

# 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



# \$ 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](#) =>



**Sam Stepanyan**  
@securestep9

@OWASPLondon Chapter Leader (#OWASP #OWASPLondon). Application Security (#AppSec) Consultant. WAF specialist. OWASP Nettacker Project co-leader. #CISSP

📍 London, UK 🌐 [medium.com/@securestep9](https://medium.com/@securestep9) 📅 Joined September 2013

# Open Web Application Security Project



The **OWASP<sup>®</sup>** 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**



meetup.com/owasp-london/

meetup Search for keywords Start a new group Log in Sign up

OWASP LONDON CHAPTER  
Join our network

Part of **OWASP® Foundation** - 227 groups

## OWASP London Chapter

London, United Kingdom

1,511 members · Public group

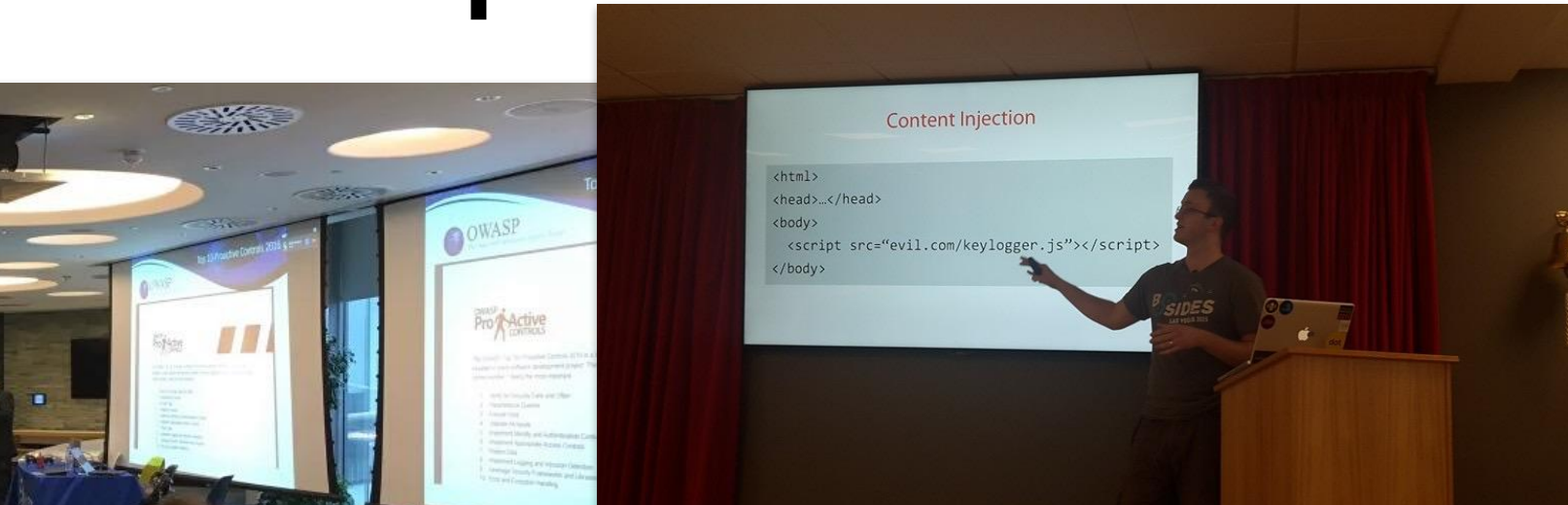
Organized by OWASP® F. and 3 others

Share:

About Events Members Photos Discussions [Join this group](#)

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

# Meetups



# Capture The Flag (CTF) Tournaments

@owaspplondon



Katy Anton @KatyAnton

29/11/2016

Awesome #hackaton session at @OWASPLondon. Really enjoyed the @codebashing code.

3 likes



Chris Highfield @cmhtw33t

28/11/2016

Continuing the @OWASPLondon hackathon on the train journey home.

1 like



Rich Fairhurst @richfairhurst

28/11/2016

Awesome session with @OWASPLondon and @codebashing this evening - I'm coming back for a drone tomorrow :-)

4 likes



KerberosMansour @Kerberosmansour

28/11/2016

@OWASPLondon hackathon has started many thanks to @thoughtworks for hosting us!

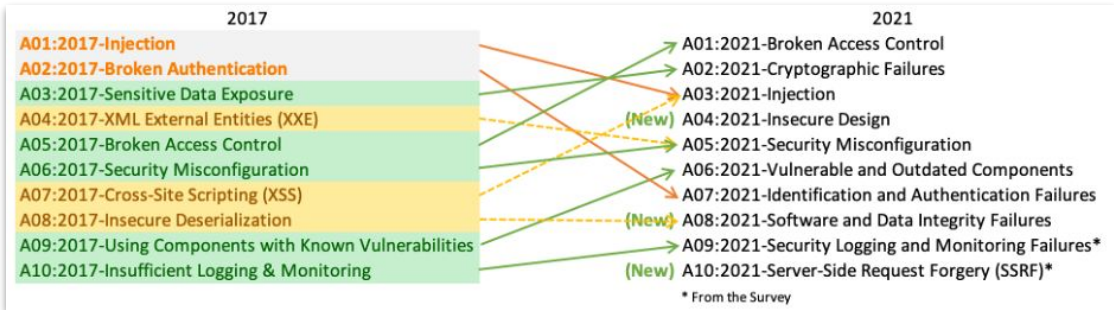




# OWASP® Top 10



- 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 Guide](#)
- [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](#)

