Twenty years of secure software development

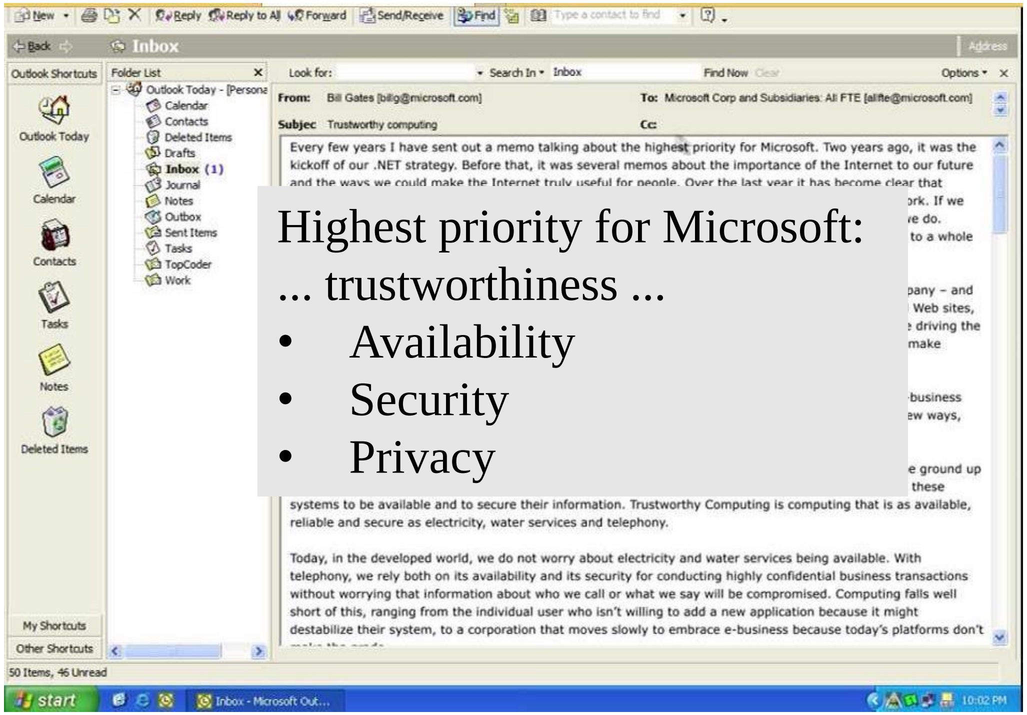
Erik Poll

NWA project INTERSCT.

Radboud University Nijmegen

Early 2000s

IT community realises that (cyber)security is becoming a problem and software is 'to blame'



From: Bill Gates [billg@microsoft.com]   **To:** Microsoft Corp and Subsidiaries: All FTE [allfte@microsoft.com]

Subjec Trustworthy computing   **Cc:**

Every few years I have sent out a memo talking about the highest priority for Microsoft. Two years ago, it was the kickoff of our .NET strategy. Before that, it was several memos about the importance of the Internet to our future and the ways we could make the Internet truly useful for people. Over the last year it has become clear that

ork. If we
ve do.
to a whole

pany – and
Web sites,
driving the
make

business
ew ways,

e ground up
these

systems to be available and to secure their information. Trustworthy Computing is computing that is as available, reliable and secure as electricity, water services and telephony.

Today, in the developed world, we do not worry about electricity and water services being available. With telephony, we rely both on its availability and its security for conducting highly confidential business transactions without worrying that information about who we call or what we say will be compromised. Computing falls well short of this, ranging from the individual user who isn't willing to add a new application because it might destabilize their system, to a corporation that moves slowly to embrace e-business because today's platforms don't

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

- Availability
- Security
- Privacy

2002 Email by Bill Gates to all Microsoft employees

OWASP
Open Web Application
Security Project

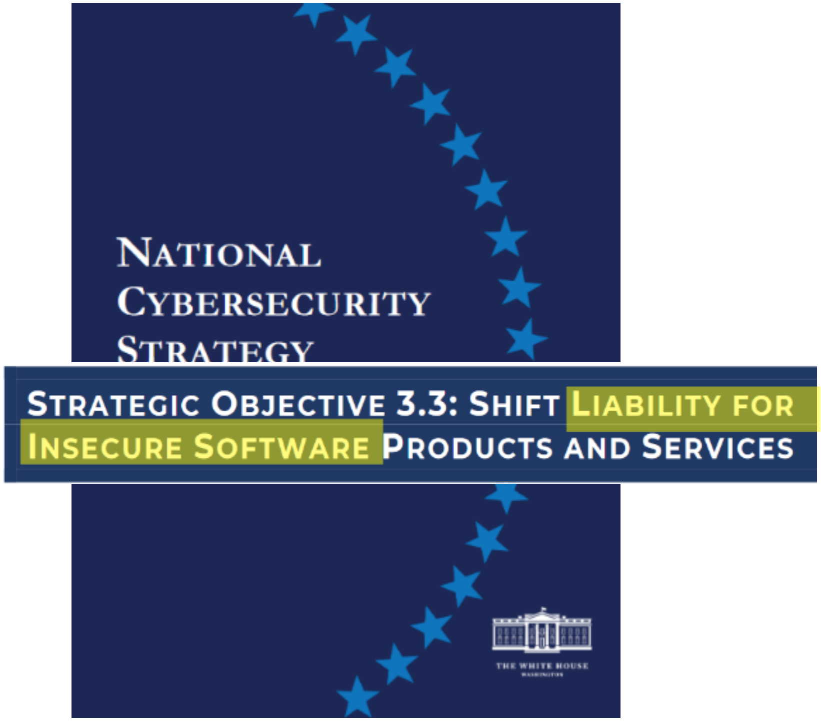
founded 2001

Twenty years later

Governments announce regulation for software security

"no known exploitable vulnerabilities"

Complements NIS2
Broader in scope than RED
(Radio Equipment Directive)



Strategic Objective 3.3: Shift Liability for Insecure Software Products and Services

(2023)

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

Lots of standards, frameworks, guidelines, tools, Top N lists, ...

- forest of vulnerabilities (CVEs)

  with CVSS, KEV, EPSS, CPR, SSVC, ... to navigate it

- forest of vulnerability categories (CWEs)

  eg. OWASP Top 10, CWE Top 25 , ...

- forest of secure development technologies

  eg. SDL, SAMM, BSIMM, NIST SSDF, ...

  focused on the process

- forest of security tools

  DAST (incl. fuzzing), SAST, SCA, SecretScanning, ...

- forest of security requirements

  eg. OWASP ASVS, OWASP SCVS, ...

  focused on the product

- ...

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

Lots of standards, frameworks, guidelines, tools, Top N lists, ...

- forest of vulnerabilities (CVEs)

  with CVSS, KEV, EPSS, CPR, SSVC, ... to navigate it

- forest of vulnerability categories (CWEs)

  eg. OWASP Top 10, CWE Top 25 , ...

- forest of secure development technologies

  eg. SDL, SAMM, BSIMM, NIST SSDF, ...

  focused on the process

- forest of security tools

  DAST (incl. fuzzing), SAST, SCA, SecretScanning, ...

- forest of security requirements

  eg. OWASP ASVS, OWASP SCVS, ...

