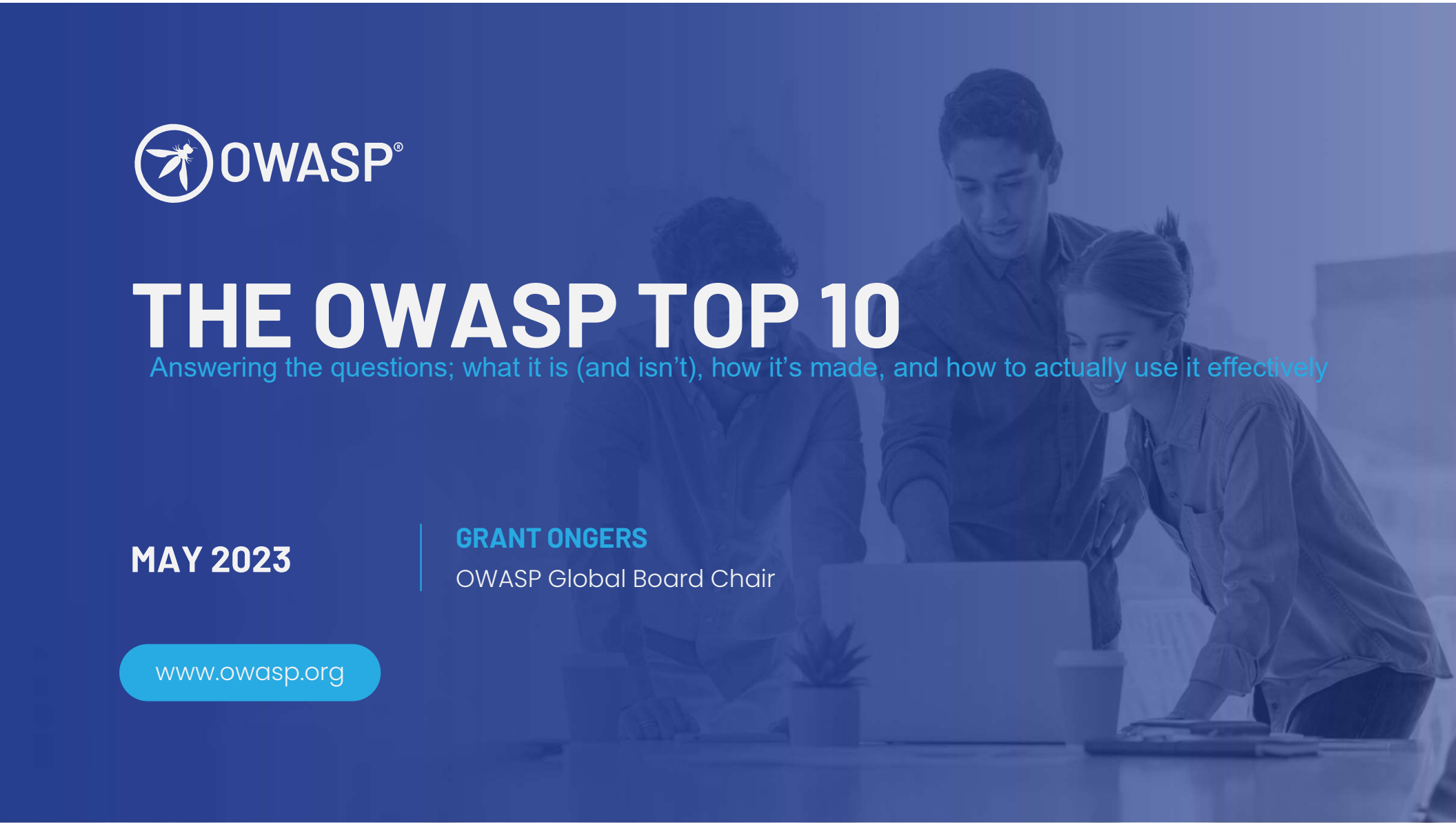


THE OWASP TOP 10

Answering the questions; what it is (and isn't), how it's made, and how to actually use it effectively

MAY 2023

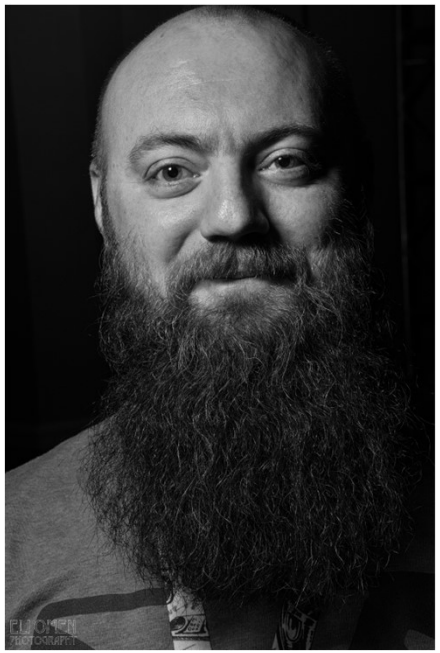
GRANT ONGERS

OWASP Global Board Chair

www.owasp.org

INTRODUCTION

This is all about the OWASP Top 10. We'll cover what the original OWASP Top 10 is, and we'll talk about what it isn't too! We'll touch on and explain a number of the projects the Top 10 spawned and how they differ from the original. We'll take a look at the process OWASP goes through with the creation of updated Top 10s and why that's important.



OWASP Global Chair | Co-Founder Secure Delivery

GRANT ONGERS

DEF CON Goon
BlackHat Staff (USA, EU)
BSides Staff (CPT, LND, LAS)
0xC0FFEE (CPT, LND)

Co-founder & CTO Secure Delivery

10+ in Dev (MSP, Telecoms, Banking);
20+ in Ops (EU Agencies, Utilities Providers); &
30+ in Sec (mostly white hat)

TWITTER / Slack / LinkedIn
@rewtd

E-MAIL
grant.ongers@owasp.org
grant@securedelivery.io

SECURE DELIVERY

TAILORED APPLICATION SOFTWARE

We believe that security cannot and should not exist in parallel to engineering teams but needs to be integrated. Only then organisations will experience the synergies and economies of scale in their product lifecycle.

OWASP

Open Worldwide Application Security Project

Community driven:

200+ chapters in 50 countries

https://www.meetup.com/owasp

OWASP German Chapter

GOD (German OWASP Day): 30 - 31 May 2023 (https://god.owasp.de/)

GERMAN OWASP DAY 2023

Update (2023-01-26): Der Call for Sponsors für den German OWASP Day 2023 ist eröffnet! Alle relevanten Informationen finden sich in unserer Broschüre.
Update (2023-02-05): Der Call for Presentations ist nun ebenfalls online!
Update (2023-02-14): **Der Verkauf der Konferenztickets ist gestartet!**

Das German Chapter des Open Web Application Security Project (OWASP) richtet jährlich ihre nationale OWASP-Konferenz aus. Wir freuen uns ankündigen zu dürfen, dass nach einer mehrjährigen Pause der German OWASP Day 2023 wieder stattfinden wird!

Dieser wird am **30. und 31.05.2023** in der **Frankfurt School of Finance and Management** in **Frankfurt am Main** stattfinden.

Am Hauptveranstaltungstag (31.05) finden eine Reihe von von spannenden technischen und nicht-technischen Vorträgen im Bereich der Anwendungssicherheit statt. Am Vortag (30.05) werden verschiedene Seminare und eine Abendveranstaltung zum gemeinsamen Erfahrungsaustausch angeboten.

Der German OWASP Day ist die führende unabhängige und nicht-kommerzielle Konferenz in Deutschland zur Sicherheit von Anwendungen. Teilnahme ist ein Muss für Sicherheitsexperten, Entwickler, IT-Sicherheitsmanager und sonstige IT-Professionals im Bereich der Anwendungssicherheit.

Der Call for Presentation läuft, für weitere Infos siehe CALL FOR PRESENTATION!