## 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 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](#)

# OWASP 's Dinis Cruz and "SecDevOps"

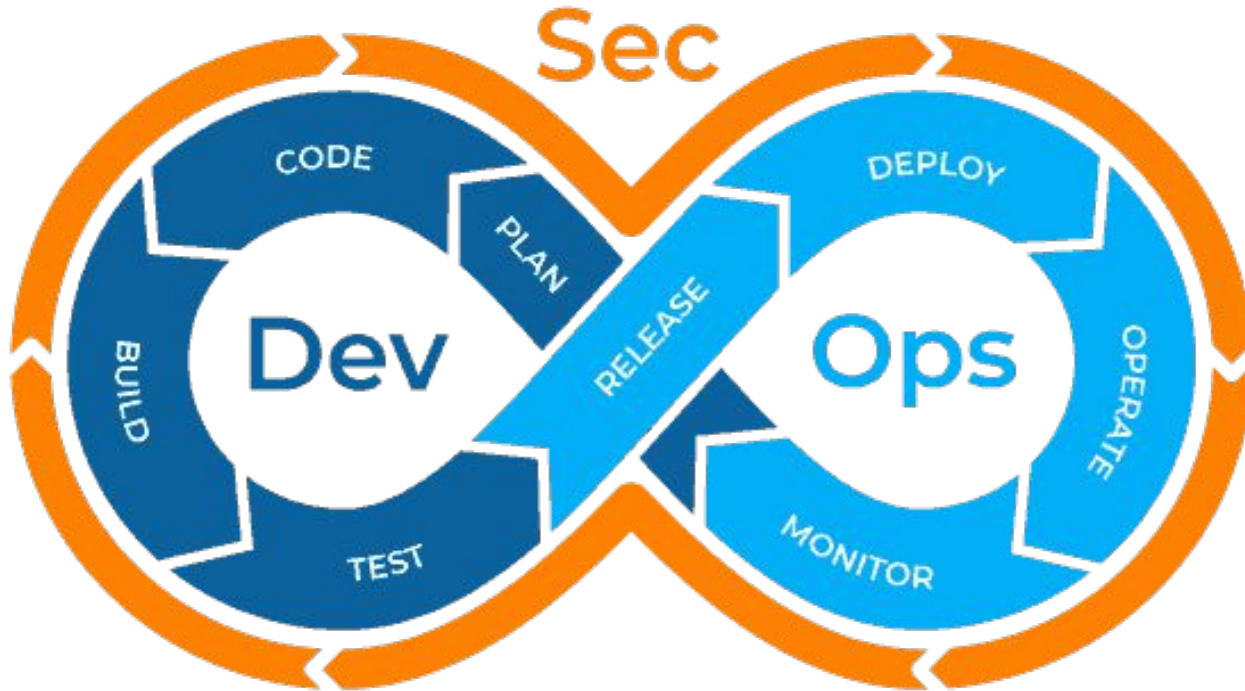


This book will give you a solution for the following common problems inside AppSec and Development teams:

- *"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/](https://github.com/DinisCruz/Book_SecDevOps_Risk_Workflow/)