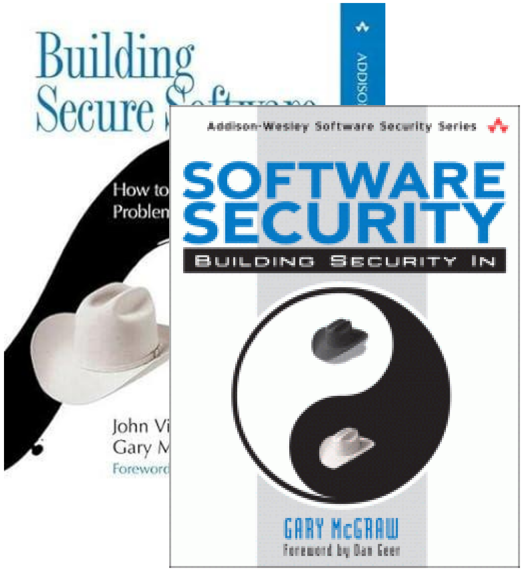
  focused on the product

- *...*

The process

‘methodologies’

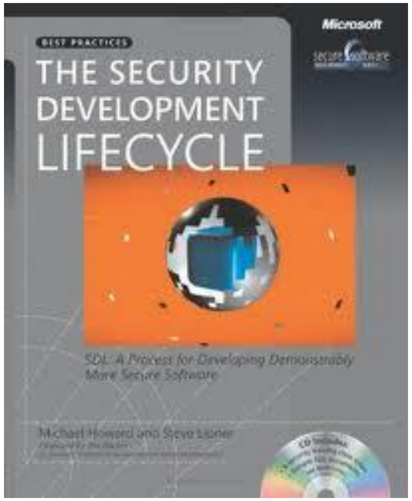
Early 2000s: Secure development methodologies



`Building Security In’ aka
Cigital Touchpoints by Gary McGraw

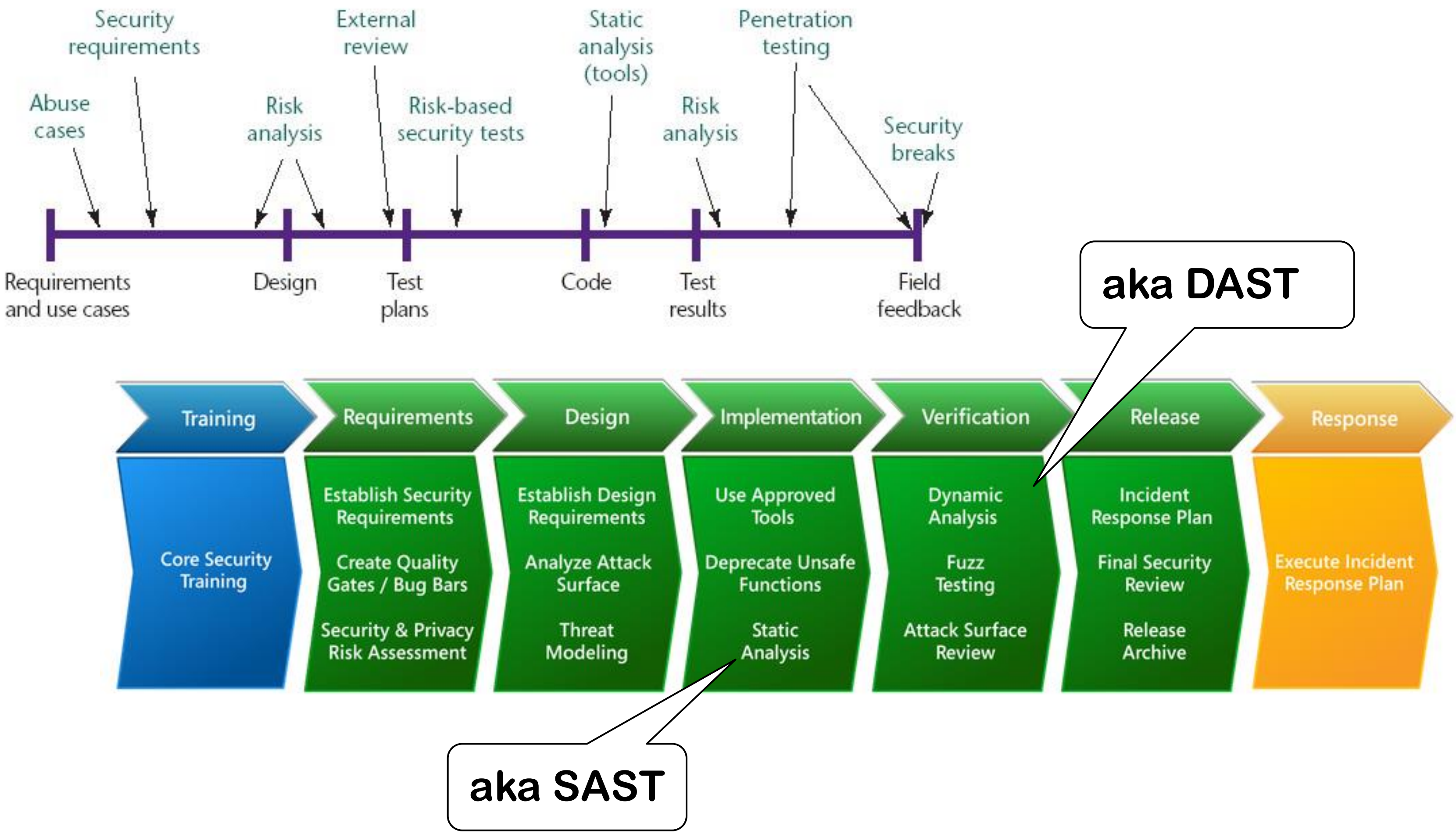
OWASP
Open Web Application
Security Project

CLASP by OWASP

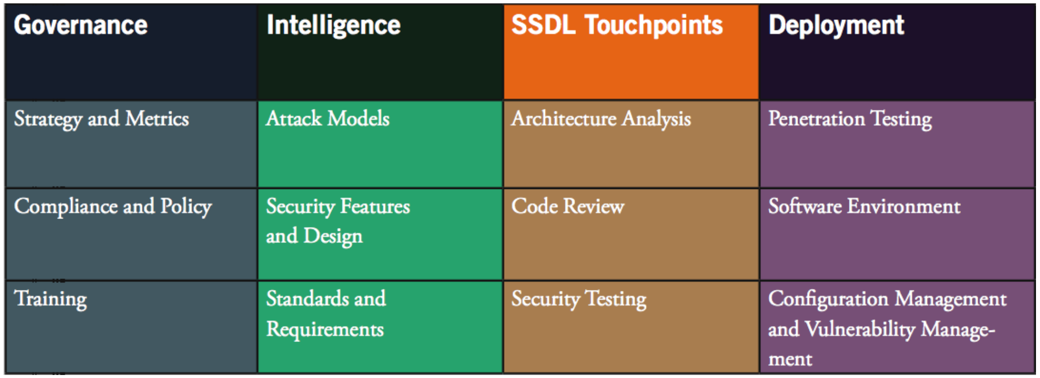

BEST PRACTICES
THE SECURITY DEVELOPMENT LIFECYCLE
Microsoft
SDL: A Process for Developing Demonstrably More Secure Software
Michael Howard and Steve Lipner

SDL by Microsoft (2004)