Schreibt euch auf die Mailingliste des German Chapters ein, um informiert zu werden, sobald es Neuigkeiten zur Planung gibt.

OWASP German Chapter

WHAT IT IS

OWASP Top 10

> “The OWASP Top 10 is primarily an awareness document...”

ANDREW VAN DER STOCK

https://owasp.org/Top10/A00_2021_How_to_use_the_OWASP_Top_10_as_a_standard/

TOP 10: WHAT IT IS

TOP 10 VULNERABILITIES

A01:2021-Broken Access Control

A02:2021-Cryptographic Failures

A03:2021-Injection

A04:2021-Insecure Design

A05:2021-Security Misconfiguration

A06:2021-Vulnerable and Outdated Components

A07:2021-Identification and Authentication Failures

A08:2021-Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures

A10:2021-Server Side Request Forgery

A01:2021-BROKEN ACCESS CONTROL

Restrictions on what authenticated users are allowed to do are often not properly enforced.

Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data or change access rights.

Prevention:

- Deny by default
- Centralise your access control mechanisms and re-use them throughout your app
- Enforce access control down to record ownership, not just CRUD permission for a type of record
- Log and alert on access control failures
- Test!

```
// This should be a strict equals.
if (route.data.roles && route.data.roles.indexOf(currentUser.role) !== -1)
    // role not authorised so redirect to home page
    this.router.navigate(['/']);
    return false;
}
// Also doesn't deny by default
return true;
```

A02:2021 – CRYPTOGRAPHIC FAILURES

Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII.

Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.

Prevention:

- Properly hash and salt stored passwords, or federate auth to an identity provider
- Encrypt sensitive data at rest and in transit and manage cleartext data at application level carefully
- Don't store sensitive data unnecessarily; discard it as soon as possible
- Never cache, or allow caching, of responses that contain sensitive data

```
// An extremely weak, really terrible password hashing function
MessageDigest md = MessageDigest.getInstance("MD5");
md.update(password.getBytes());
byte[] digest = md.digest();
String encryptedPassword = DatatypeConverter.printHexBinary(digest).toUpperCase();
```

https://owasp.org/Top10/A02_2021-Cryptographic_Failures/

TOP 10: WHAT IT IS

A03:2021 – INJECTION

Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query.

The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

Prevention:

- Use parameterised queries or an ORM
- "Allow list" server-side validation of input
- Escape special characters
- **Use built-ins**, don't roll your own
- If you're only expecting one result, be very surprised if you see more. Use LIMIT or similar to prevent mass disclosure of records

```
// SQL Injection
String query = "SELECT * FROM accounts WHERE custID='" + request.getParameter("id") + "'";

# Command Injection
os.system(socket.recv(1024))
```

https://owasp.org/Top10/A03_2021-Injection/

A04:2021 – INSECURE DESIGN

Insecure design is a broad category representing different weaknesses, expressed as “missing or ineffective control design.”

A secure design can still have implementation defects leading to vulnerabilities that may be exploited.

An insecure design cannot be fixed by a perfect implementation as by definition, needed security controls were never created to defend against specific attacks.

Prevention:

- Establish and use a library of secure design patterns or paved road ready to use components
- Use threat modeling for critical authentication, access control, business logic, and key flows
- Integrate security language and controls into user stories
- Compile use-cases and misuse-cases for each tier of your application.

```
// There isn't really a code example for this. Have a Dachshund.
//
//                        __
//   ,                   ," e`--o
//   ((                  (  | __,'
//   \\~----------------' \_;/
//   (                     /
//   /) .________________.  )
//   (( (                ( (
//   ``-'                ``-'
```

A05:2021 – SECURITY MISCONFIGURATION

Commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information.

Operating systems, frameworks, libraries, and applications must be securely configured, and patched/upgraded promptly.

Prevention:

- Automated, repeatable environment configuration
- Don't overcomplicate - your configuration templating doesn't need to be Turing-complete
- Peer review and test configurations as well as code
- Remove everything you don't need to run - components, documents, sample apps, default accounts

```
if (env.IsDevelopment()) {
    app.UseDeveloperExceptionPage();
} else {
    // TODO: Remove this
    app.UseDeveloperExceptionPage();
    //app.UseExceptionHandler("/Error");
    //app.UseHsts();
}
```

https://owasp.org/Top10/A05_2021-Security_Misconfiguration/

A06:2021 – VULN. & OUTDATED COMPONENTS

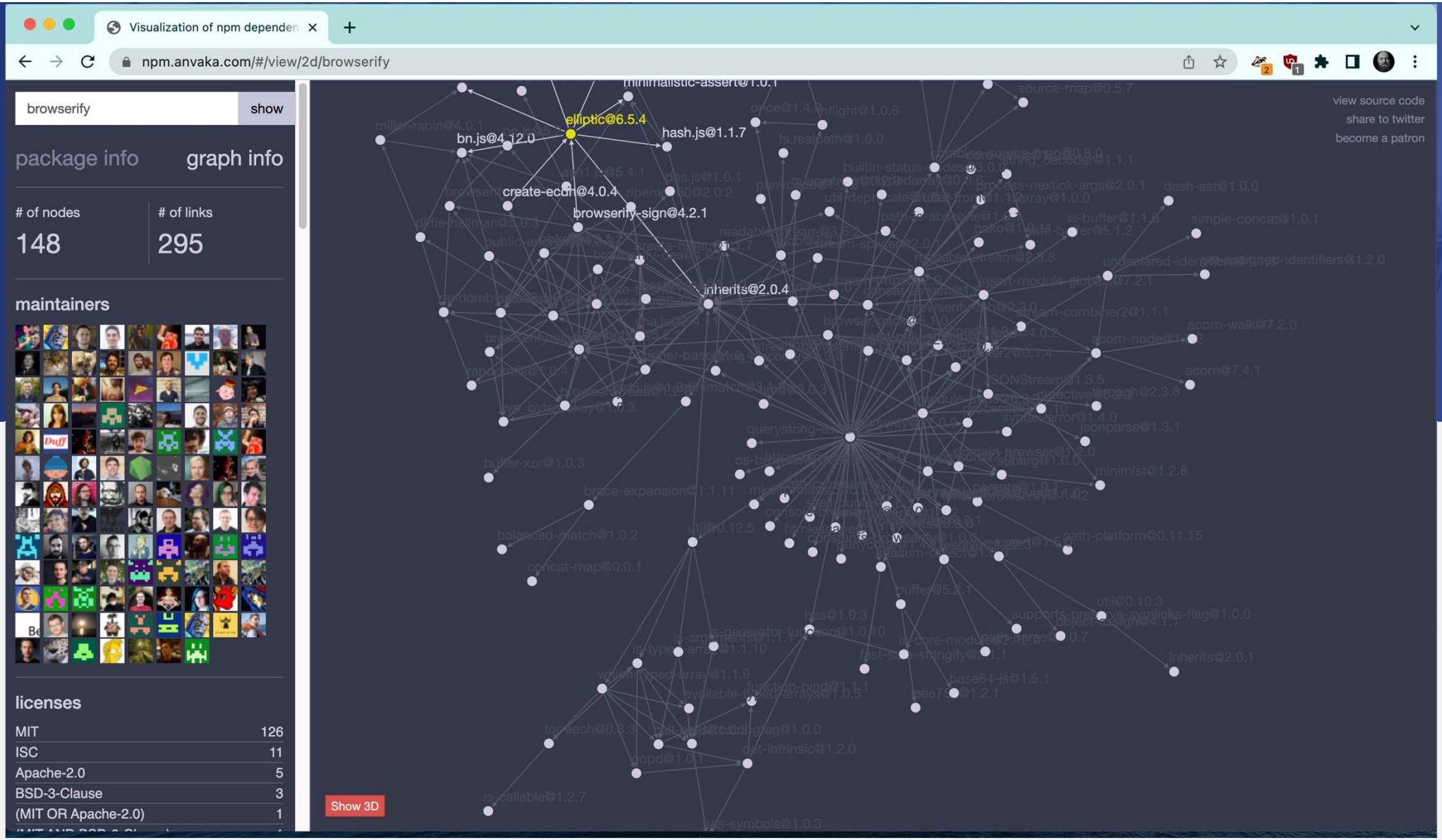
Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover.

Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.
Example 1: HackerNoon
Example 2: Browserify / Left-Pad (kik)

Prevention:

- Properly hash and salt stored passwords, or federate auth to an identity provider
- Encrypt sensitive data at rest and in transit and manage cleartext data at application level carefully
- Don't store sensitive data unnecessarily; discard it as soon as possible
- Never cache, or allow caching, of responses that contain sensitive data

```
// There isn't really a code example for this. Have another Dachshund.
//                      __
//   ,                 ," e`--o
//   ((                (  | __,'
//   \\~---------------' \_;/
//   (                   /
//   /) .________________.  )
//   (( (               (( (
//   ``-'               ``-'
```


Visualization of npm dependen
npm.anvaka.com/#/view/2d/browserify
browserify
show
package info
graph info
# of nodes
148
# of links
295
maintainers
licenses
MIT 126
ISC 11
Apache-2.0 5
BSD-3-Clause 3
(MIT OR Apache-2.0) 1
view source code
share to twitter
become a patron
elliptic@6.5.4
bn.js@4.12.0
hash.js@1.1.7
create-ecdh@4.0.4
browserify-sign@4.2.1
inherits@2.0.4
Show 3D

CVE-2022-37623 - CVE.report

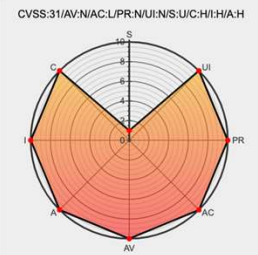
cve.report/CVE-2022-37623

Home

CVE.report Search

CVE-2022-37623

Source: Mitre | Source: NIST | CVE.ORG | Print: PDF



Certain versions of Browserify-shim from Browserify-shim Project contain the following vulnerability:

Prototype pollution vulnerability in function resolveShims in resolve-shims.js in thlorenz browserify-shim 3.8.15 via the shimPath variable in resolve-shims.js.

CVE-2022-37623 has been assigned by cve@mitre.org to track the vulnerability - currently rated as CRITICAL severity.

CVSS3 Score: 9.8 - CRITICAL

Attack Vector	Attack Complexity	Privileges Required	User Interaction
NETWORK	LOW	NONE	NONE
Scope	**Confidentiality Impact**	**Integrity Impact**	**Availability Impact**
UNCHANGED	HIGH	HIGH	HIGH

CVE References

Description	Tags	Link

A07:2021 – ID & AUTHENTICATION FAILURES

Application functions related to authentication and session management are often implemented incorrectly.

This allows attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.

Prevention:

- Strong passwords & MFA. Use the updated NIST guidance, it's good now!
- Block or heavily rate limit repeated authentication attempts
- Expire inactive sessions
- Never implicitly trust

```
// Very silly session IDs
https://example.com/userApp?sessionId=14632
https://example.com/userApp?sessionId=14633
https://example.com/userApp?sessionId=14634

// Default credentials
if (userName == 'admin' && password == 'letmein') { }
```

A08:2021 – SOFT. & DATA INTEGRITY FAILURES

Insecure deserialization often leads to remote code execution.

Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.

Prevention:

- Avoid native serialisation formats, use a pure data format like JSON
- If you have to, consider cryptographically signing and only deserialising authenticated messages
- Don’t allow the datastream to define the type of object being deserialised to
- Don’t write your own deserialiser - use a common, secured, actively maintained one

```
import pickle

data = """ data from an untrusted source """
pickle.loads(data)
```

A09:2021 – SEC. LOG- & MONITORING FAILURES

Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data.

Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

Prevention:

- Log all authentication, authorisation and validation failures
- Alert on serious events or combination of events - be aware of Alert Fatigue
- Ship logs to places where attackers cannot easily access and secure them against erasure
- Use a smart log management or Security information and event management (SIEM) tool

A10:2021 – SERVER-SIDE REQUEST FORGERY

SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL.

It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network access control list (ACL).

Prevention:

- Sanitize and validate all client-supplied input data
- Enforce the URL schema, port, and destination with a positive allow list
- Do not send raw responses to clients
- Disable HTTP redirections
- Be aware of the URL consistency to avoid attacks such as DNS rebinding and TOCTOU race conditions

```
POST /product/stock HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 118

stockApi=http://stock.example.net:8080/product/stock/check%3FproductId%3D6%26storeId%3D1
```

OWASP®

HOW IT IS MADE

OWASP Top 10

FROM 2003 to 2004

	2003	2004
A1	Unvalidated Parameters	Unvalidated Input (renamed from A1:2003)
A2	Missing Functional Level Access Control	Broken Access Control (renamed from A2:2003)
A3	Broken Authentication and Session Management	Broken Authentication and Session Management
A4	Cross Site Scripting (XSS)	Cross Site Scripting (XSS)
A5	Buffer Overflows	Buffer Overflows
A6	Command Injection Flaws	Injection Flaws (renamed from A6:2003)
A7	Improper Error Handling	Improper Error Handling
A8	Web and Application Server	Insecure Use of Cryptography (renamed from A8:2003)
A9	Remote Administration Flaws (dropped off)	Denial of Service (split from A2:2003)
A10	Security Misconfiguration (dropped off)	Insecure Use of Cryptography (new)

TOP 10: HOW IT IS MADE - BRIEF HISTORY

FROM 2004 to 2007

	2004	2007
A1	Unvalidated Input (dropped off)	Cross Site Scripting (XSS)
A2	Broken Access Control	Injection Flaws
A3	Broken Authentication and Session Management	Malicious File Execution (new)
A4	Cross Site Scripting (XSS)	Insecure Direct Object Reference (split from A2:2004)
A5	Buffer Overflows (dropped off)	Cross Site Request Forgery (CSRF) (new)
A6	Injection Flaws	Information Leakage & Improper Error Handling (renamed from A7:2004)
A7	Improper Error Handling	Broken Authentication and Session Management
A8	Insecure Use of Cryptography	Insecure Cryptographic Storage (renamed from A8:2004)
A9	Denial of Service (dropped off)	Insecure Communications (renamed from A10-2004)

TOP 10: HOW IT IS MADE - BRIEF HISTORY

FROM 2007 to 2010

	2007	2010
A1	Cross Site Scripting (XSS)	Injection (renamed from A2:2007)
A2	Injection Flaws	Cross Site Scripting (XSS)
A3	Malicious File Execution (dropped off)	Broken Authentication and Session Management
A4	Insecure Direct Object Reference	Insecure Direct Object Reference
A5	Cross Site Request Forgery (CSRF)	Cross Site Request Forgery (CSRF)
A6	Information Leakage & Improper Error Handling (dropped off)	Security Misconfiguration (returned from A10:2004)
A7	Broken Authentication and Session Management	Insecure Cryptographic Storage
A8	Insecure Cryptographic Storage	Failure to Restrict URL Access
A9	Insecure Communications	Insufficient Transport Layer Protection (renamed from A9:2007
A10	Failure to Restrict URL Access	Unvalidated Redirects and Forwards (new)

FROM 2010 to 2013

	2010	2013
A1	Injection	Injection
A2	Cross Site Scripting (XSS)	Broken Authentication and Session Management
A3	Broken Authentication and Session Management	Cross Site Scripting (XSS)
A4	Insecure Direct Object Reference	Insecure Direct Object Reference
A5	Cross Site Request Forgery (CSRF)	Security Misconfiguration
A6	Security Misconfiguration	Sensitive Data Exposure (merging A7:2010 and A9:2010)
A7	Insecure Cryptographic Storage	Missing Functional Level Access Control (renamed from A8:2010)
A8	Failure to Restrict URL Access	Cross Site Request Forgery (CSRF)
A9	Insufficient Transport Layer Protection	Using Known Vulnerable Components (new)
A10	Unvalidated Redirects and Forwards	Unvalidated Redirects and Forwards