# DevSecOps is...

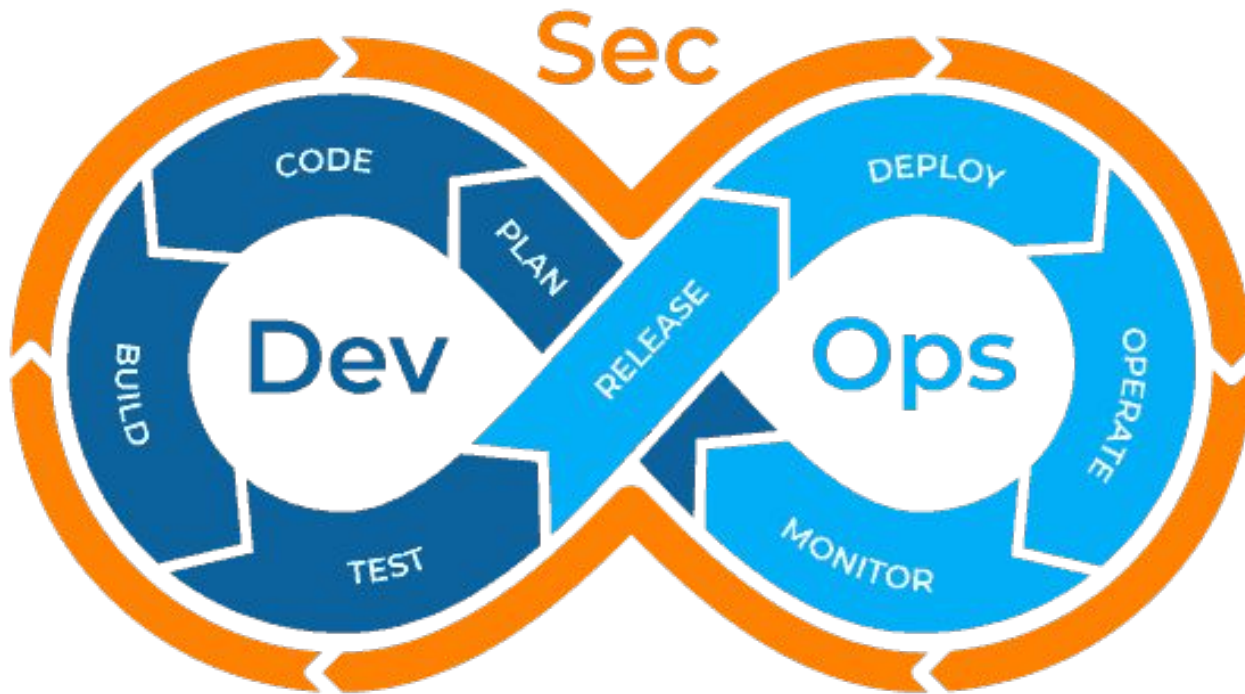


**Integration of Security Practices & Culture into DevOps Processes as a Shared Responsibility?**

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

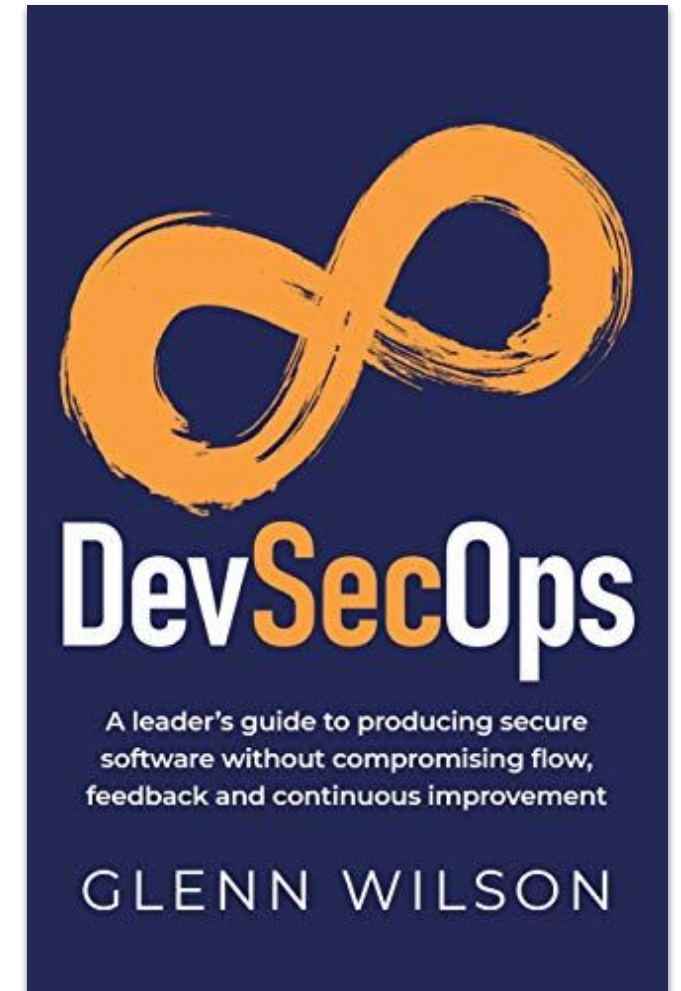
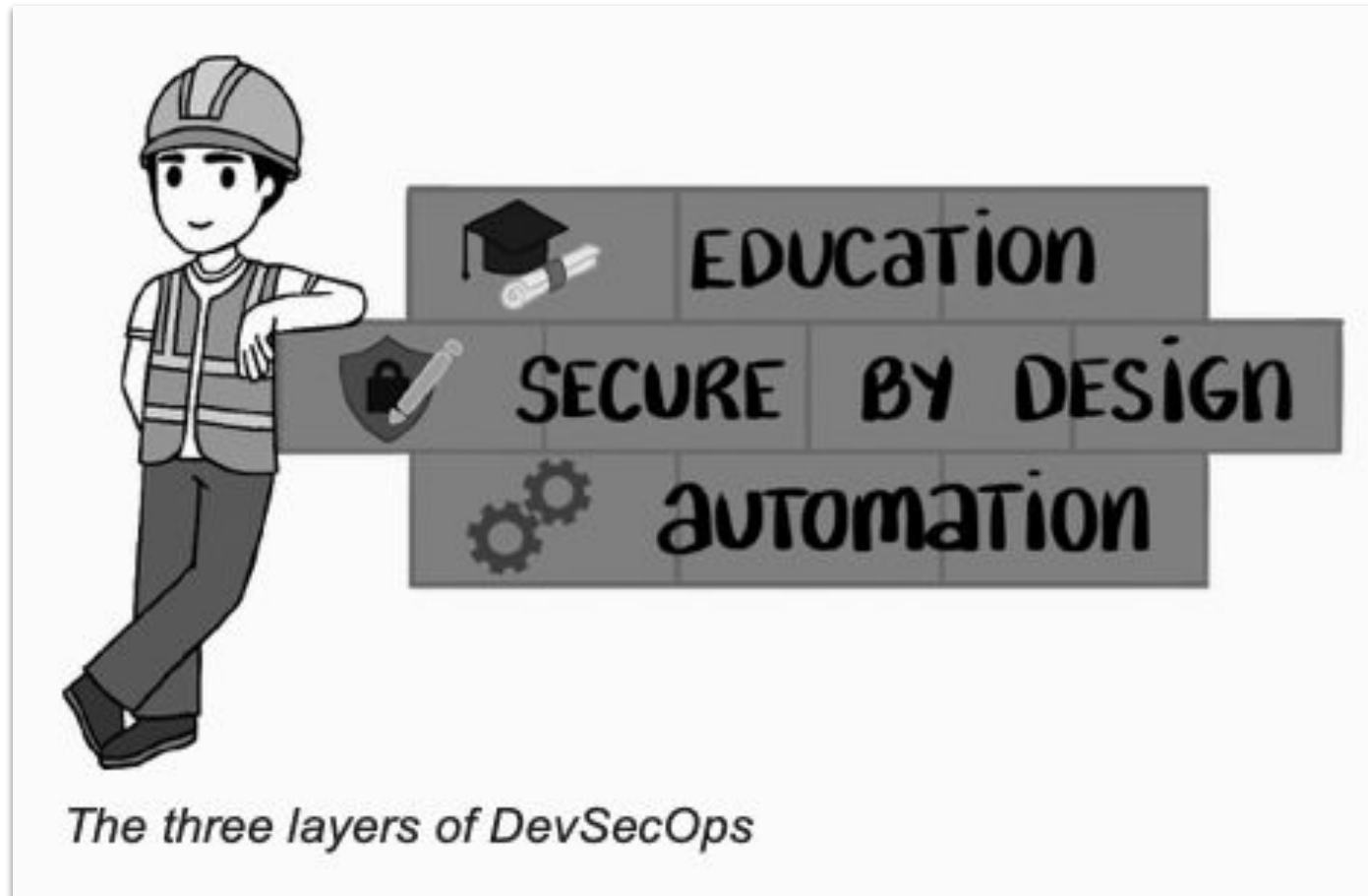
**People -> Process -> Technology**

# Sprinkle or Bake In?

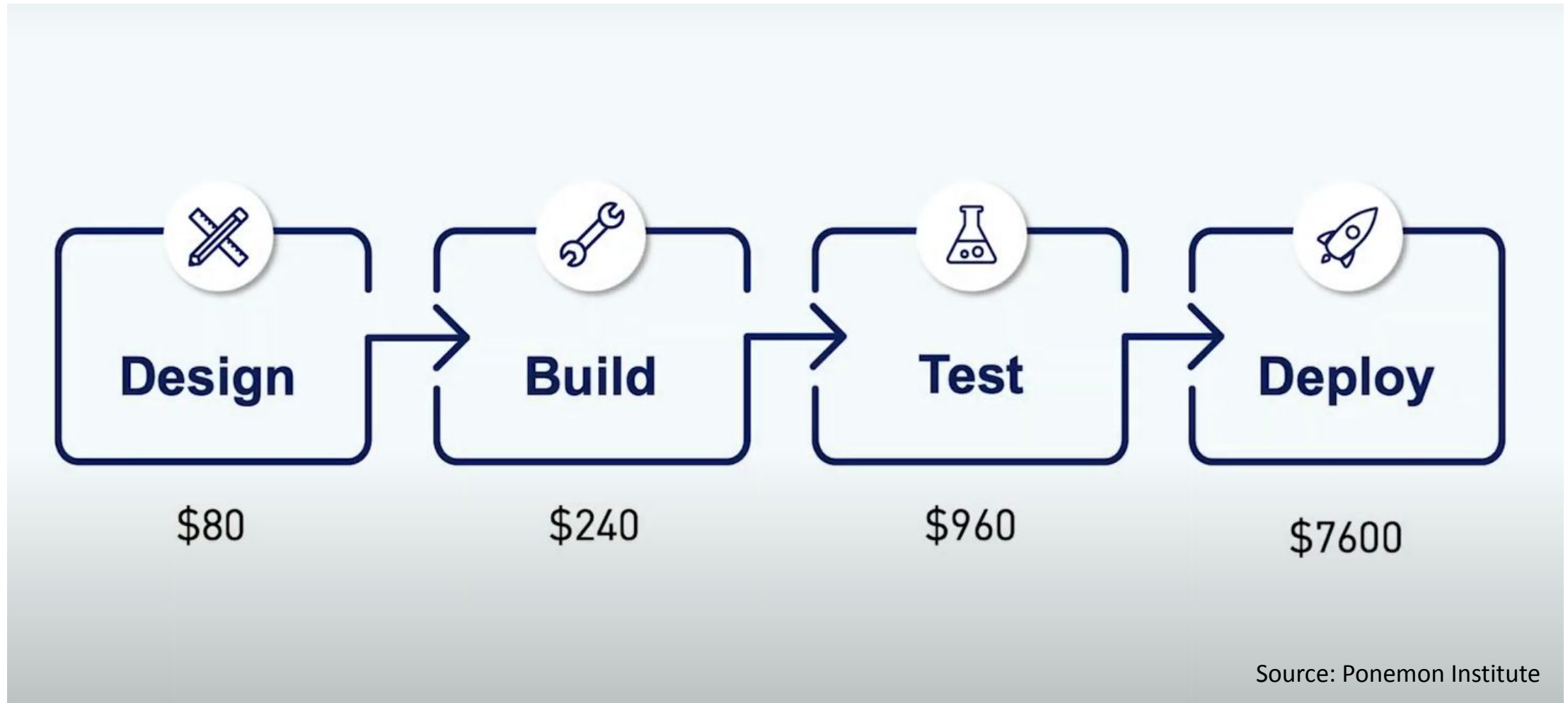


*Instead of “sprinkling”  
Security on top of  
your DevOps  
Processes - bake it in!*