Key idea:
security activities *throughout* development lifecycle



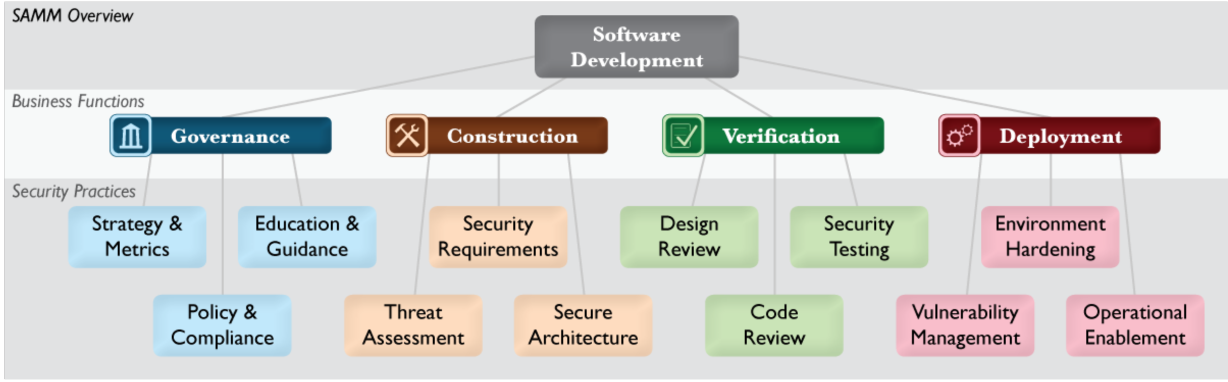
Maturity models for this



Governance	Intelligence	SSDL Touchpoints	Deployment
Strategy and Metrics	Attack Models	Architecture Analysis	Penetration Testing
Compliance and Policy	Security Features and Design	Code Review	Software Environment
Training	Standards and Requirements	Security Testing	Configuration Management and Vulnerability Management

- **BSIMM**
  - by Synopsis, since 2009
  - lists 126 activities grouped in 12 practices across 4 domains
  - to compare methodologies & measure maturity

- **OWASP SAMM**



SAMM Overview

Software Development

Business Functions

Governance

Construction

Verification

Deployment

Security Practices

Strategy & Metrics

Education & Guidance

Policy & Compliance

Security Requirements

Threat Assessment

Secure Architecture

Design Review

Security Testing

Code Review

Environment Hardening

Vulnerability Management

Operational Enablement

What’s changed in these methodologies in the past 20 years?

New slogans

- **Shifting Left**

  attention to security to earlier in the development lifecycle

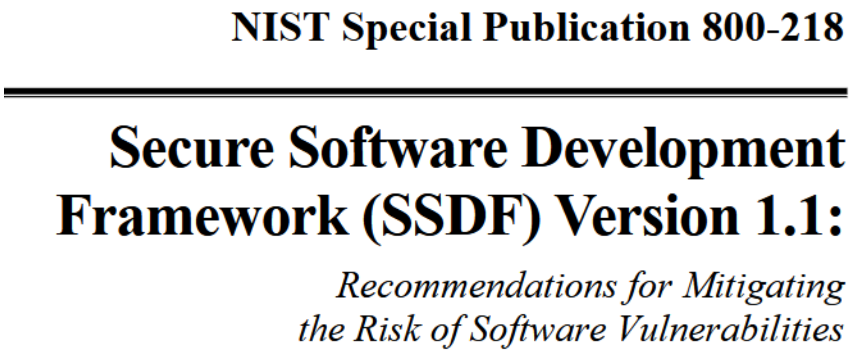
- **Security by Design**

  This does not just mean security in *the design phase*,
  but security 'on purpose' in *all phases of the development cycle*

- **Security by Default**

More of the same

Many more methodologies,
all mentioning the same or similar 'practices' & 'activities'



NIST Special Publication 800-218

Secure Software Development Framework (SSDF) Version 1.1:

Recommendations for Mitigating the Risk of Software Vulnerabilities

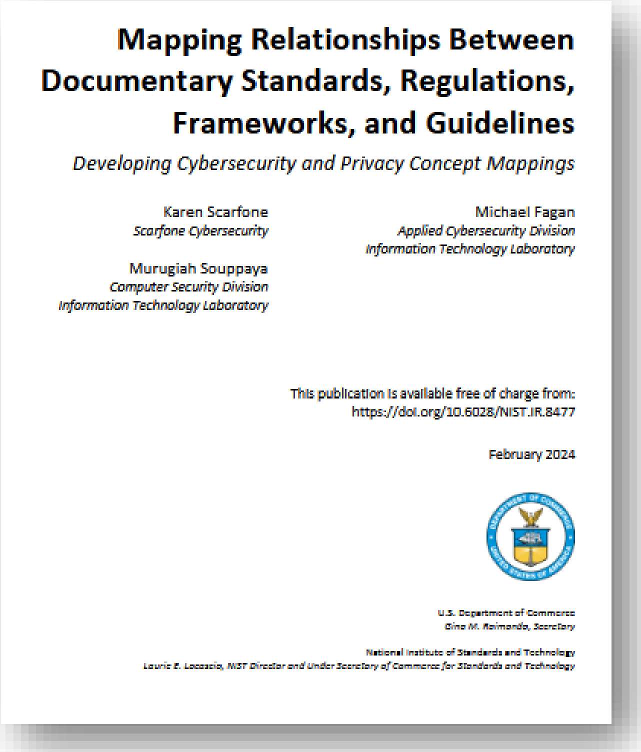
Recent example: NIST SSDF (2022)

draws from 25 other standards:

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

How to cope with ever more security standards?

- OWASP OpenCRE initiative provides mappings between security standards [https://www.opencre.org]
- In 2024 NIST released a methodology for mapping relations between cybersecurity standards (IR 8477)



Mapping Relationships Between Documentary Standards, Regulations, Frameworks, and Guidelines

Developing Cybersecurity and Privacy Concept Mappings

Karen Scarfone
Scarfone Cybersecurity

Murugiah Souppaya
Computer Security Division
Information Technology Laboratory

Michael Fagan
Applied Cybersecurity Division
Information Technology Laboratory

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8477

February 2024

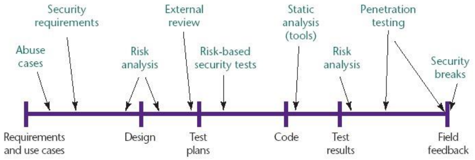
U.S. Department of Commerce
Gina M. Raimondo, Secretary

National Institute of Standards and Technology
Laurie E. Locascio, NIST Director and Under Secretary of Commerce for Standards and Technology

What’s changed in software engineering in the past 20 years?

1. Agile & DevOps

Security methodologies typically use waterfall model as frame of reference



How can we cope with Agile or DevOps?

We cannot do pen-test for every new feature or weekly release

No new activities, but changes in when & how often to do them

And: *more important to shift left!* Eg.

- use DAST and – further to the left – SAST
- train developers
- integrate SAST & DAST into CD/CI pipelines

With DevSecOps as new buzzword

2. Code repositories

Lots of code reuse from code repositories

github, Maven, PyPi, ...

New attack vector: supply chain attacks

Eg Log4J, SolarWinds, XZ utils

New countermeasures

1) SCA (Software Composition Analysis)
   static analysis tools to check software supply chain for CVEs

2) SBOM (Software Bill of Materials)
   Required by US executive Order 14028 (May 2021)

And more standards: OWASP SCVS, SafeCode Third Party Components, ...

3. ‘Services’

Software increasingly built using (cloud-based) *services* instead of libraries as *components* with SaaS, Service-Oriented Architectures, micro-services, cloud APIs

This introduces

- more attack surface
- need for authentication to cloud APIs

New security risk: leaking credentials (JWT tokens, AWS security tokens, ...)