TOP 10: HOW IT IS MADE - BRIEF HISTORY

FROM 2013 to 2017

	2013	2017
A1	Injection	Injection
A2	Broken Authentication and Session Management	Broken Authentication (renamed from A2:2013)
A3	Cross Site Scripting (XSS)	Sensitive Data Exposure
A4	Insecure Direct Object Reference	XML External Entities (XXE) (new)
A5	Security Misconfiguration	Broken Access Control (merged A4:2013 and A7:2013 / returned A2:2004)
A6	Sensitive Data Exposure	Security Misconfiguration
A7	Missing Functional Level Access Control	Cross-Site Scripting (XSS)
A8	Cross Site Request Forgery (CSRF) (dropped off)	Insecure Deserialization (new)
A9	Using Known Vulnerable Components	Using Known Vulnerable Components
A10	Unvalidated Redirects and Forwards (dropped off)	Insufficient Logging & Monitoring (new)

TOP 10: HOW IT IS MADE - BRIEF HISTORY

FROM 2017 to 2021

	2017	2021
A1	Injection	Broken Access Control
A2	Broken Authentication	Cryptographic Failure (renamed from A3:2017)
A3	Sensitive Data Exposure	Injection
A4	XML External Entities (XXE) (dropped off)	Insecure Design (new)
A5	Broken Access Control	Security Misconfiguration
A6	Security Misconfiguration	Vulnerable & Outdated Components (renamed from A9:2017)
A7	Cross-Site Scripting (XSS) (dropped off)	Identification & Authentication Failures (renamed from A2:2017)
A8	Insecure Deserialization (dropped off)	Software and Data Integrity Failures (new)
A9	Using Known Vulnerable Components	Security Logging and Monitoring Failures (renamed from A10:2017)
A10	Insufficient Logging & Monitoring	Server-Side Request Forgery (new)

CORE PRINCIPLES

- OWASP Top 10 is a baseline, not a ceiling
- Data is good, data isn't everything
- Data is looking in the past, hence the community survey
- Stability is good
- Need to raise the minimum bar
- Drive the right behaviour to improve software security
- Focus on root cause over symptom

DATA ANALYSIS

DETERMINE CATEGORIES

Not just the 30 (or so) CWEs previously - an open data request was put out pulling about 400 CWEs

WHAT'S A CWE AGAIN?

Common Weakness Enumeration - the MITRE published, community-developed list of software and hardware weakness types.

COMMUNITY

DATA IS RETROSPECTIVE

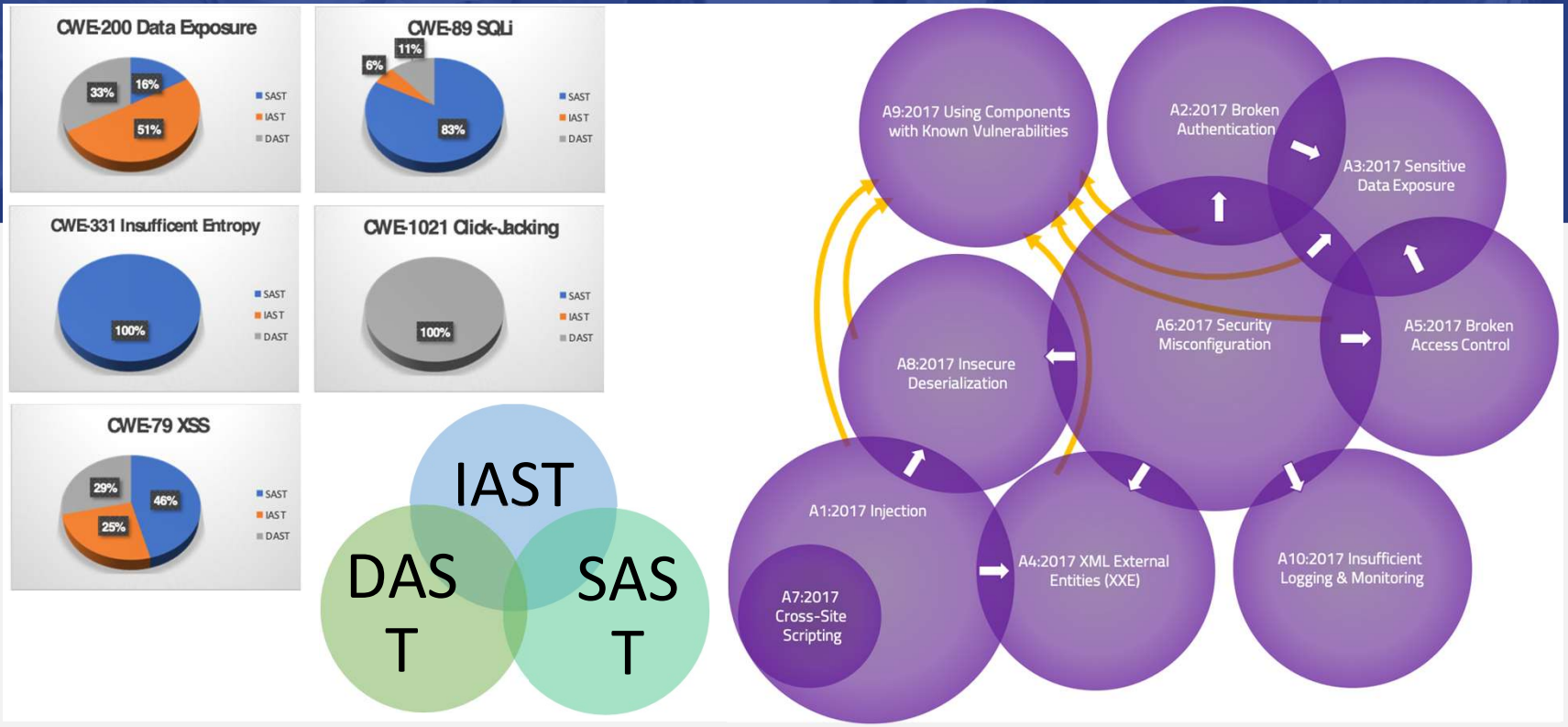
Using just data means we can only see the past - and the Top 10 needs to be more forward focused