# DevSecOps – The Three Layers



# Average Cost Of Fixing One Vulnerability

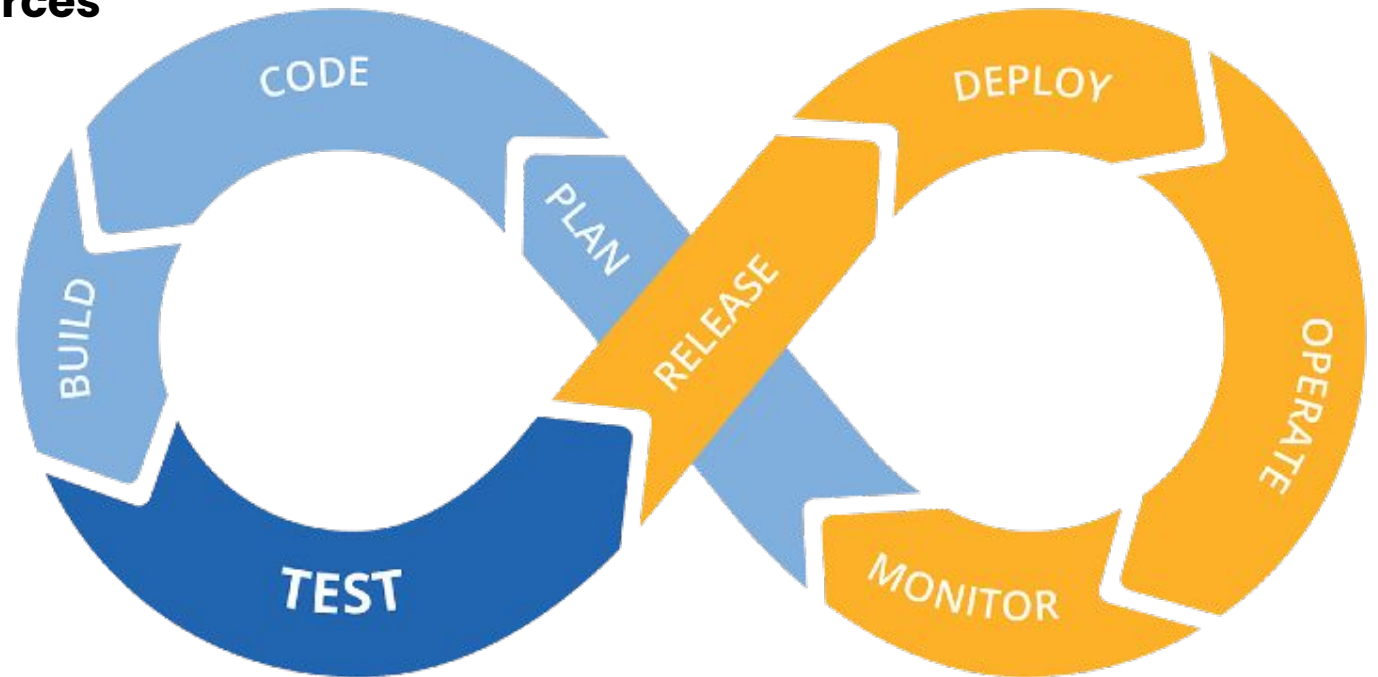


# OWASP TOOLS ARE FREE & OPENSOURCE

- **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**







**TOP 10**

**2021**

- **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**

# 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

- Factors
- Overview
- Description
  - Requirements and Resource Management
  - Secure Design
  - Secure Development Lifecycle
- How to Prevent
- Example Attack Scenarios
- References
- List of Mapped CWEs

## 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*.





# 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

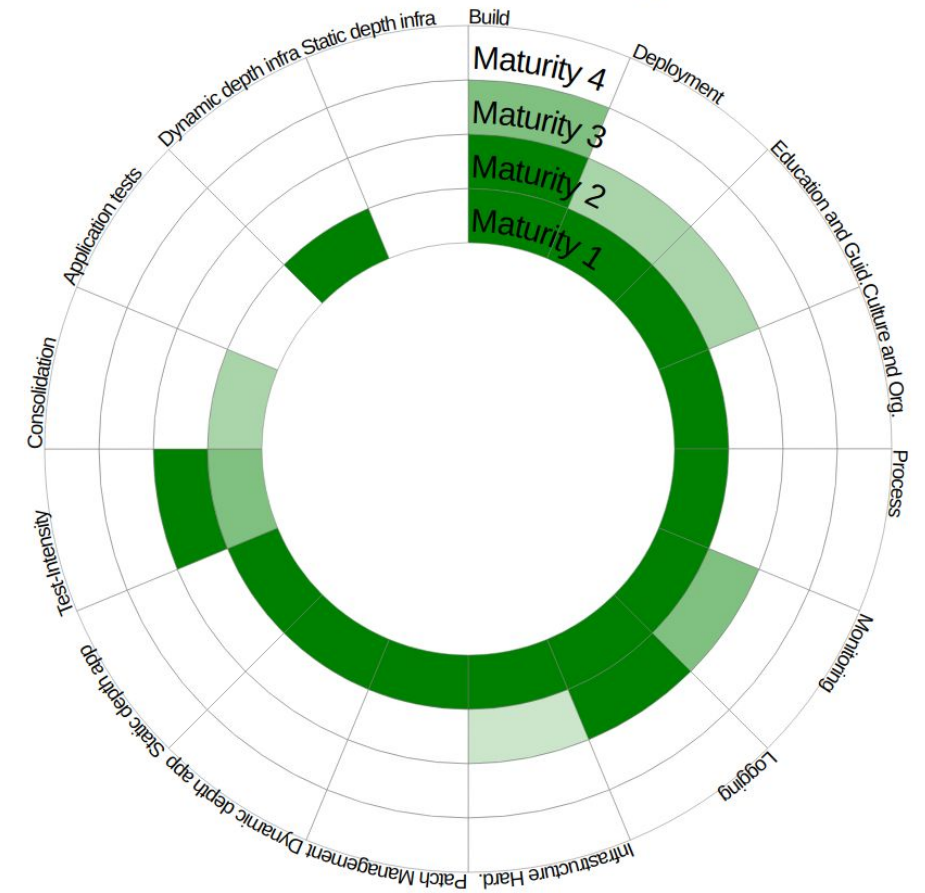
**Level 1: Basic understanding of security practices**

**Level 2: Adoption of basic security practices**

**Level 3: High adoption of security practices**

**Level 4: Advanced deployment of security practices at scale**

Identification of the degree of the implementation





# 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

github.com/OWASP/user-security-stories

Search or jump to... Pull req

OWASP / user-security-stories Public

<> Code Issues 2 Pull requests 1

master 1 branch 0 tags

DinisCruz Merge pull request #3 from ejohn20/mast

- LICENSE Initial commit
- README.md Added links to storie
- security-acceptance-criteri... Fixed word quotes.
- user-security-stories.md Fixed word quotes.