New countermeasures:

1) SAST tools for secret scanning, eg TruffleHog
2) first proposals for SaaSBOMs

The product

as opposed to the *process*

‘guidelines’ & ‘standards’

Security advice for the software product

Methodologies & tools need to be fed with more concrete advice:

- **Lists of common vulnerabilities – *anti*-guidelines**
  Eg. OWASP Top 10, CWE Top 25, KEV Top 10, …
    - **Also Mobile Top 10, API Top 10, Top 10 for LLM applications, …**
- **Coding guidelines**
  Eg. SEI/CERT guidelines for C , C++, Java, Perl, Android, …
- **Standards with security requirements & controls**
    - **OWASP ASVS (Application Security Verification Standard)**
    - **CIP-overheid.nl 'Grip op SSD' normen**

  that can be used as metric, as guidance, or in procurement
- **Design patterns for security**

  Eg. Secure Builders for secure input handling

From don’ts to dos

Turning Top N lists of common flaws (dont’s)

OWASP TOP 10

MOST DANGEROUS Top 25 CWE SOFTWARE WEAKNESSES

into more constructive guidance (dos)

OWASP Application Security Verification Standard (ASVS)

OWASP | Standard

SCVS

Software Component Verification Standard

CIP centrum informatiebeveiliging en privacybescherming

Grip op Secure Software Development (SSD) Beveiligingseisen

Typical security flaws

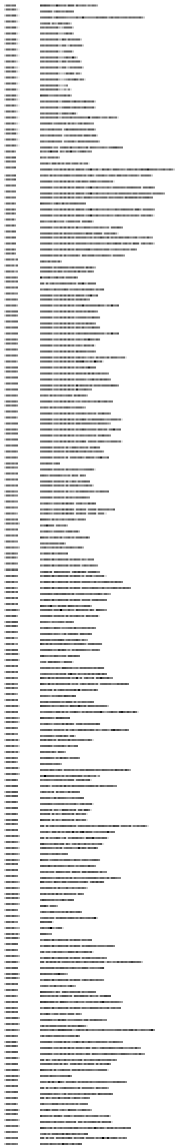
OWASP Top 10 [2017]

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XXE)
5. Broken Access Control
6. Security Misconfiguration
7. Cross-Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging & Monitoring

CWE TOP 25 [2022]

1 Out-of-bounds Write
2 Cross-site Scripting
3 SQL Injection
4 Improper Input Validation
5 Out-of-bounds Read
6 OS Command Injection
7 Use After Free
8 Path Traversal
9 Cross-Site Request Forgery (CSRF)
10 Unrestricted Upload of File with Dangerous Type
11 NULL Pointer Dereference
12 Deserialization of Untrusted Data
13 Integer Overflow or Wraparound
14 Improper Authentication
15 Use of Hard-coded Credentials
16 Missing Authorization
17 Command Injection
18 Missing Authentication for Critical Function
19 Improper Restriction of Bounds of Memory Buffer
20 Incorrect Default Permissions
21 Server-Side Request Forgery (SSRF)
22 Race Condition
23 Uncontrolled Resource Consumption
24 Improper Restriction of XML External Entity Reference
25 Code Injection

CWE TOP 1000



The big 3

Three big families of security problems:

CWE TOP 25 [2024]

1 Cross-site Scripting
2 Out-of-bounds Write
3 SQL Injection
4 Cross-Site Request Forgery (CSRF)
5 Path Traversal
6 Out-of-bounds Read
7 OS Command Injection
8 Use After Free
9 Missing Authorization
10 Upload of File with Dangerous Type
11 Code Injection
12 Improper Input Validation
13 Command Injection
14 Improper Authentication
15 Improper Privilege Management
16 Deserialization of Untrusted Data
17 Exposure of Sensitive Data
18 Incorrect Authorization
19 Server-Side Request Forgery (SSRF)
20 Improper Restriction of Operation in Buffer Bounds
21 NULL pointer deference
22 Use of Hard-coded Credentials
23 Integer Overflow
24 Uncontrolled Resource Consumption
25 Missing Authentication

The big 3

Three big families of security problems:

1) memory corruption

CWE TOP 25 [2024]

1 Cross-site Scripting
2 Out-of-bounds Write
3 SQL Injection
4 Cross-Site Request Forgery (CSRF)
5 Path Traversal
6 Out-of-bounds Read
7 OS Command Injection
8 Use After Free
9 Missing Authorization
10 Upload of File with Dangerous Type
11 Code Injection
12 Improper Input Validation
13 Command Injection
14 Improper Authentication
15 Improper Privilege Management
16 Deserialization of Untrusted Data
17 Exposure of Sensitive Data
18 Incorrect Authorization
19 Server-Side Request Forgery (SSRF)
20 Improper Restriction of Operation in Buffer Bounds
21 NULL pointer deference
22 Use of Hard-coded Credentials
23 Integer Overflow
24 Uncontrolled Resource Consumption
25 Missing Authentication

The big 3

Three big families of security problems:

1) memory corruption

2) input handling, esp.

- injection attacks
- improper input validation

CWE TOP 25 [2024]

1 Cross-site Scripting
2 Out-of-bounds Write
3 SQL Injection
4 Cross-Site Request Forgery (CSRF)
5 Path Traversal
6 Out-of-bounds Read
7 OS Command Injection
8 Use After Free
9 Missing Authorization
10 Upload of File with Dangerous Type
11 Code Injection
12 Improper Input Validation
13 Command Injection
14 Improper Authentication
15 Improper Privilege Management
16 Deserialization of Untrusted Data
17 Exposure of Sensitive Data
18 Incorrect Authorization
19 Server-Side Request Forgery (SSRF)
20 Improper Restriction of Operation in Buffer Bounds
21 NULL pointer deference
22 Use of Hard-coded Credentials
23 Integer Overflow
24 Uncontrolled Resource Consumption
25 Missing Authentication

The big 3

Three big families of security problems:

1) memory corruption

2) input handling, esp.

- injection attacks
- improper input validation

3) access control, incl.

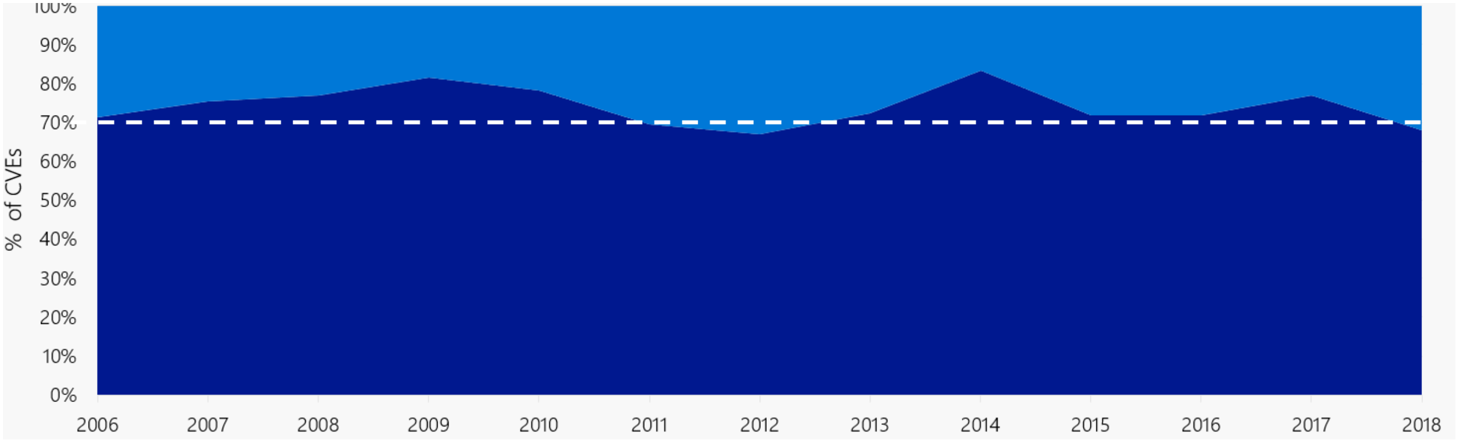
- authentication flaws
- authorisation flaws
- insufficient logging & monitoring

