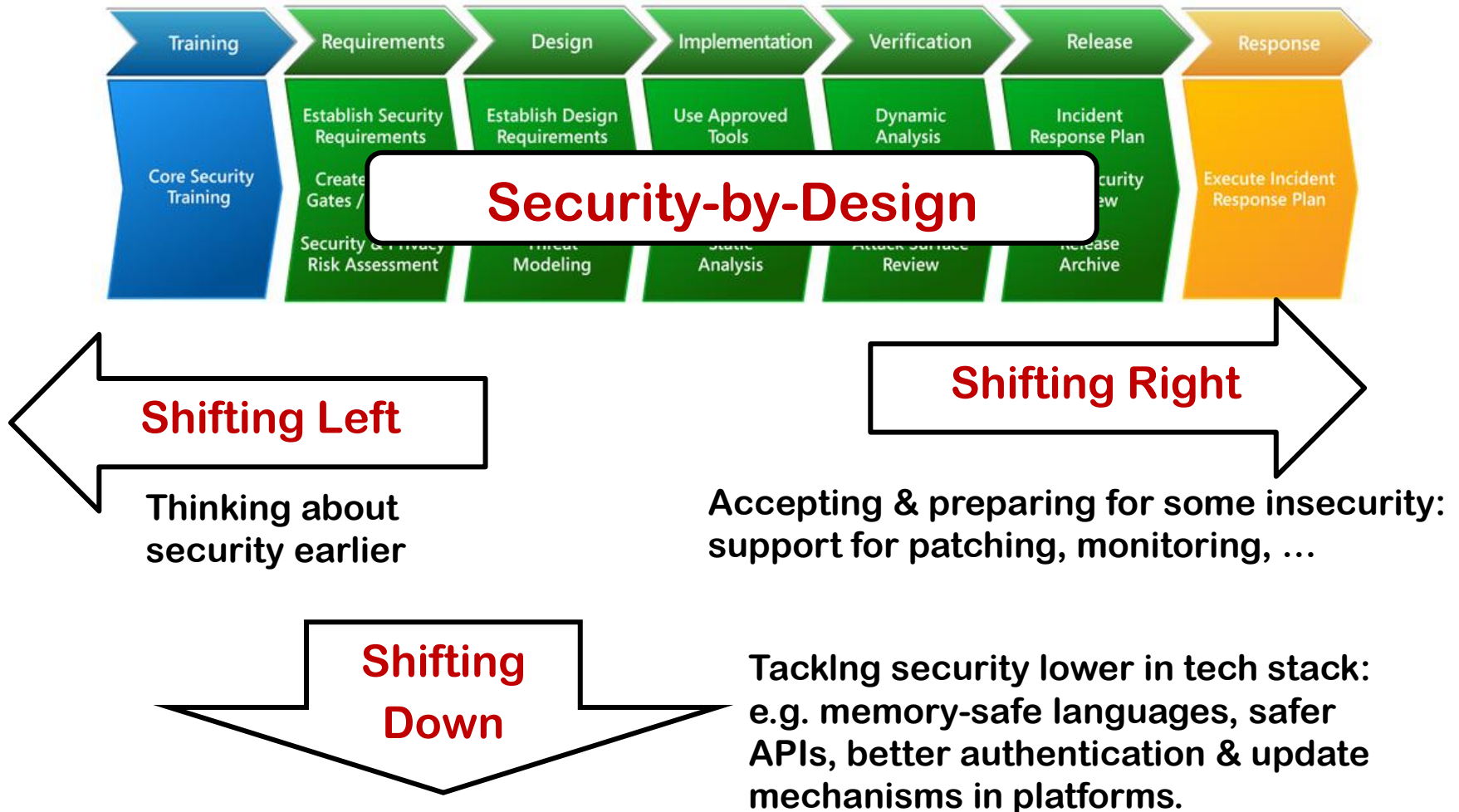
- Guidance for the software *product*

- Negative advice** – about common flaws
 - e.g. OWASP Top 10, CWE Top 25,
- Positive advice** – about how to avoid this
 - e.g. OWASP ASVS, OWASP SCVS, [LangSec](#), ...