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



Threat Dragon v2.0.0-demo en

Logged in as Guest

Example

Client → Request → Web server → Response (HTTP) → Client

Web server → write → Github repo

Github repo → read → Web server

Properties

Name: Web server Description: The server providing the single-page web application

Out of Scope:  Reason for out of scope: [ ]

Privilege Level: [ ]

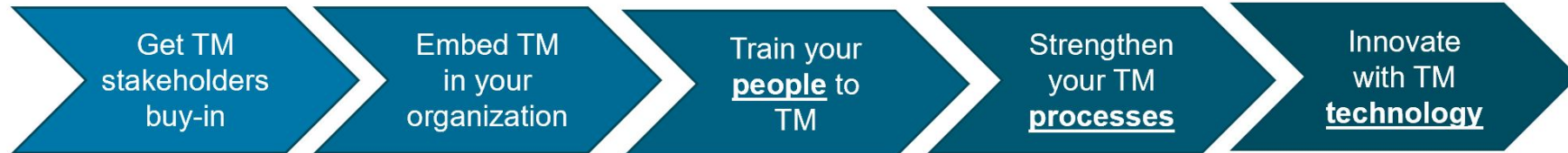
Threats

- # Insufficient logging Repudiation STRIDE
- # Log files are accessible Information disclosure STRIDE
- # Denial of Service using high rate of requests Denial of service STRIDE

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

## Threat Modeling Playbook



- Involve people and allocate time
- Inject TM expertise
- Show threat modeling ROI

- Establish context
- Assess and treat risk
- Monitor and review
- Communicate

- Identify stakeholders
- Create TM specialist role
- Train your people
- Create a positive TM culture

- Understand current process
- Introduce application risk levels
- Choose a TM methodology
- Perform and persist the TM
- Integrate with risk framework
- Follow up TM action items
- Optimize methodology and risk calculation

- Select the right tools
- Process the tools outcome
- Integrate in your TM methodology

# OWASP Cheatsheets



OWASP Cheat Sheet Series

Authentication Cheat Sheet

Introduction

**Authentication** is the process of verifying that an individual, entity or website is whom it claims to be. Authentication in the context of web applications is commonly performed by submitting a username or ID and one or more items of private information that only a given user should know.

**Session Management** is a process by which a server maintains the state of an entity interacting with it. This is required for a server to remember how to react to subsequent requests throughout a transaction. Sessions are maintained on the server by a session identifier which can be passed back and forth between the client and server when transmitting and receiving requests. Sessions should be unique per user and computationally very difficult to predict. The [Session Management Cheat Sheet](#) contains further guidance on the best practices in this area.

Authentication General Guidelines

User IDs

Make sure your usernames/user IDs are case-insensitive. User 'smith' and user 'Smith' should be the same user. Usernames should also be unique. For high-security applications, usernames could be assigned and secret instead of user-defined public data.

Email address as a User ID

For information on validating email addresses, please visit the [input validation cheatsheet email discussion](#).

Authentication Solution and Sensitive Accounts

- Do **NOT** allow login with sensitive accounts (i.e. accounts that can be used internally within

Table of contents

- Introduction
- Authentication General Guidelines
- User IDs
  - Email address as a User ID
- Authentication Solution and Sensitive Accounts
- Implement Proper Password Strength Controls
  - For more detailed information check
- Implement Secure Password Recovery Mechanism
- Store Passwords in a Secure Fashion
- Compare Password Hashes Using Safe Functions
- Change Password Feature
- Transmit Passwords Only Over TLS or Other Strong Transport
- Require Re-authentication for Sensitive Features
- Consider Strong Transaction Authentication
  - TLS Client Authentication
- Authentication and Error Messages
  - Authentication Responses
    - Incorrect and correct response examples
  - Login
  - Password recovery
  - Account creation

94 CheatSheets Published as of Dec 2024

# OWASP Secure Design Cheatsheet



OWASP Cheat Sheet Series



Search



OWASP/CheatSheetSeries  
☆ 27.1k 🗨 3.8k

OWASP Cheat Sheet Series

Password Storage

Pinning

Prototype Pollution Prevention

Query Parameterization

REST Assessment

REST Security

Ruby on Rails

SAML Security

SQL Injection Prevention

Secrets Management

Secure Cloud Architecture

Secure Product Design

Securing Cascading Style Sheets

Server Side Request Forgery Prevention

Session Management

Software Supply Chain Security.md

Symfony

TLS Cipher String

Third Party Javascript Management

Threat Modeling

Transaction Authorization

## Secure Product Design Cheat Sheet

### Introduction

The purpose of Secure Product Design is to ensure that all products meet or exceed the security requirements laid down by the organization as part of the development lifecycle and to ensure that all security decisions made about the product being developed are explicit choices and result in the correct level of security for the product being developed.

### Methodology

As a basic start, establish secure defaults, minimise the attack surface area, and fail securely to those well-defined and understood defaults.

Secure Product Design comes about through two processes:

1. **Product Inception;** and
2. **Product Design**

The first process happens when a product is conceived, or when an existing product is being re-invented. The latter is continuous, evolutionary, and done in an agile way, close to where the code is being written.

### Security Principles

Table of contents

Introduction

Methodology

Security Principles

1. The principle of Least Privilege and Separation of Duties
2. The principle of Defense-in-Depth
3. The principle of Zero Trust
4. The principle of Security-in-the-Open

Security Focus Areas

1. Context
2. Components
3. Connections
4. Code
5. Configuration

# 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.

Application Security Verification Standard 4.0.3  
Final  
October 2021

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.

Level	ASVS	Security	Configuration	Code	Architecture	Deployment	Operational	Testing	Tools	Other
Level 1	ASVS	ASVS	ASVS	ASVS	ASVS	ASVS	ASVS	ASVS	ASVS	ASVS
Level 2	ASVS	ASVS	ASVS	ASVS	ASVS	ASVS	ASVS	ASVS	ASVS	ASVS
Level 3	ASVS	ASVS	ASVS	ASVS	ASVS	ASVS	ASVS	ASVS	ASVS	ASVS

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 (binary) 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 flow and access control testing is only possible using human assistance. These should be turned into unit and integration tests.

OWASP Application Security Verification Standard 4.0.3 11

OWASP

VS.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, ports 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

VS.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 performed, but used to render the output safe in the appropriate output context for immediate use. Failing to output encode will result in an insecure, irrefutable, 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 ' or O'Hara). (C6)	✓	✓	✓	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 content-aware, preferably automated - or at least manual - output encoding protects against reflected, stored, and DOM based XSS. (C2)	✓	✓	✓	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 Application Security Verification Standard 4.0.3 39

# OWASP MobileASVS (MASVS)



## MASVS

Mobile Application Security  
Verification Standard




Sven Schleier  
Bernhard Mueller  
Jeroen Beckers

Carlos Holguera  
Jeroen Willemsen




## MASTG

Mobile Application Security  
Testing Guide



Sven Schleier  
Bernhard Mueller

Carlos Holguera  
Jeroen Willemsen



## MASWE

Mobile Application Security  
Weakness Enumeration



Sven Schleier  
Carlos Holguera  
Jeroen Beckers



# OWASP CSVS

## OWASP Container Security Verification Standard



Version 1.0  
Date: July 26<sup>th</sup>, 2019



### 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 namespaces 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



### 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 <i>dockerd</i> .			✓	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 <i>dockerd</i> is configured with <i>live restore</i> enabled.		✓	✓	1.0
2.10	Verify that permissions to the configuration of <i>dockerd</i> 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: JSESSIONID=34JHURHD894LOP04957HR49E3E383940123K
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.