CWE TOP 25 [2024]

1 Cross-site Scripting
2 Out-of-bounds Write
3 SQL Injection
4 Cross-Site Request Forgery (CSRF)
5 Path Traversal
6 Out-of-bounds Read
7 OS Command Injection
8 Use After Free
9 Missing Authorization
10 Upload of File with Dangerous Type
11 Code Injection
12 Improper Input Validation
13 Command Injection
14 Improper Authentication
15 Improper Privilege Management
16 Deserialization of Untrusted Data
17 Exposure of Sensitive Data
18 Incorrect Authorization
19 Server-Side Request Forgery (SSRF)
20 Improper Restriction of Operation in Buffer Bounds
21 NULL pointer deference
22 Use of Hard-coded Credentials
23 Integer Overflow
24 Uncontrolled Resource Consumption
25 Missing Authentication

Memory corruption bugs

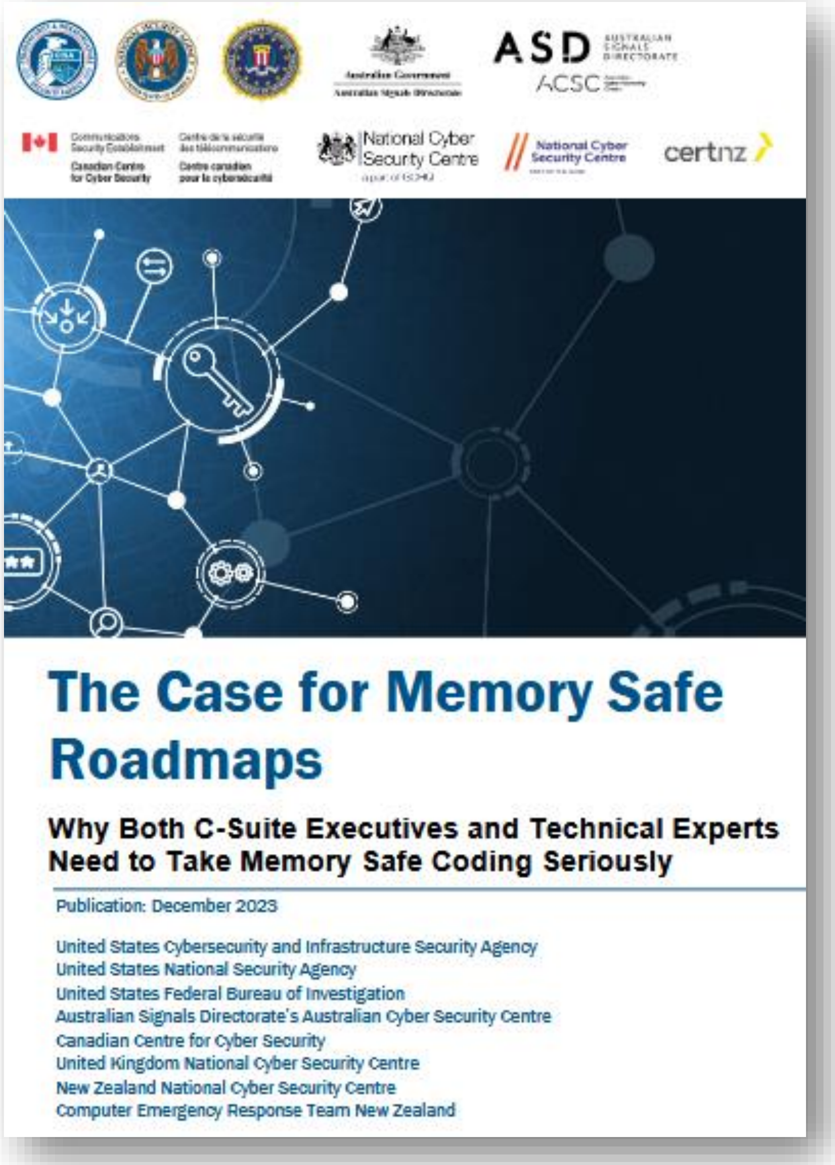
Tackling memory corruption bugs has been dismal failure



memory safety vs **non-memory safety** bugs at Microsoft

Only solution: move to memory safe languages, eg Rust

In Feb 2025 CISA & FBI declared memory corruption bugs as unforgivable bugs



The Case for Memory Safe Roadmaps

Why Both C-Suite Executives and Technical Experts Need to Take Memory Safe Coding Seriously

Publication: December 2023

United States Cybersecurity and Infrastructure Security Agency
United States National Security Agency
United States Federal Bureau of Investigation
Australian Signals Directorate's Australian Cyber Security Centre
Canadian Centre for Cyber Security
United Kingdom National Cyber Security Centre
New Zealand National Cyber Security Centre
Computer Emergency Response Team New Zealand

Input handling problems

- Common mistake: seeing input validation as the only or best solution. Output encoding & safer APIs may be better!

- Most input handling problems are PARSING problems

  a) buggy & insecure parsing of complex data formats.
     Eg buffer overflows in Flash, PDF, or OpenVPN parsers

  b) unintended parsing leading to injection attacks
     Eg user data being parsed as SQL command