OWASP IS THE COMMUNITY

The best, and brightest working in AppSec are involved

TOP 10: HOW IT IS MADE

DATA OVERLAP 2017 / 2021

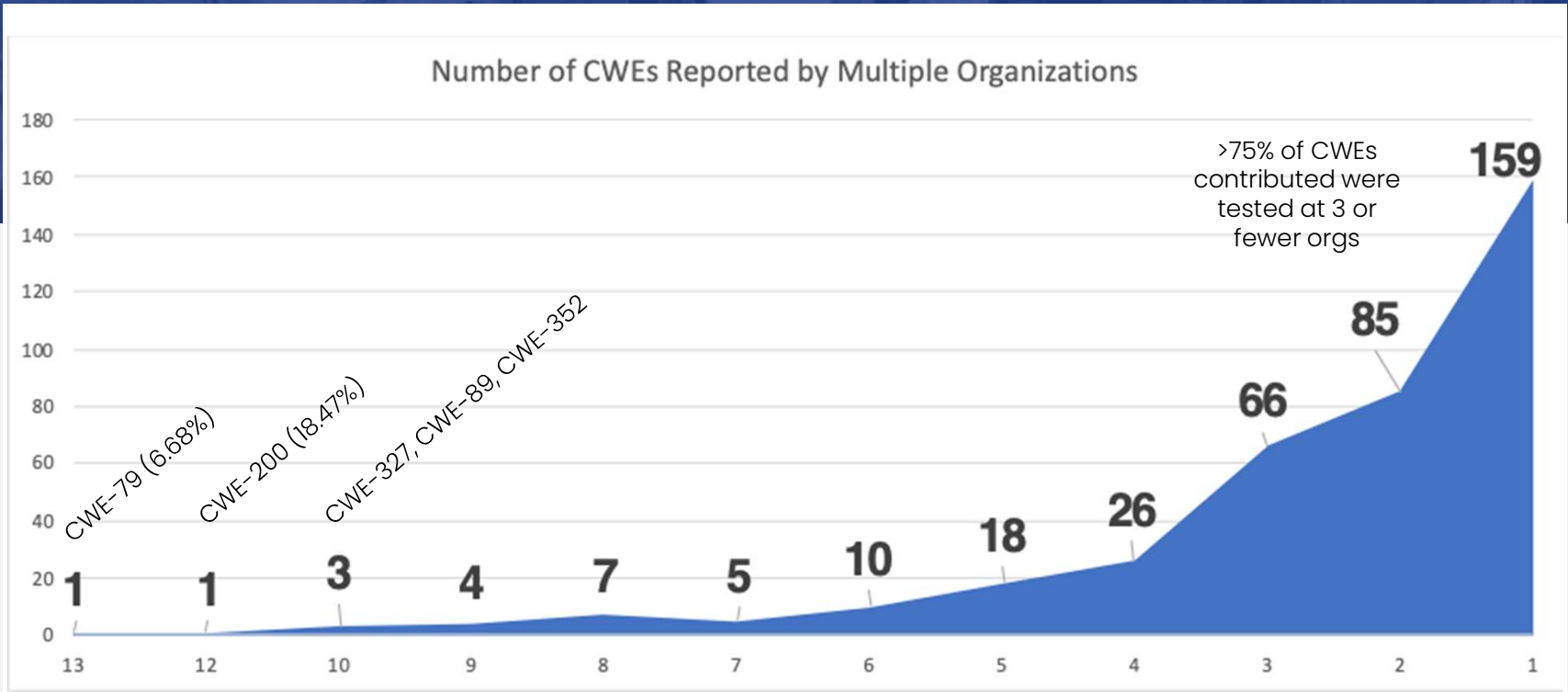


TOP 10: HOW IT IS MADE

DATA PROVIDERS

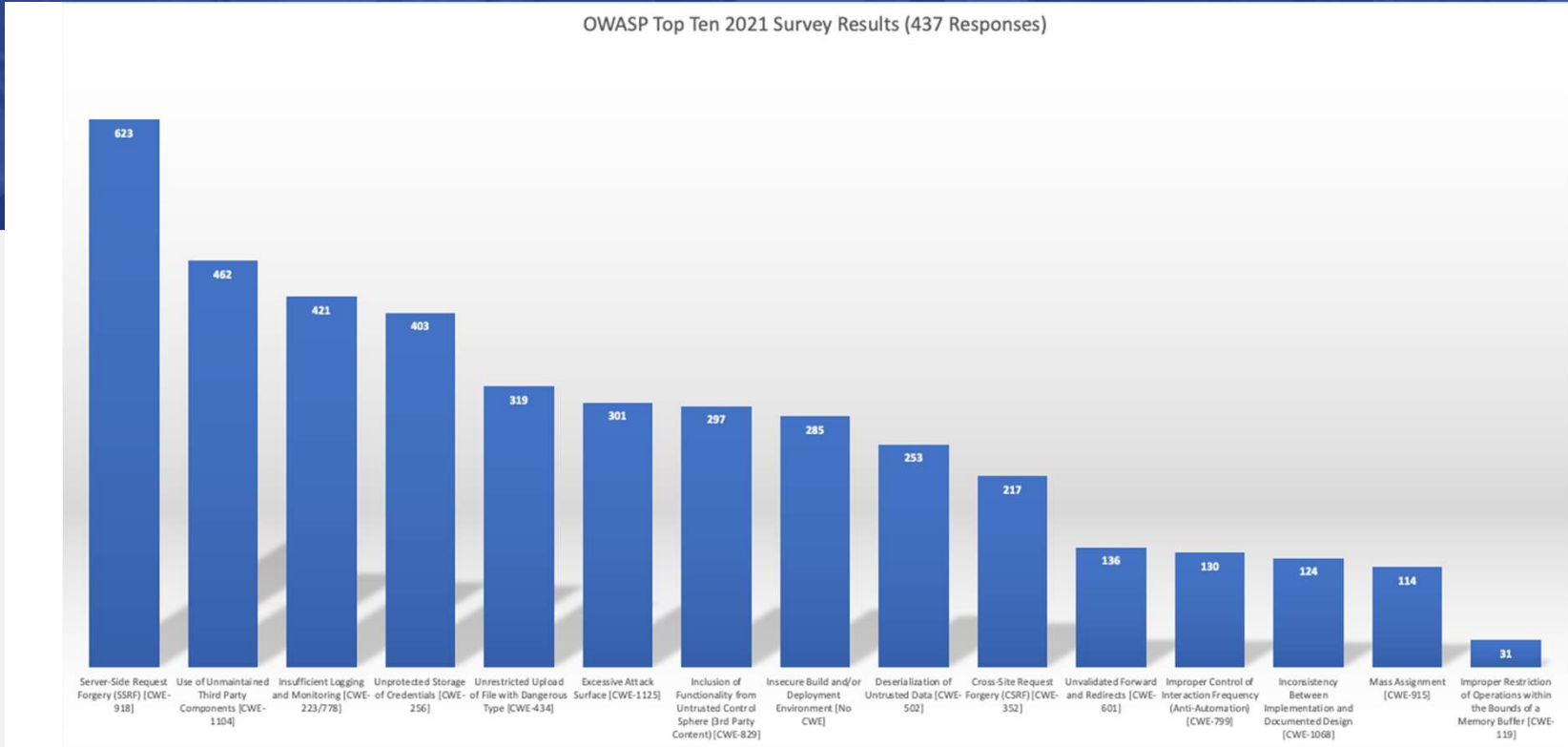
TOP 10: HOW IT IS MADE

DATA BY CWE



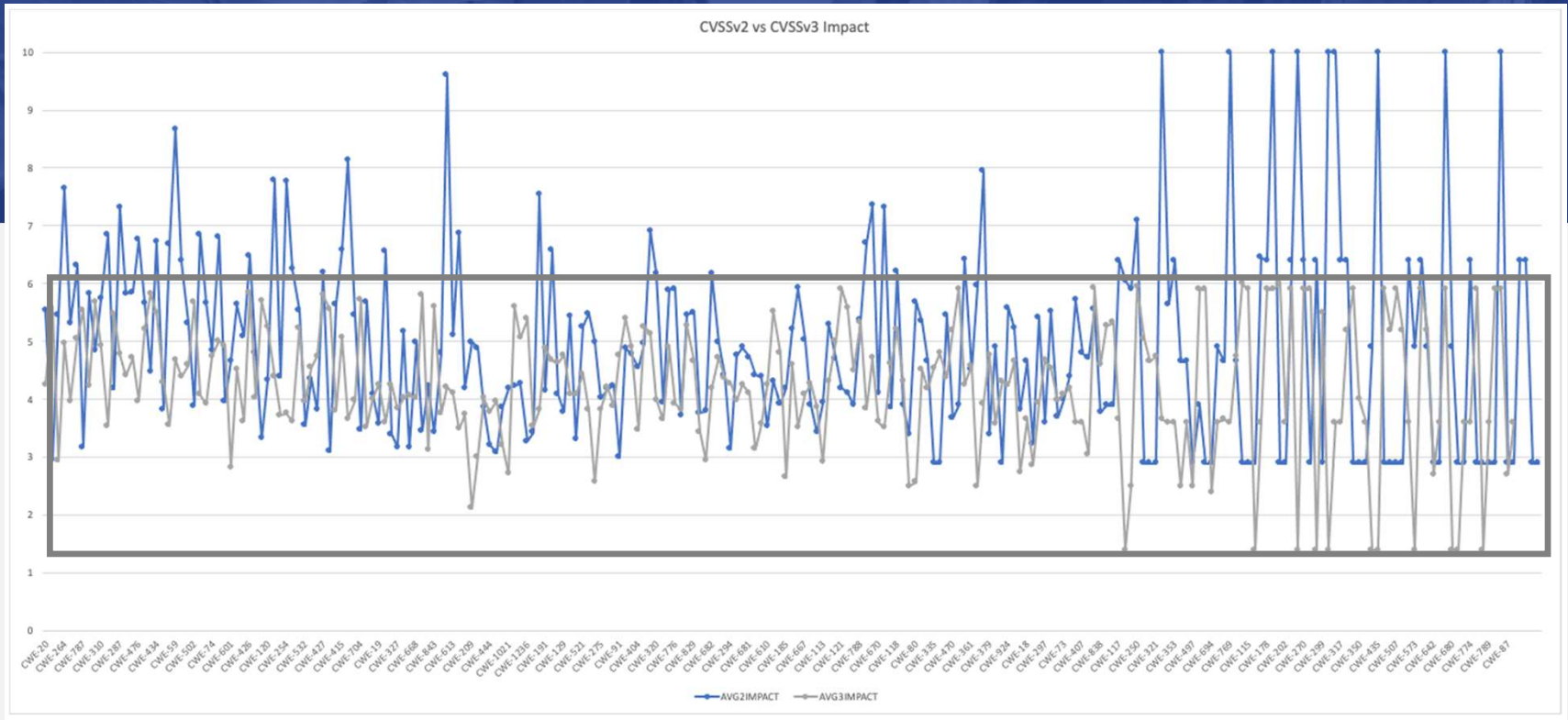
TOP 10: HOW IT IS MADE

SURVEY RESULTS



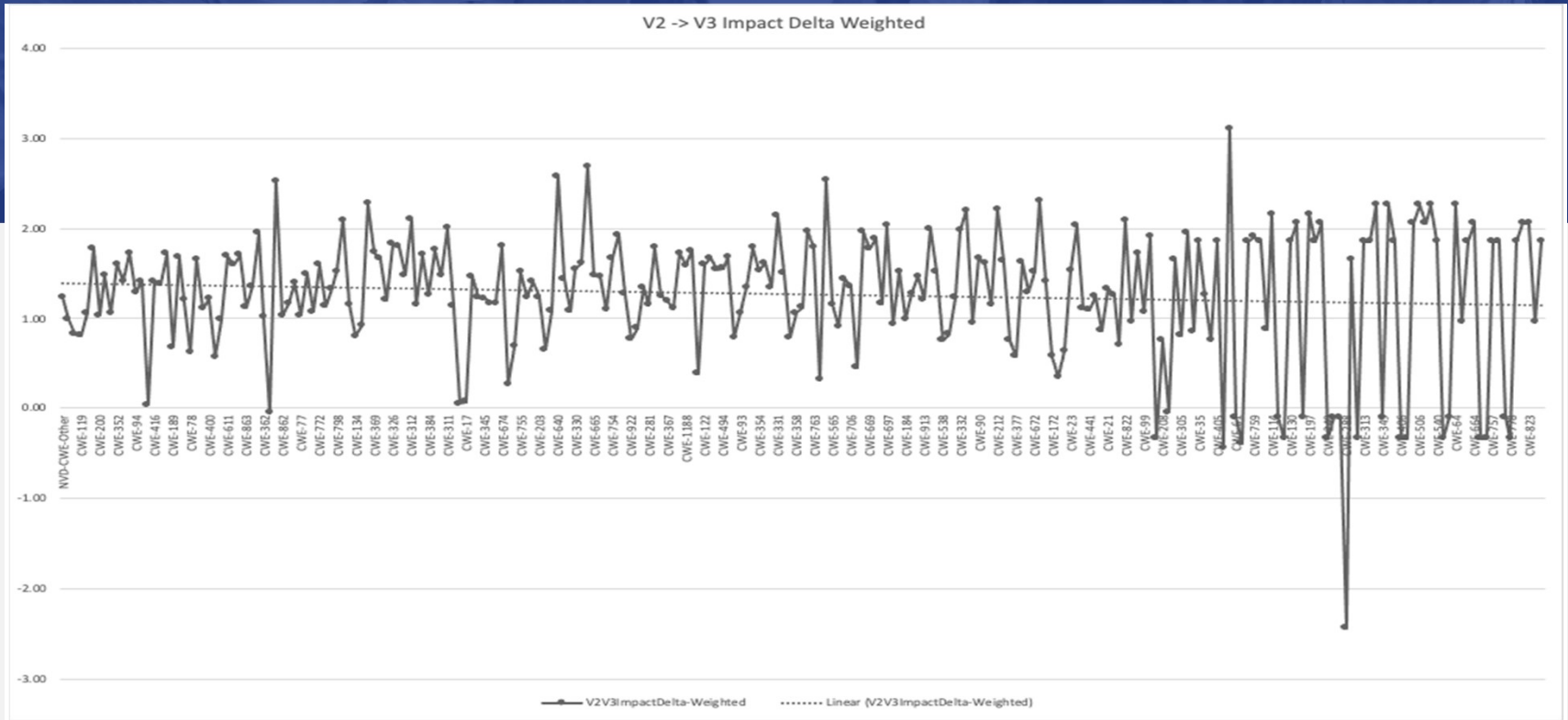
TOP 10: HOW IT IS MADE

CVSS SCORING



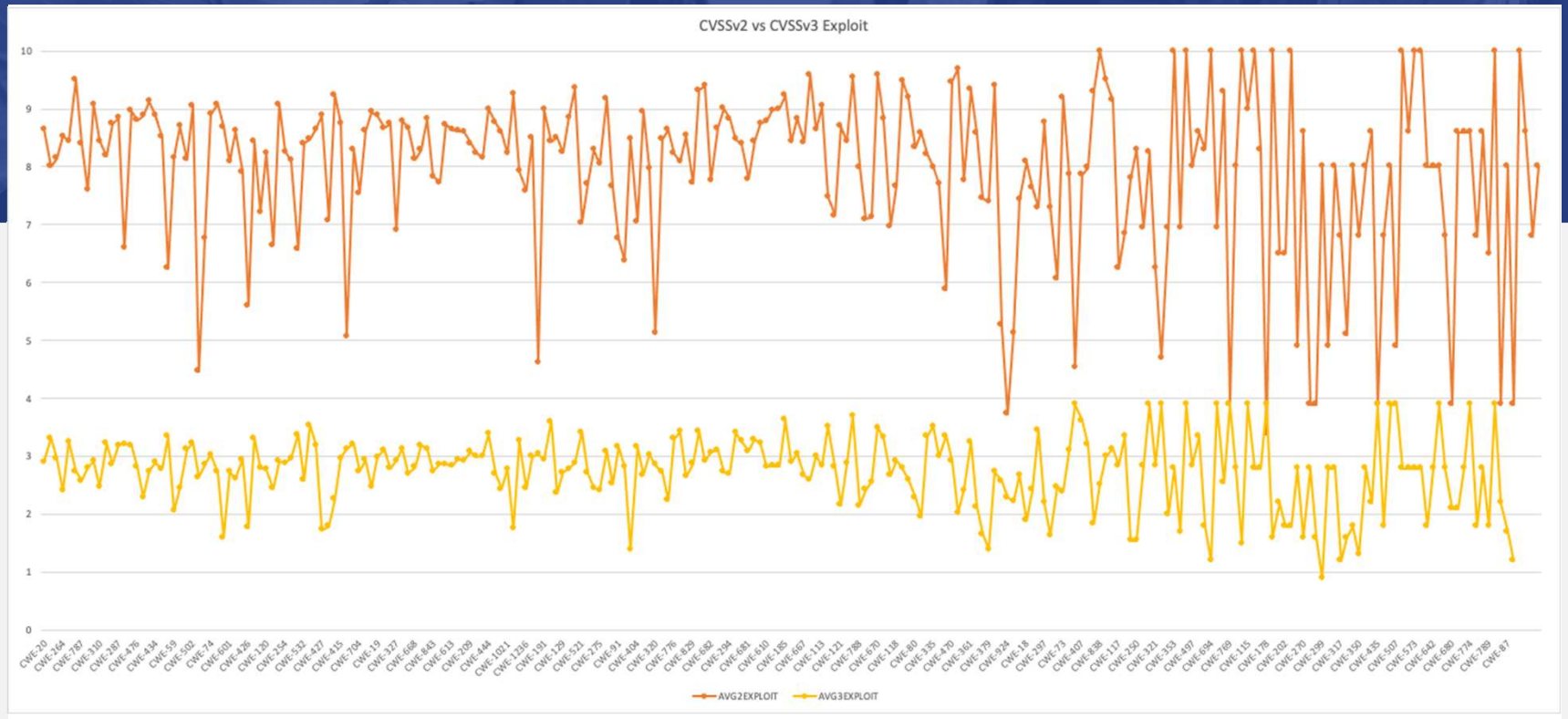
TOP 10: HOW IT IS MADE

CVSS SCORING



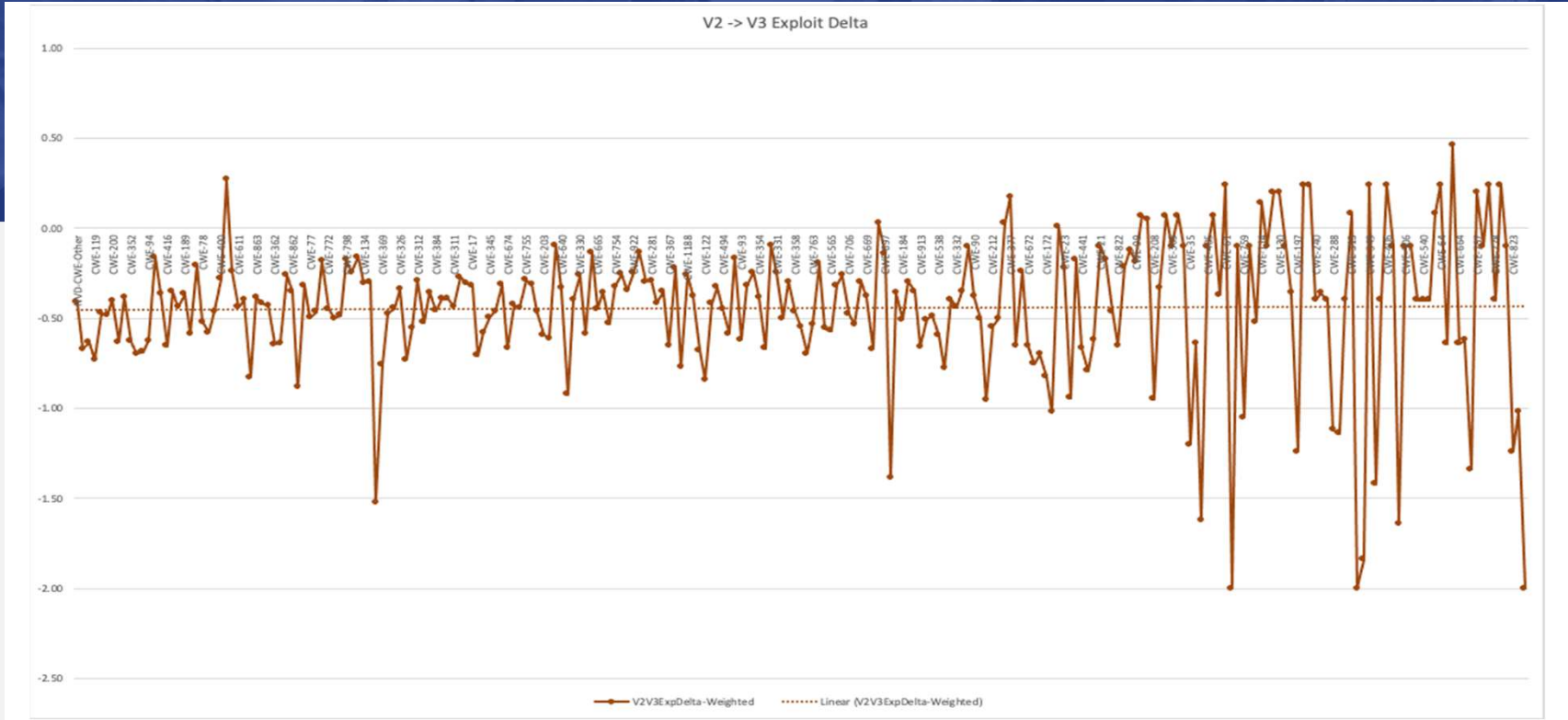
TOP 10: HOW IT IS MADE

CVSS SCORING



TOP 10: HOW IT IS MADE

CVSS SCORING



TOP 10: HOW IT IS MADE

(NOT SO) SECRET FORMULA

Category	Incidence	Coverage	Exploit	Impact	Occurances	Score	Rank
Vulnerable Components	83.88	31.07	7.5	15.0	3.05	140.49	6