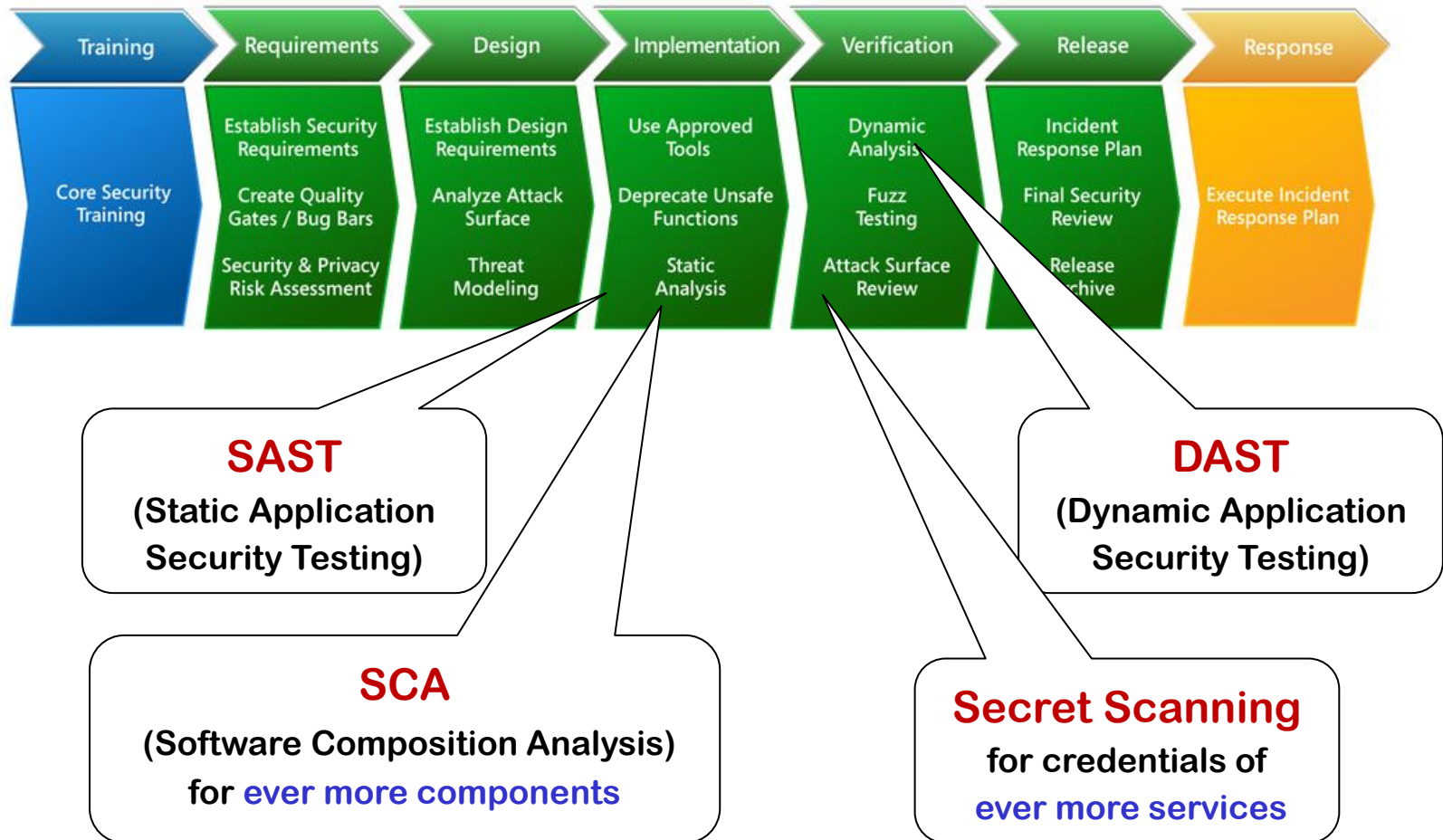
Central idea for process: security activities *throughout* development lifecycle