  Aggrevated by *many*, *complex*, *poorly defined* data formats/input languages

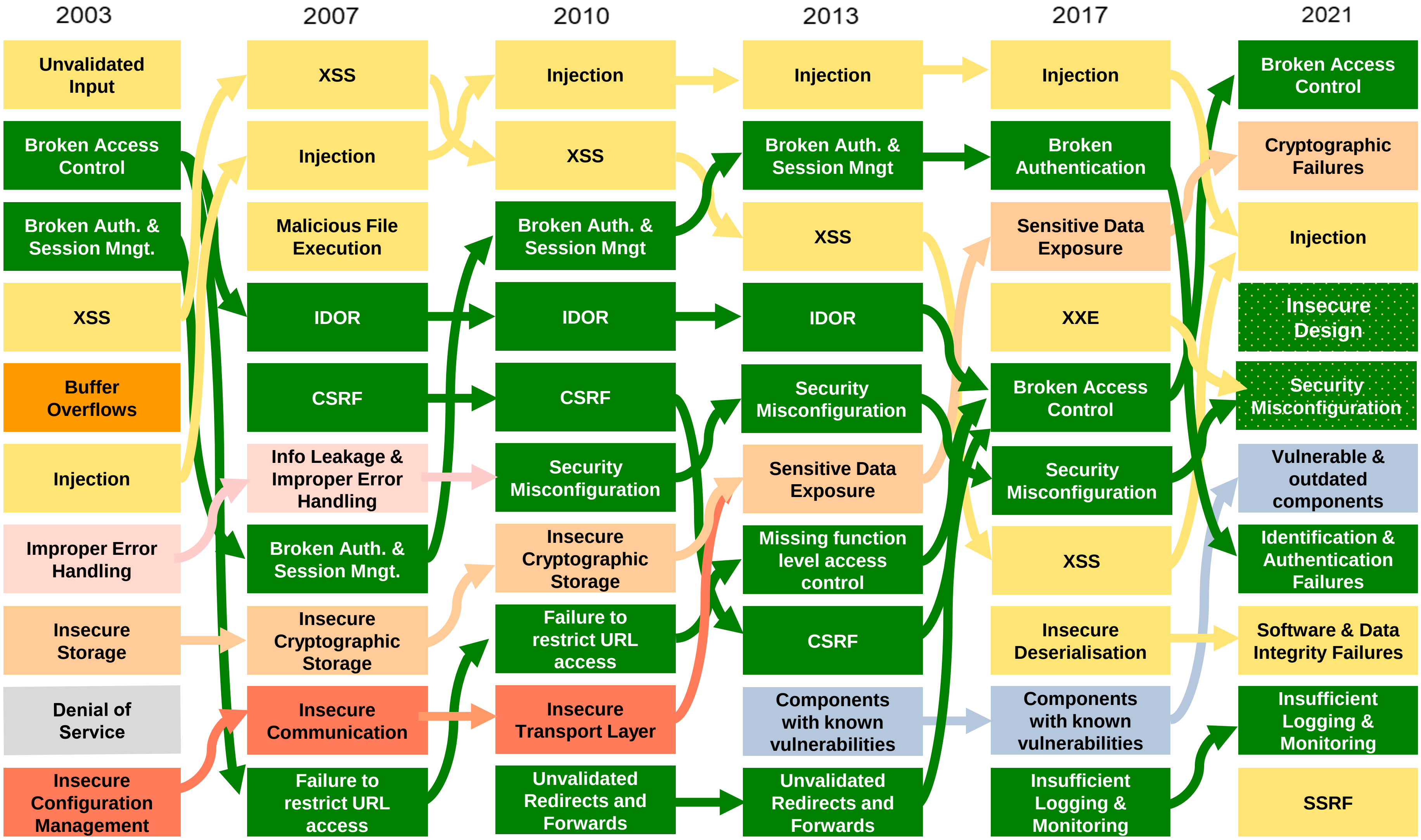
- We can structurally tackle these by

  a) LangSec: clearer specs of input formats & generated parser code

  b) safer APIs where API & type system prevent misinterpretation

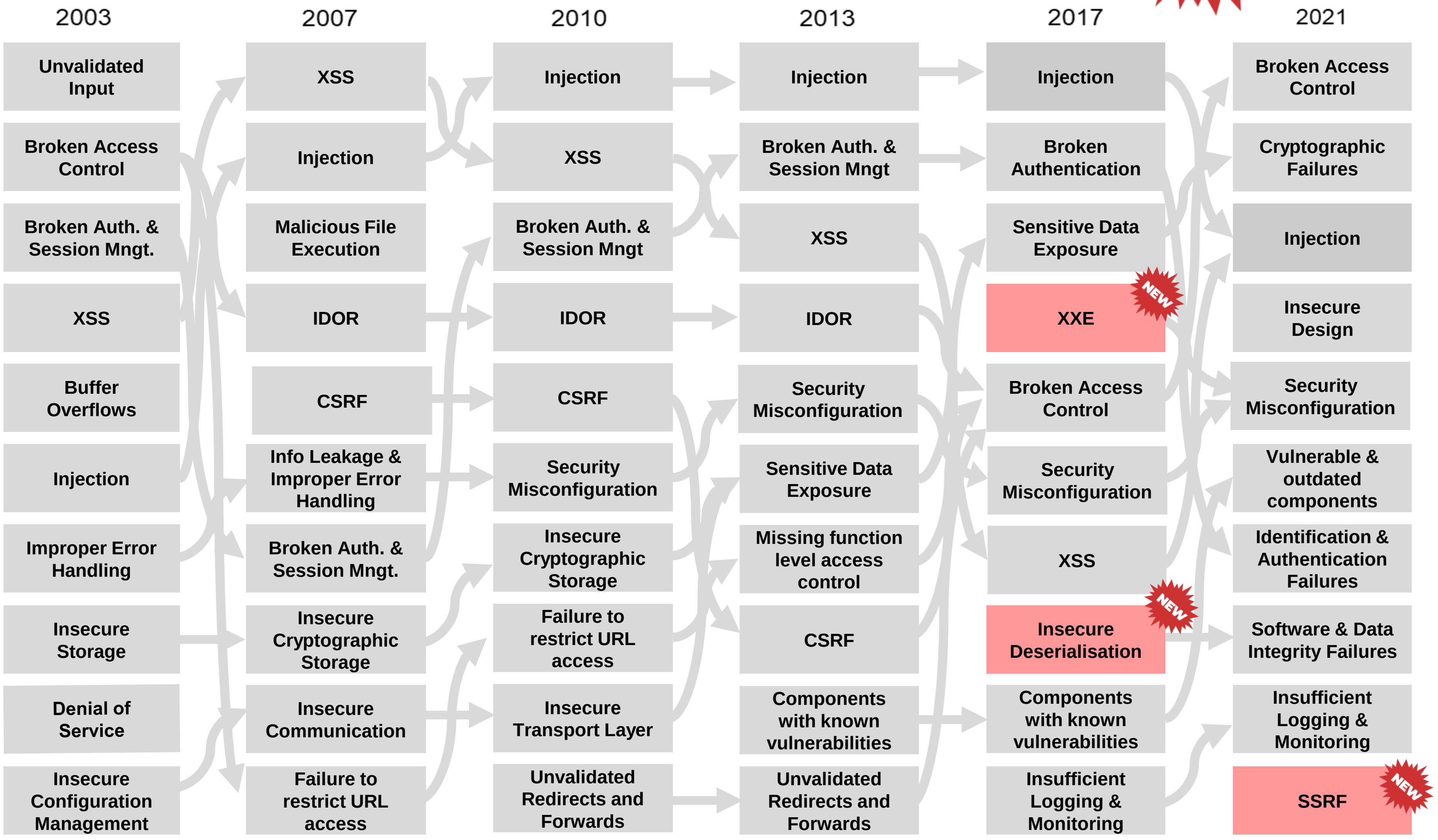
     Eg Google re-engineered Trusted Types DOM API to prevent XSS

Evolution of the OWASP Top 10



2003	2007	2010	2013	2017	2021
Unvalidated Input	XSS	Injection	Injection	Injection	Broken Access Control
Broken Access Control	Injection	XSS	Broken Auth. & Session Mngt	Broken Authentication	Cryptographic Failures
Broken Auth. & Session Mngt.	Malicious File Execution	Broken Auth. & Session Mngt	XSS	Sensitive Data Exposure	Injection
XSS	IDOR	IDOR	IDOR	XXE	Insecure Design
Buffer Overflows	CSRF	CSRF	Security Misconfiguration	Broken Access Control	Security Misconfiguration
Injection	Info Leakage & Improper Error Handling	Security Misconfiguration	Sensitive Data Exposure	Security Misconfiguration	Vulnerable & outdated components
Improper Error Handling	Broken Auth. & Session Mngt.	Insecure Cryptographic Storage	Missing function level access control	XSS	Identification & Authentication Failures
Insecure Storage	Insecure Cryptographic Storage	Failure to restrict URL access	CSRF	Insecure Deserialisation	Software & Data Integrity Failures
Denial of Service	Insecure Communication	Insecure Transport Layer	Components with known vulnerabilities	Components with known vulnerabilities	Insufficient Logging & Monitoring
Insecure Configuration Management	Failure to restrict URL access	Unvalidated Redirects and Forwards	Unvalidated Redirects and Forwards	Insufficient Logging & Monitoring	SSRF