Cryptographic Failures	139.31	47.60	10.9	20.4	23.38	241.64	2
Security Misconfiguration	59.51	53.75	12.2	19.7	20.84	165.97	5
Identification or Authentication Failures	44.52	47.70	11.1	19.5	13.22	136.03	7
Software or Data Integrity Failures	50.00	45.03	10.4	23.8	4.80	134.04	8
Insecure Design	72.57	46.35	9.7	20.3	26.24	175.21	3
Injection	57.27	56.43	10.9	21.5	27.42	173.45	4
Broken Access Control	167.92	56.73	10.4	17.8	31.85	284.67	1
Auditing/Logging Failures	57.68	32.20	10.3	15.0	5.36	120.52	9
SSRF	8.17	40.63	12.4	20.2	0.95	82.35	10
Weight	300	60	1.5	3	10000		

Category	Incidence	Coverage	Exploit	Impact	Occurances
Vulnerable Components	60%	22%	5%	11%	2%
Cryptographic Failures	58%	20%	5%	8%	10%
Security Misconfiguration	36%	32%	7%	12%	13%
Identification or Authentication Failures	33%	35%	8%	14%	10%
Software or Data Integrity Failures	37%	34%	8%	18%	4%
Insecure Design	41%	26%	6%	12%	15%
Injection	33%	33%	6%	12%	16%
Broken Access Control	59%	20%	4%	6%	11%
Auditing/Logging Failures	48%	27%	9%	12%	4%
SSRF	10%	49%	15%	24%	1%

https://github.com/OWASP/Top10/tree/master/2021

WHAT IT ISN'T

OWASP Top 10

https://owasp.org/Top10/A00_2021_How_to_use_the_OWASP_Top_10_as_a_standard/

 |

> “If you want to use the OWASP Top 10 as a coding or testing standard, know that it is the bare minimum and just a starting point.”
>
> ANDREW VAN DER STOCK

https://owasp.org/Top10/A00_2021_How_to_use_the_OWASP_Top_10_as_a_standard/

https://owasp.org/Top10/A00_2021_How_to_use_the_OWASP_Top_10_as_a_standard/

HOW TO USE IT

OWASP Top 10

"The ASVS is the only acceptable choice for tool vendors."