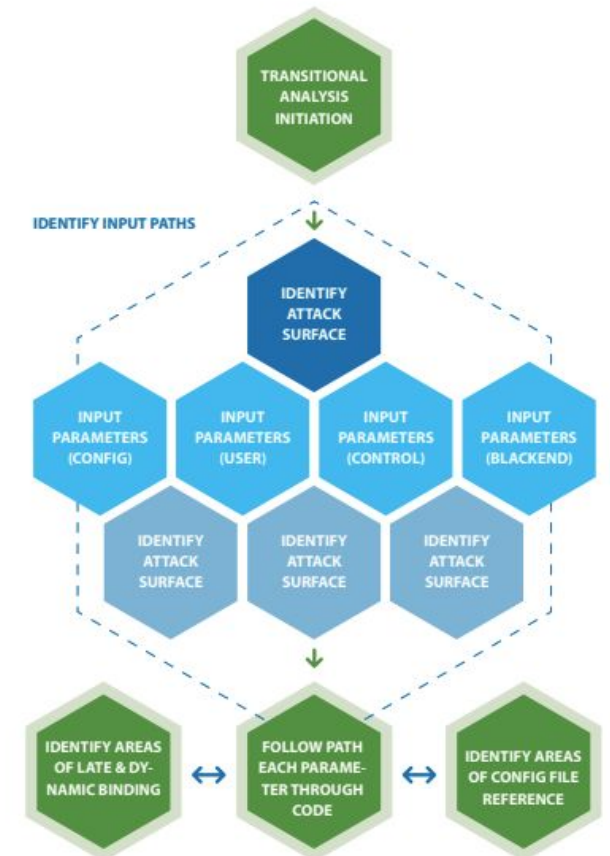
#### Prevention Measures That Do NOT Work

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.

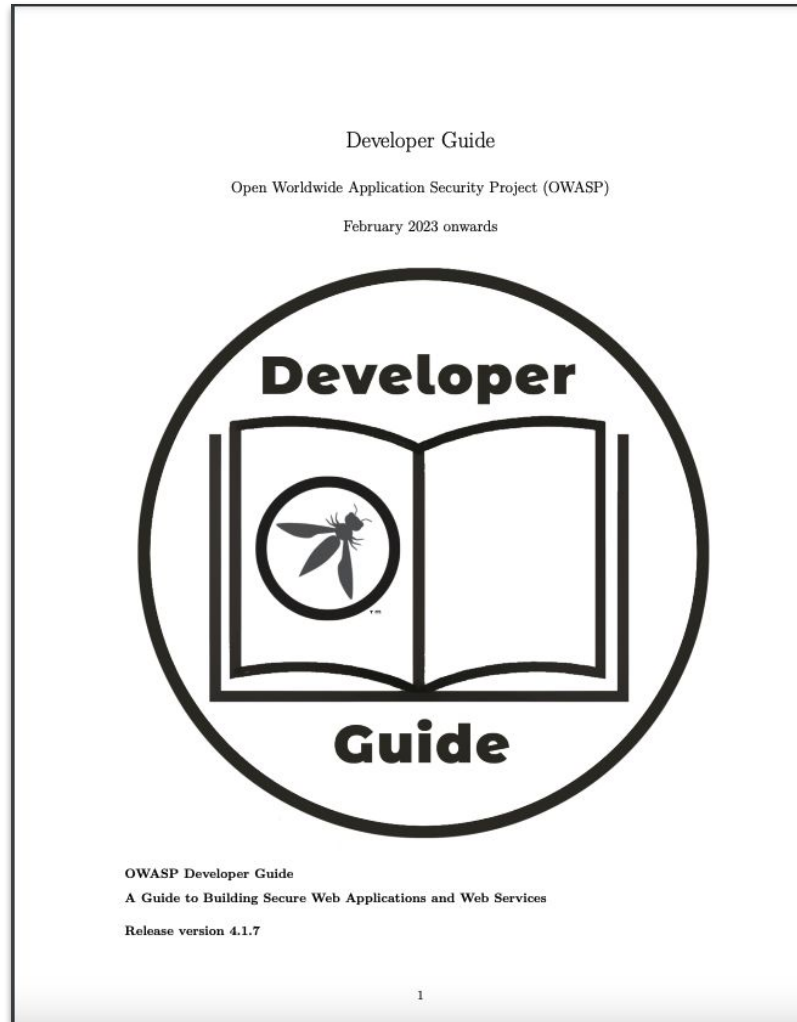
**Table 21:** Unsuccessful Countermeasures For Csrf Attacks

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



## 1 Introduction

## 2 Foundations

- 2.1 Security fundamentals
- 2.2 Secure development and integration
- 2.3 Principles of security
- 2.4 Principles of cryptography
- 2.5 OWASP Top 10

## 3 Requirements

- 3.1 Requirements in practice
- 3.2 Risk profile
- 3.3 OpenCRE
- 3.4 SecurityRAT
- 3.5 ASVS requirements
- 3.6 MAS requirements
- 3.7 SKF requirements

## 4 Design

- 4.1 Threat modeling
  - 4.1.1 Threat modeling in practice
  - 4.1.2 pytm
  - 4.1.3 Threat Dragon
  - 4.1.4 Cornucopia
  - 4.1.5 LINDDUN GO
  - 4.1.6 Threat Modeling toolkit
- 4.2 Web application checklist
  - 4.2.1 Checklist: Define Security Requirements
  - 4.2.2 Checklist: Leverage Security Frameworks and Libraries
  - 4.2.3 Checklist: Secure Database Access
  - 4.2.4 Checklist: Encode and Escape Data
  - 4.2.5 Checklist: Validate All Inputs