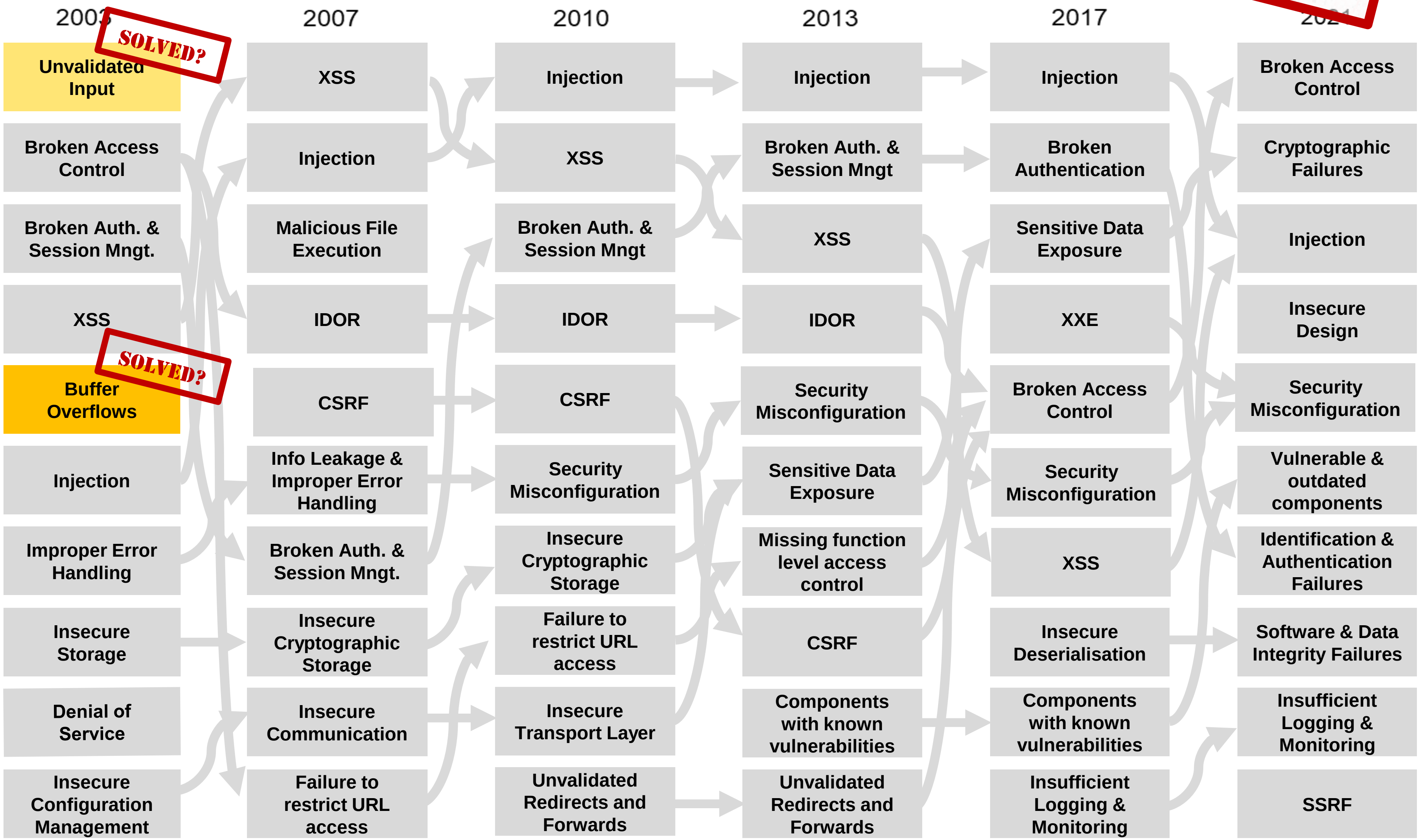
Evolution of the OWASP Top 10



2003	2007	2010	2013	2017	2021
Unvalidated Input	XSS	Injection	Injection	Injection	Broken Access Control
Broken Access Control	Injection	XSS	Broken Auth. & Session Mngt	Broken Authentication	Cryptographic Failures
Broken Auth. & Session Mngt.	Malicious File Execution	Broken Auth. & Session Mngt	XSS	Sensitive Data Exposure	Injection
XSS	IDOR	IDOR	IDOR	XXE	Insecure Design
Buffer Overflows	CSRF	CSRF	Security Misconfiguration	Broken Access Control	Security Misconfiguration
Injection	Info Leakage & Improper Error Handling	Security Misconfiguration	Sensitive Data Exposure	Security Misconfiguration	Vulnerable & outdated components
Improper Error Handling	Broken Auth. & Session Mngt.	Insecure Cryptographic Storage	Missing function level access control	XSS	Identification & Authentication Failures
Insecure Storage	Insecure Cryptographic Storage	Failure to restrict URL access	CSRF	Insecure Deserialisation	Software & Data Integrity Failures
Denial of Service	Insecure Communication	Insecure Transport Layer	Components with known vulnerabilities	Components with known vulnerabilities	Insufficient Logging & Monitoring
Insecure Configuration Management	Failure to restrict URL access	Unvalidated Redirects and Forwards	Unvalidated Redirects and Forwards	Insufficient Logging & Monitoring	SSRF

Evolution of the OWASP Top 10

SOLVED?



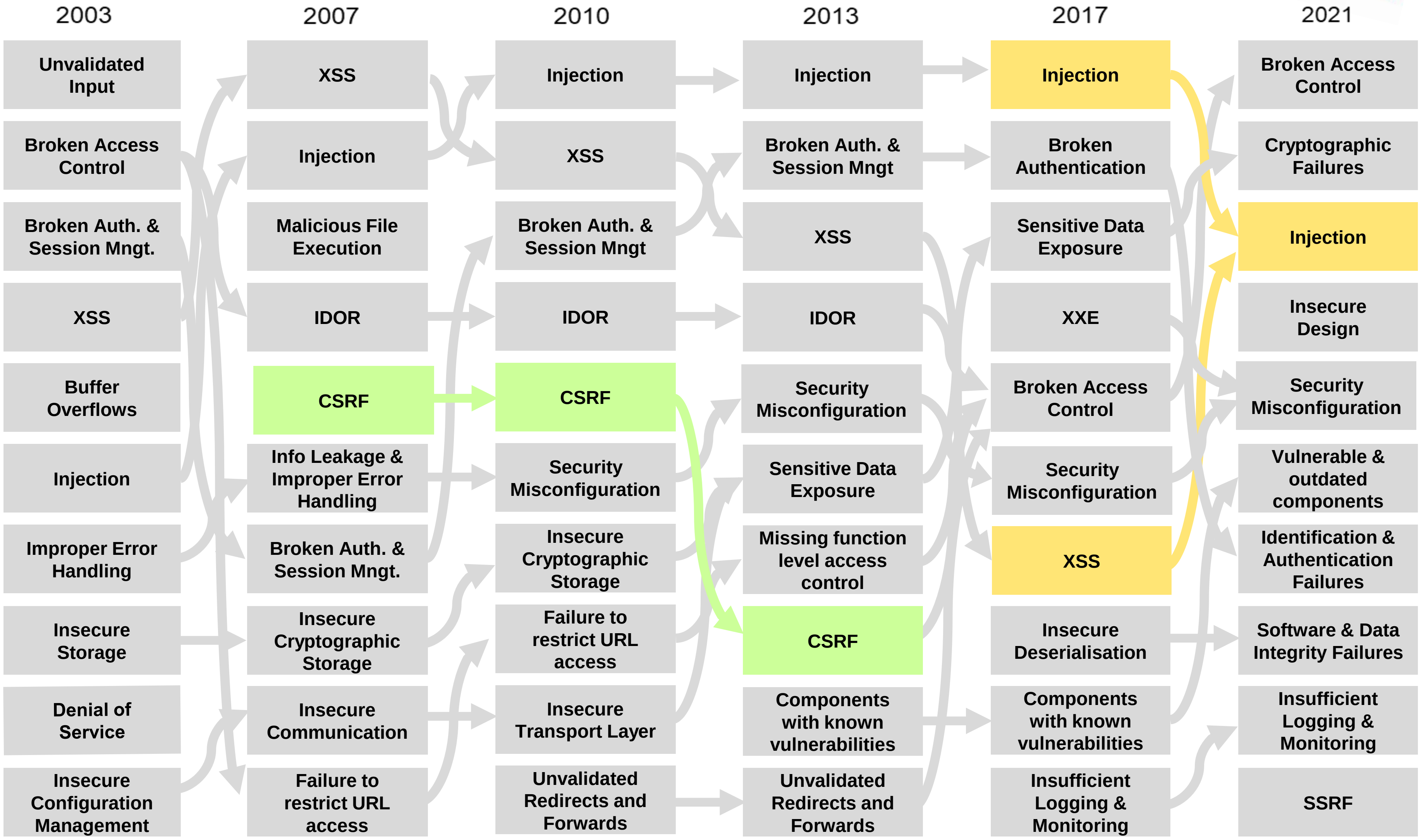
2003	2007	2010	2013	2017	2021
Unvalidated Input	XSS	Injection	Injection	Injection	Broken Access Control
Broken Access Control	Injection	XSS	Broken Auth. & Session Mngt	Broken Authentication	Cryptographic Failures
Broken Auth. & Session Mngt.	Malicious File Execution	Broken Auth. & Session Mngt	XSS	Sensitive Data Exposure	Injection
XSS	IDOR	IDOR	IDOR	XXE	Insecure Design
Buffer Overflows	CSRF	CSRF	Security Misconfiguration	Broken Access Control	Security Misconfiguration
Injection	Info Leakage & Improper Error Handling	Security Misconfiguration	Sensitive Data Exposure	Security Misconfiguration	Vulnerable & outdated components
Improper Error Handling	Broken Auth. & Session Mngt.	Insecure Cryptographic Storage	Missing function level access control	XSS	Identification & Authentication Failures
Insecure Storage	Insecure Cryptographic Storage	Failure to restrict URL access	CSRF	Insecure Deserialisation	Software & Data Integrity Failures
Denial of Service	Insecure Communication	Insecure Transport Layer	Components with known vulnerabilities	Components with known vulnerabilities	Insufficient Logging & Monitoring
Insecure Configuration Management	Failure to restrict URL access	Unvalidated Redirects and Forwards	Unvalidated Redirects and Forwards	Insufficient Logging & Monitoring	SSRF