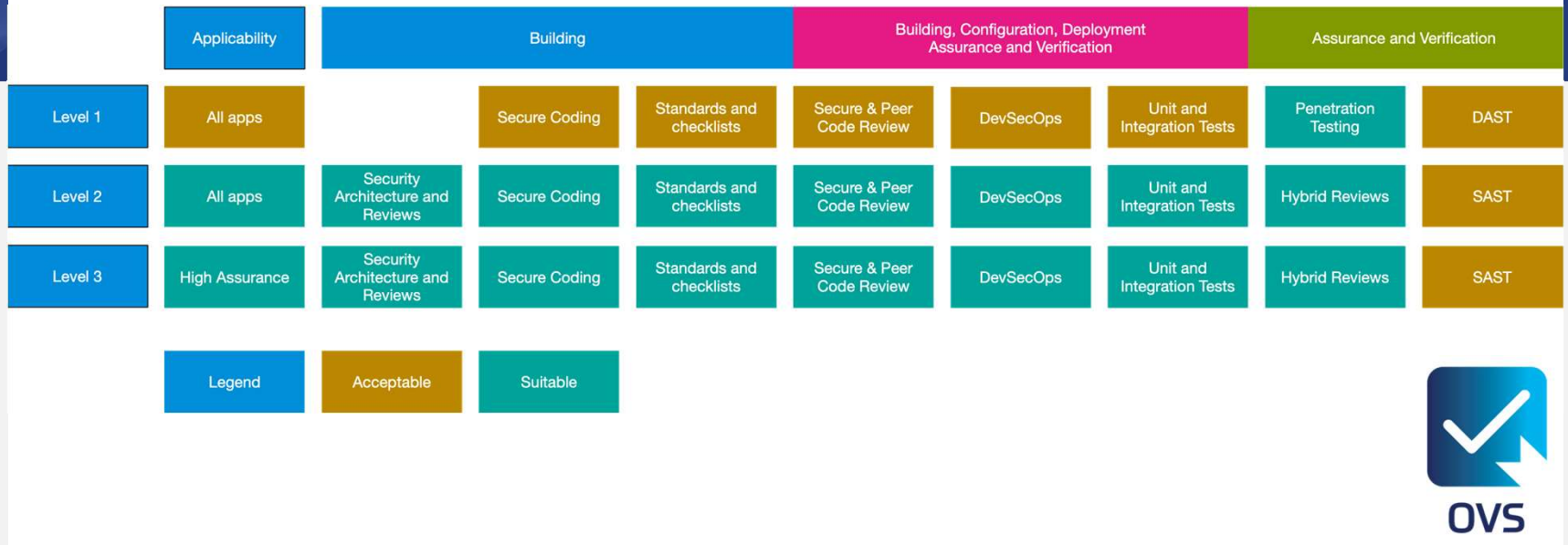
ANDREW VAN DER STOCK

https://owasp.org/Top10/A00_2021_How_to_use_the_OWASP_Top_10_as_a_standard/

TOP 10: HOW TO USE IT

WITH THE ASVS

Application Security Verification Standard



https://owasp.org/www-project-application-security-verification-standard/

https://www.crest-approved.org/membership/crest-ovs-programme/

TOP 10: HOW TO USE IT

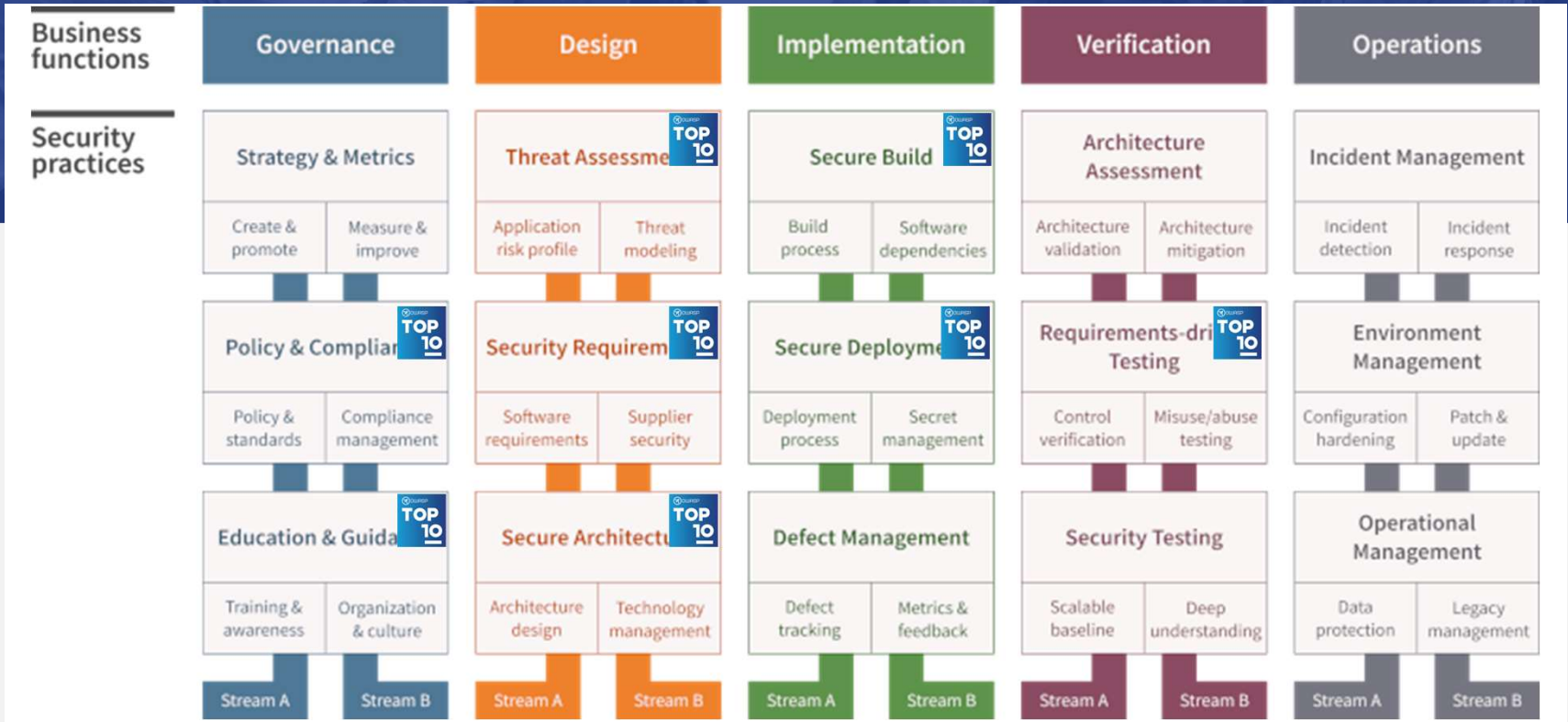
MAPPING TO ASVS

Use Case	Top 10	ASVS
Awareness	Yes	
Training	Entry level	Comprehensive
Design and architecture	Occasionally	Yes
Coding standard	Bare minimum	Yes
Secure code review	Bare minimum	Yes
Peer review checklist	Bare minimum	Yes
Unit testing	Occasionally	Yes
Integration testing	Occasionally	Yes
Penetration testing	Bare minimum	Yes
Tool support	Bare minimum	Yes
Secure supply chain	Occasionally	Yes

https://owasp.org/Top10/A00_2021_How_to_use_the_OWASP_Top_10_as_a_standard/

TOP 10: HOW TO USE IT

WITH THE SAMM



WHERE TO FROM HERE

01

USE THE TOP 10

as designed, ensure that your devs all understand it

02

REVIEW SSDLC

how far from baseline SAMM are you now?

03

REVIEW PEN TESTS

do yours cover more than just the Top 10?

04

REVIEW YOUR TOOLS

how far will they take you, are your vendors doing ASVS?

05

USE OWASP SAMM

start down the path of SAMM you'll not regret it

06

USE OWASP ASVS

make this the baseline requirements for devs

THANK YOU

FOR YOUR ATTENTION

WEB ADDRESS

https://owasp.org

https://securedelivery.io

TWITTER / SLACK / LINKEDIN

@rewtd

E-MAIL

grant.ongers@owasp.org

grant@securedelivery.io