What's changed? New *slogans & trends*

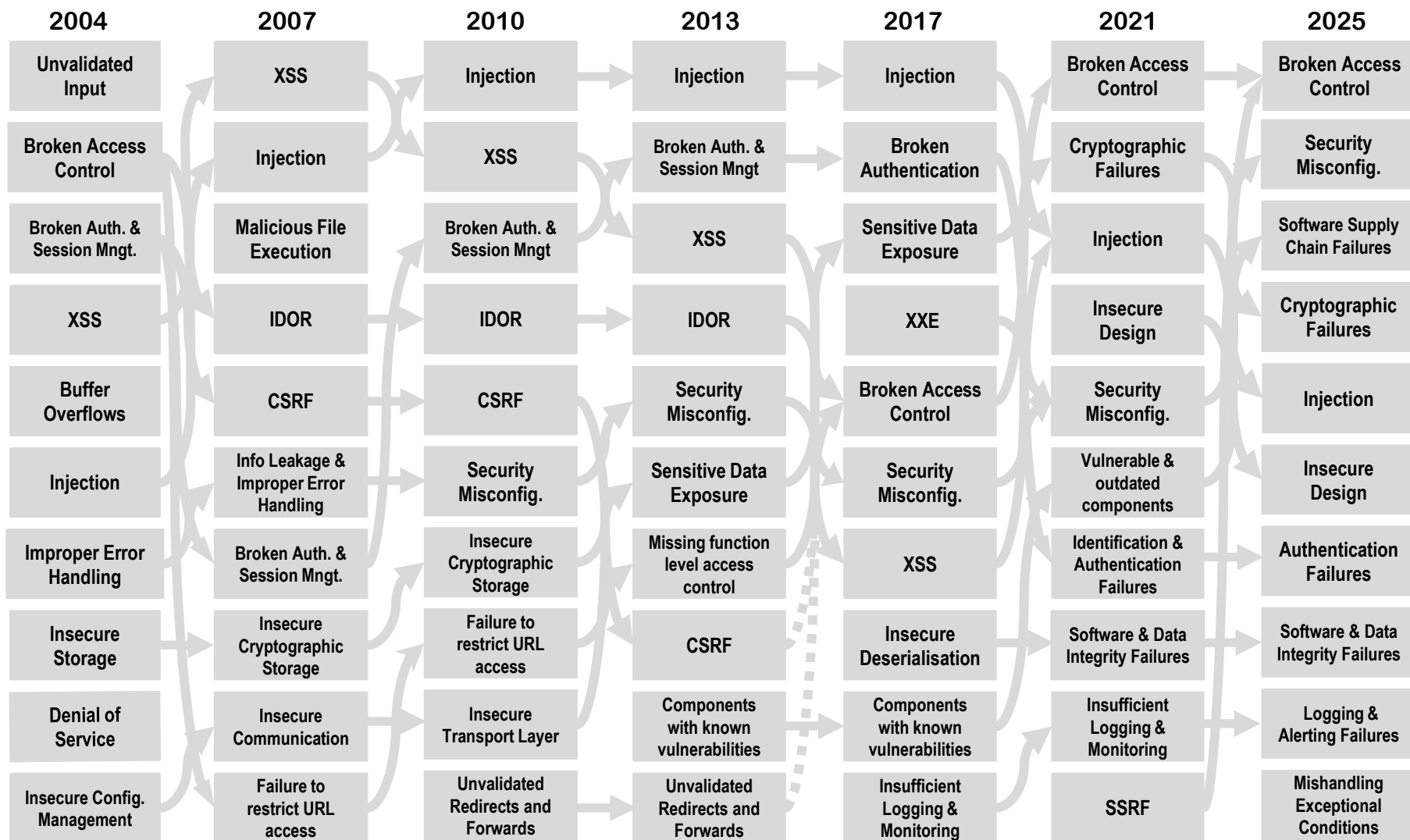


What's changed? New *acronyms & tools*



Have things improved?