SOLVED?

SOLVED?

Evolution of the OWASP Top 10

IMPROVED



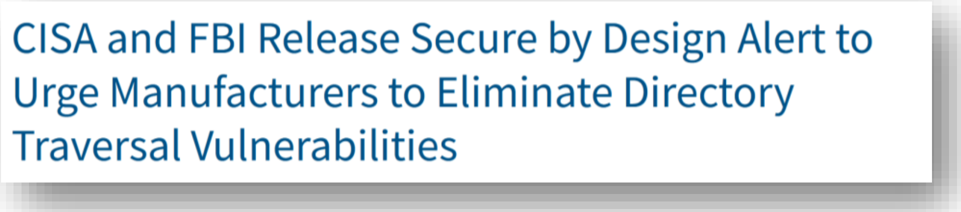
2003	2007	2010	2013	2017	2021
Unvalidated Input	XSS	Injection	Injection	Injection	Broken Access Control
Broken Access Control	Injection	XSS	Broken Auth. & Session Mngt	Broken Authentication	Cryptographic Failures
Broken Auth. & Session Mngt.	Malicious File Execution	Broken Auth. & Session Mngt	XSS	Sensitive Data Exposure	Injection
XSS	IDOR	IDOR	IDOR	XXE	Insecure Design
Buffer Overflows	CSRF	CSRF	Security Misconfiguration	Broken Access Control	Security Misconfiguration
Injection	Info Leakage & Improper Error Handling	Security Misconfiguration	Sensitive Data Exposure	Security Misconfiguration	Vulnerable & outdated components
Improper Error Handling	Broken Auth. & Session Mngt.	Insecure Cryptographic Storage	Missing function level access control	XSS	Identification & Authentication Failures
Insecure Storage	Insecure Cryptographic Storage	Failure to restrict URL access	CSRF	Insecure Deserialisation	Software & Data Integrity Failures
Denial of Service	Insecure Communication	Insecure Transport Layer	Components with known vulnerabilities	Components with known vulnerabilities	Insufficient Logging & Monitoring
Insecure Configuration Management	Failure to restrict URL access	Unvalidated Redirects and Forwards	Unvalidated Redirects and Forwards	Insufficient Logging & Monitoring	SSRF

Conclusions

We know how to make software *more* secure

just use one of the many secure development methodologies
and try to shift left

But: lots of ‘unforgivable bugs still common


CISA and FBI Release Secure by Design Alert to Urge Manufacturers to Eliminate Directory Traversal Vulnerabilities

Release Date: May 02, 2024

Tackling security is an ongoing process that will never be finished

In 2024, over 20 years after their initial software security initiative
Microsoft signed up to CISA’s Security-by-Design pledge

America’s Cyber Defense Agency
NATIONAL COORDINATOR FOR CRITICAL INFRASTRUCTURE SECURITY AND RESILIENCE
SECURE BY DESIGN
PLEDGE

Shifting down and shifting right

- The best way to shift left: shift *down*
  ie. address security lower in the technology stack API

  Eg. - memory-safe programming languages like Rust
  - safer APIs that are less injection-prone
  - session management frameworks that resists CSRF

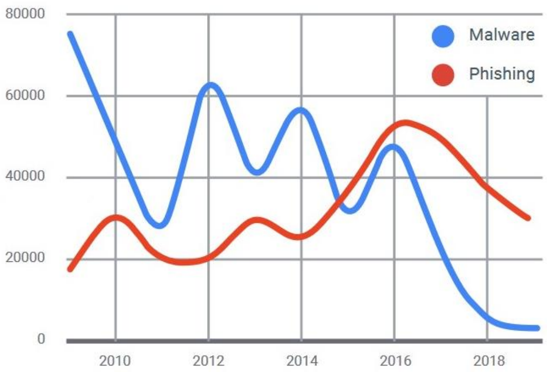
- But shifting *right* is also important
  ie. detect & react to security incidents

  Eg. having a SOC or deploying EDR

The ‘good’ news

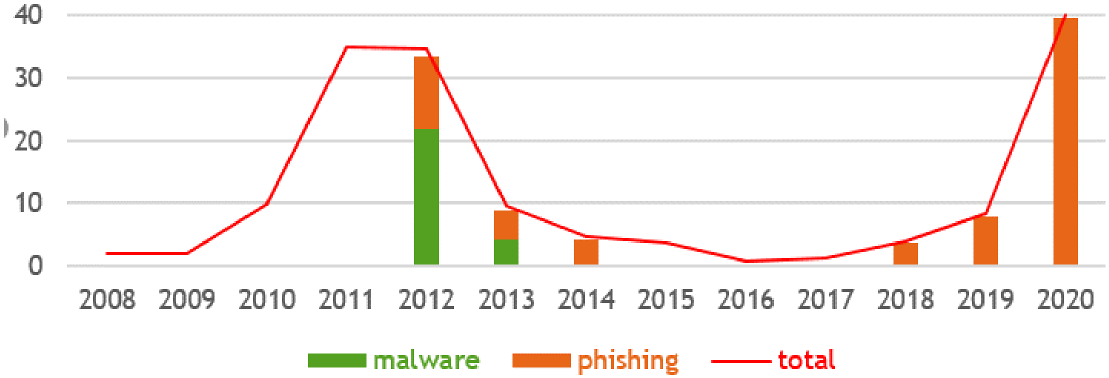
Software exploits no longer main root cause in some areas

- Exploit malware vs phishing sites detected by Google



[Source: Safe Browsing/
Google Transparency Report]

- Internet banking losses in the Netherlands



[Source: Betaalvereniging]