# OWASP CycloneDX



GETTING STARTED

SPECIFICATION

ABOUT



## Specification Overview

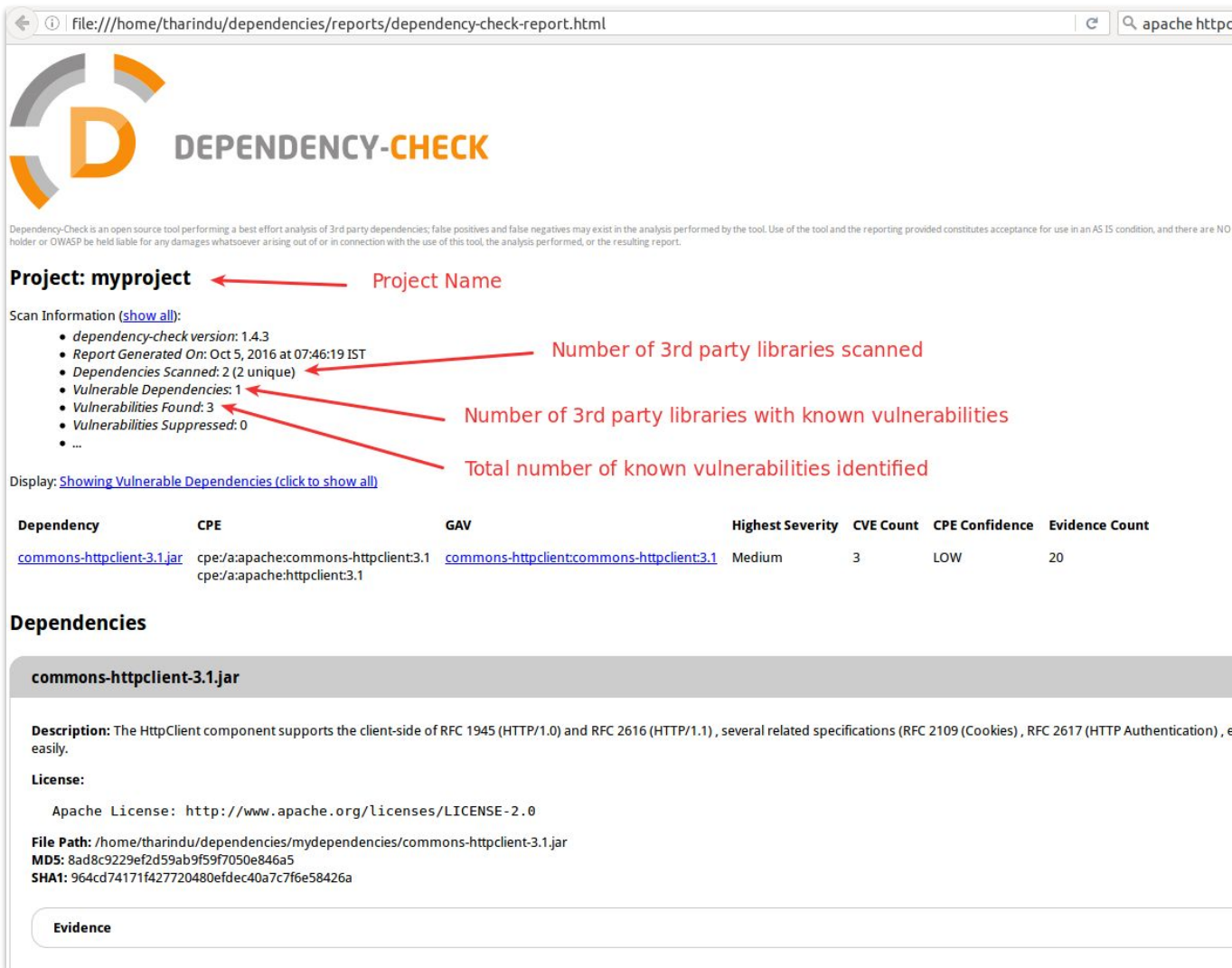


The CycloneDX object model:

- 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](#), [SaaSOM](#), [OBOM](#), [MBOM](#), and [VEX](#) use cases
- can easily describe complex relationships
- is [extensible](#) to support specialized and future use cases

BOM
BOM Metadata Information
Components Information
Services Information
Dependency Relationships
Compositions
Vulnerabilities
Extensions

# OWASP Dependency Check



The screenshot shows the OWASP Dependency Check report for a project named 'myproject'. The report includes scan information, a table of vulnerable dependencies, and a detailed view of the 'commons-httpclient-3.1.jar' dependency.

**Project:** myproject (Project Name)

**Scan Information (show all):**

- dependency-check version: 1.4.3
- Report Generated On: Oct 5, 2016 at 07:46:19 IST
- Dependencies Scanned: 2 (2 unique) (Number of 3rd party libraries scanned)
- Vulnerable Dependencies: 1 (Number of 3rd party libraries with known vulnerabilities)
- Vulnerabilities Found: 3 (Total number of known vulnerabilities identified)
- Vulnerabilities Suppressed: 0
- ...

Display: [Showing Vulnerable Dependencies \(click to show all\)](#)

Dependency	CPE	GAV	Highest Severity	CVE Count	CPE Confidence	Evidence Count
<a href="#">commons-httpclient-3.1.jar</a>	cpe:/a:apache:commons-httpclient:3.1 cpe:/a:apache:httpclient:3.1	<a href="#">commons-httpclient:commons-httpclient:3.1</a>	Medium	3	LOW	20

**Dependencies**

**commons-httpclient-3.1.jar**

**Description:** The HttpClient component supports the client-side of RFC 1945 (HTTP/1.0) and RFC 2616 (HTTP/1.1), several related specifications (RFC 2109 (Cookies), RFC 2617 (HTTP Authentication)), et easily.

**License:**

Apache License: <http://www.apache.org/licenses/LICENSE-2.0>

**File Path:** /home/tharindu/dependencies/mydependencies/commons-httpclient-3.1.jar

**MD5:** 8ad8c9229ef2d59ab9f59f7050e846a5

**SHA1:** 964cd74171f427720480efdec40a7c7f6e58426a

**Evidence**

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



<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 1<sup>ST</sup> 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



## OWASP Top 10 for LLM Applications 2025

---

**Version 2025**  
**November 18, 2024**

OWASP PDF v4.2.0a 20241114-202703



# GenAI.OWASP.org

The screenshot shows the website [genai.owasp.org](https://genai.owasp.org). The navigation menu includes: [GETTING STARTED](#), [PROJECT](#), [INITIATIVES](#), [BLOG](#), and [ABOUT](#). The main banner features the OWASP logo and the text "TOP 10 LLM APPLICATIONS & GENERATIVE AI". The central focus is the "AI Security Center of Excellence Guide", marked as "NEW!". The guide's description states: "Establishing a Generative AI Security COE is essential. This document offers a best practices framework for teams, including cross-functional OKRs and KPIs, to streamline implementation." Below the text are two buttons: "Learn More" and "Download Now". At the bottom, there is a "Join the Newsletter" section with an "Email\*" input field containing the placeholder "Enter Your Email" and a "SUBSCRIBE" button.