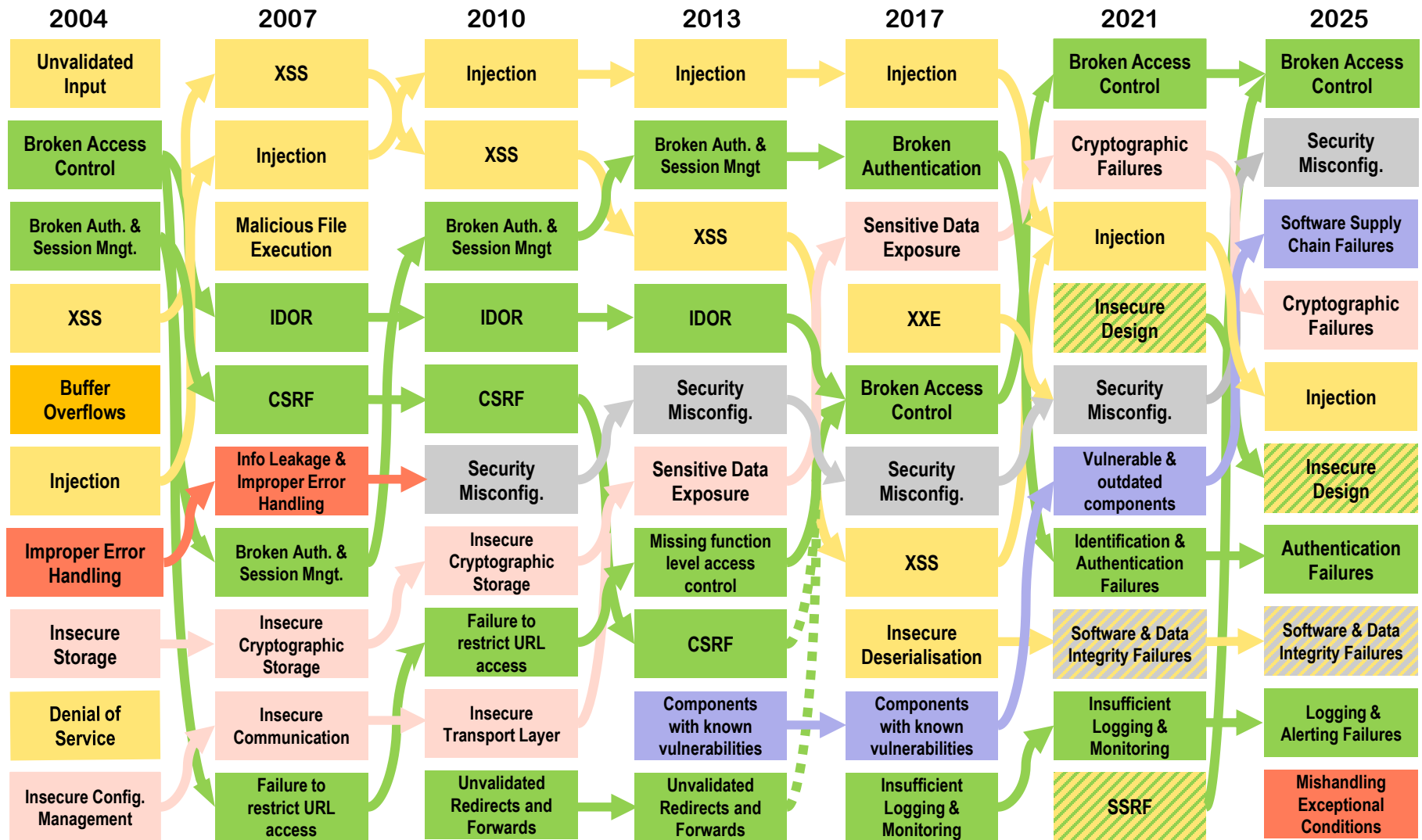
Evolution of the OWASP Top 10



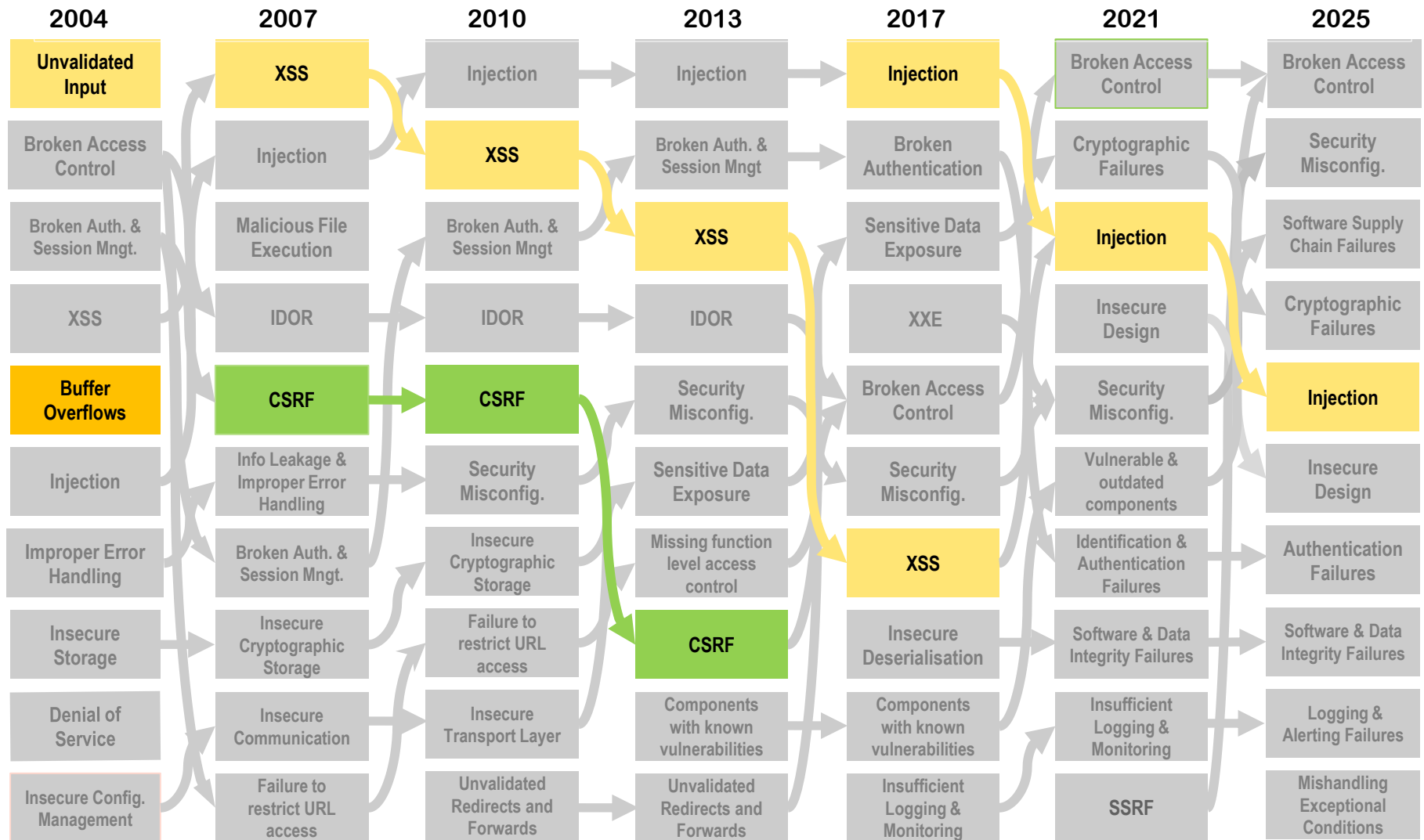
memory corruption

input handling

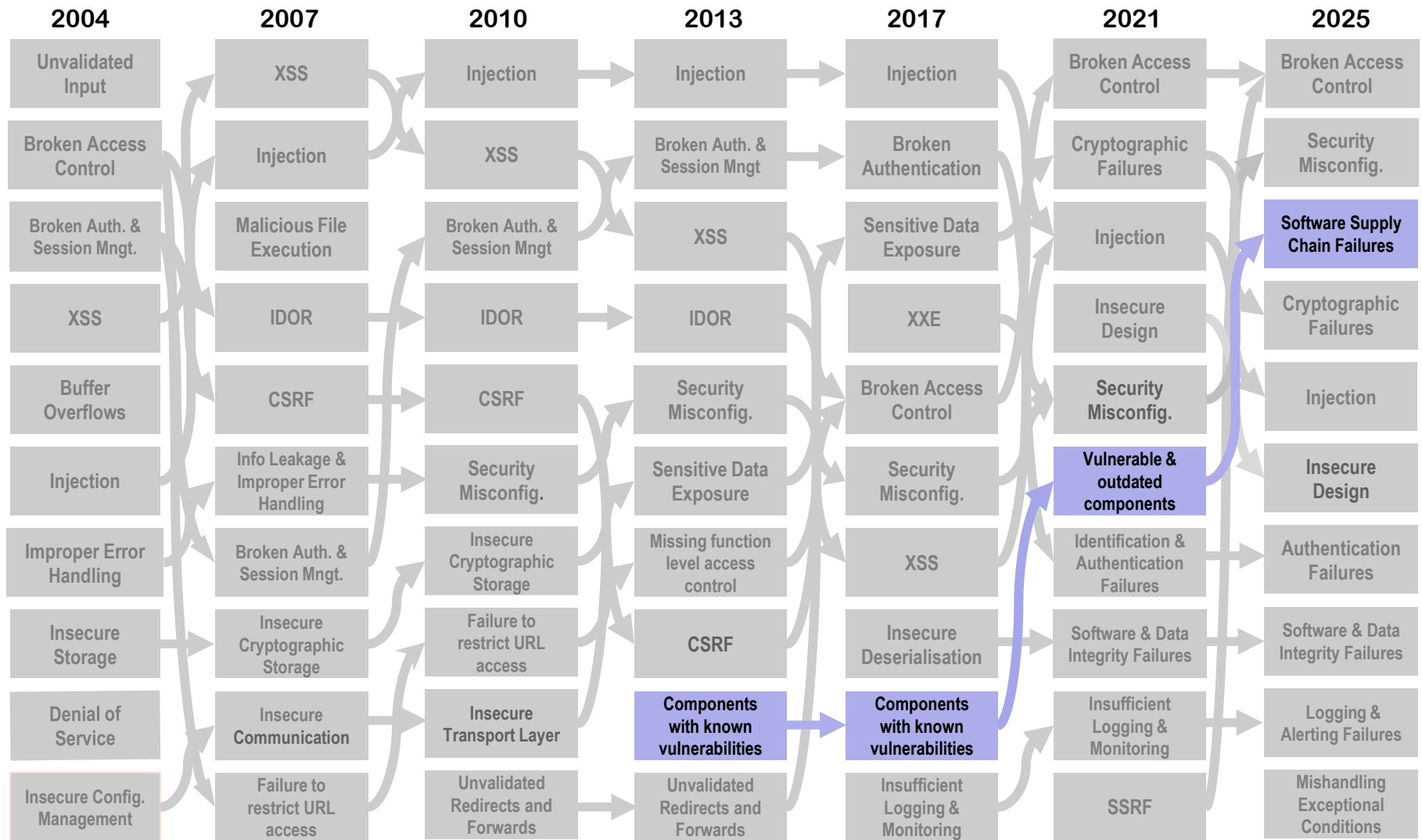
access control



Downward trends, thanks in part to shifting down



Upward trends, due to more complex supply chain



Have things improved? Looking at CVEs

CWE TOP 25 [2010]

- 1 Cross-Site Scripting
- 2 SQL injection
- 3 Classic Buffer overflow
- 4 Cross-Site Request Forgery (CSRF)
- 5 Improper Authorization
- 6 Security Decision based on Untrusted Input
- 7 Path traversal
- 8 Unrestricted Upload of File with Dangerous Type
- 9 OS command injection
- 10 Missing Encryption
- 11 Use of Hard-coded Credentials
- 12 Buffer Access with Incorrect Length
- 13 PHP file injection
- 14 Improper Validation of Array Index
- 15 Improper Error Handling
- 16 Information Leak through Error Message
- 17 Integer Overflow
- 18 Incorrect Calculation of Buffer Size
- 19 Missing Authentication
- 20 Download of Code without Integrity Check
- 21 Incorrect Permission
- 22 Allocation of Resources without Throttling
- 23 Open Redirect
- 24 Use of Broken or Risky Cryptography
- 25 Race Condition

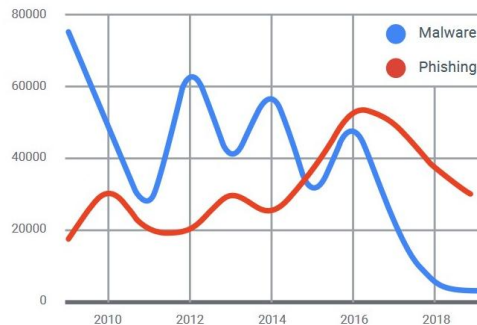
CWE TOP 25 [2025]

- 1 Cross-Site Scripting
- 2 SQL Injection
- 3 Cross-Site Request Forgery (CSRF)
- 4 Missing Authorization
- 5 Out-of-bounds Write
- 6 Path traversal
- 7 Use After Free
- 8 Out-of-bounds Read
- 9 OS command injection
- 10 Code injection
- 11 Classic Buffer Overflow
- 12 Unrestricted Upload of File with Dangerous Type
- 13 NULL pointer dereference
- 14 Stack-based buffer overflow
- 15 Deserialization of Untrusted Data
- 16 Heap-based Buffer overflow
- 17 Incorrect Authorization
- 18 Improper Input Validation
- 19 Improper Access Control
- 20 Exposure of Sensitive Information
- 21 Missing Authentication
- 22 Server-Side Request Forgery (SSRF)
- 23 Command Injection
- 24 Authorization Bypass with User-Control Key
- 25 Allocation of Resources without Throttling

The good news

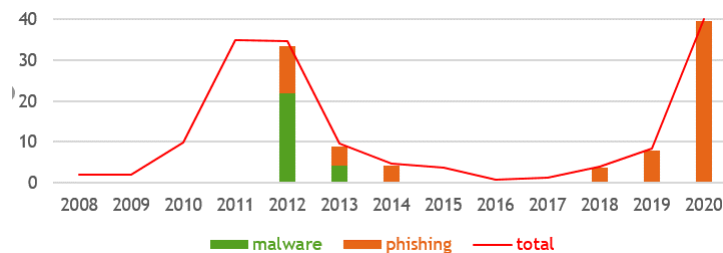
Software exploits no longer main root cause of problem

Malware vs phishing



[Source: Google Transparency Report]

Internet banking losses in Netherlands



[Source: Betaalvereniging]

The bad news?

AI is a major game changer in finding & exploiting software flaws

The Hacker News

Anthropic's Claude Mythos Finds Thousands of Zero-Day Flaws Across Major Systems

Ravie Lakshmanan Apr 08, 2026

We can no longer rely on security by obscurity for software flaws

Will this benefit attackers or defenders more? In short/long term

Will this replace SAST and DAST?