# 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:

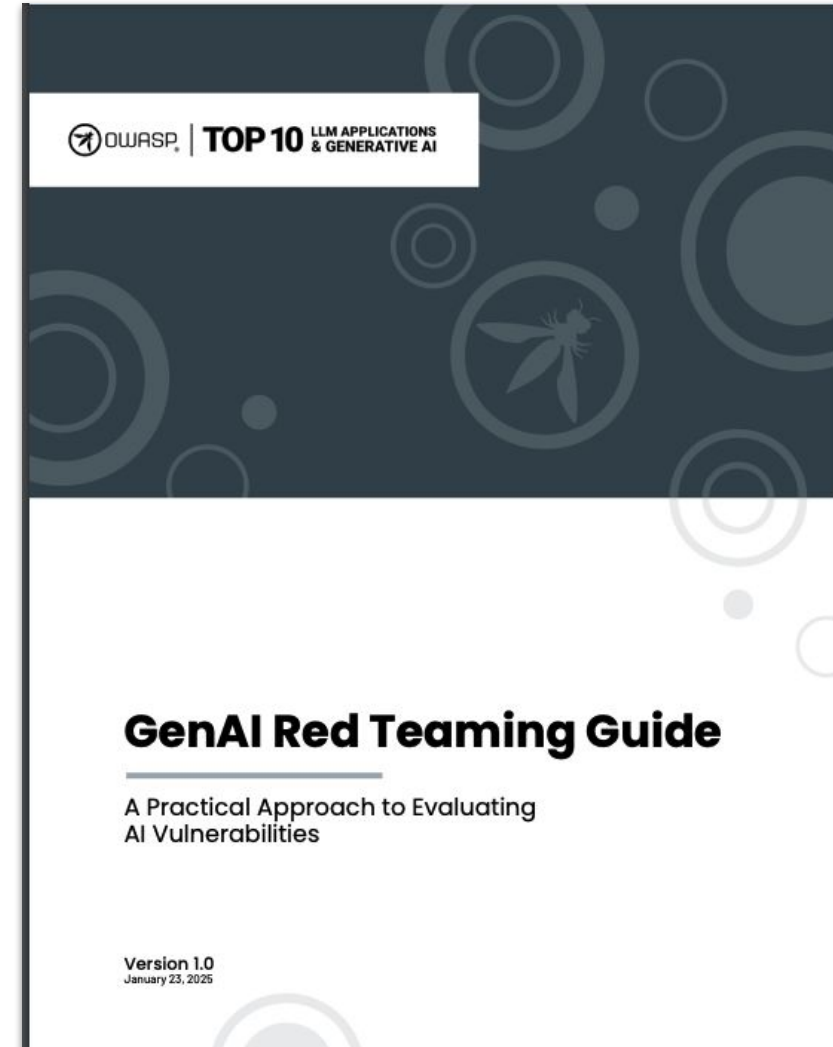
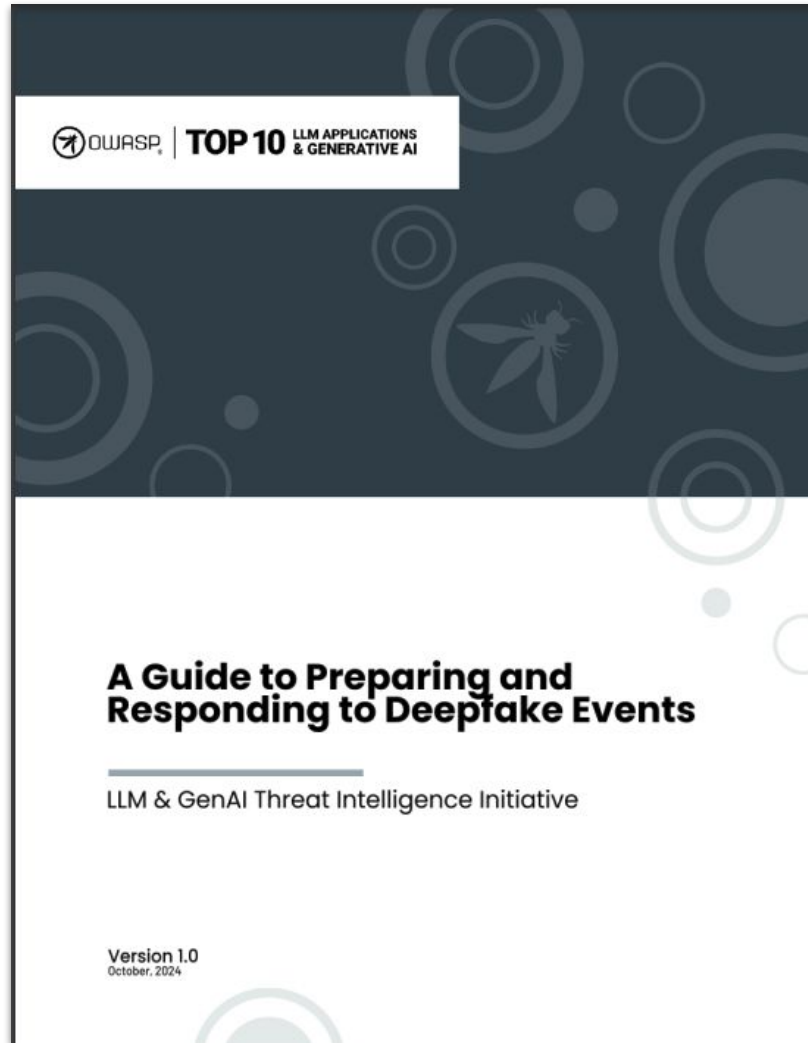
1. [How to address AI security](#)
2. [How to address AI privacy](#)

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 – OWASP AI.org

## AI security threats and controls navigator from the OWASP AI Exchange at [owaspai.org](https://owaspai.org)

LEGEND:

Group of controls, ordered by threat or type (clickable)

▶ Standard information security CONTROL (with attention points)

▶ Runtime Data science CONTROL

▶ Development-time Data science CONTROL

▶ Other CONTROL

Impact on Confidentiality, Integrity or Availability

### 1 General controls against all threats

#### Governance

- ▶ AIPROGRAM
- ▶ SECPROGRAM
- ▶ SECDEVPROGRAM
- ▶ DEVPROGRAM
- ▶ CHECKCOMPLIANCE
- ▶ SECEDUCATE

#### Deal with behaviour integrity issues

- ▶ OVERSIGHT
- ▶ LEASTMODELPRIVILEGE
- ▶ AITRANSARENCY
- ▶ CONTINUOUSVALIDATION
- ▶ EXPLAINABILITY
- ▶ UNWANTEDBIATESTING

#### Sensitive data limitation

- ▶ DATAMINIMIZE
- ▶ ALLOWEDDATA
- ▶ SHORTRETAIN
- ▶ OBFUSCATETRAININGDATA
- ▶ DISCRETE

### 2 Controls against threats through runtime use

#### Always against use threats

- ▶ MONITORUSE
- ▶ RATELIMIT
- ▶ MODELACCESSCONTROL

#### Integrity of model behaviour

##### 2.1 Against evasion

- ▶ See Always
- ▶ DETECTODDINPUT
- ▶ DETECTADVERSARIALINPUT
- ▶ EVASIONROBUSTMODEL
- ▶ TRAINADVERSARIAL
- ▶ INPUTDISTORTION
- ▶ ADVERSARIALROBUSTDISTILLATION

#### Confidentiality of train data

##### 2.2 Against data disclosure by use

###### 2.2.1 Against data disclosure by model

- ▶ See always
- ▶ FILTERSENSITIVEMODELOUTPUT

###### 2.2.2 Against model inversion and membership inference

- ▶ See always
- ▶ OBSCURECONFIDENCE
- ▶ SMALLMODEL

#### Confidentiality of model IP

##### 2.3 Against model theft by use

- ▶ See always

#### Availability of model

##### 2.4 Against failure by use

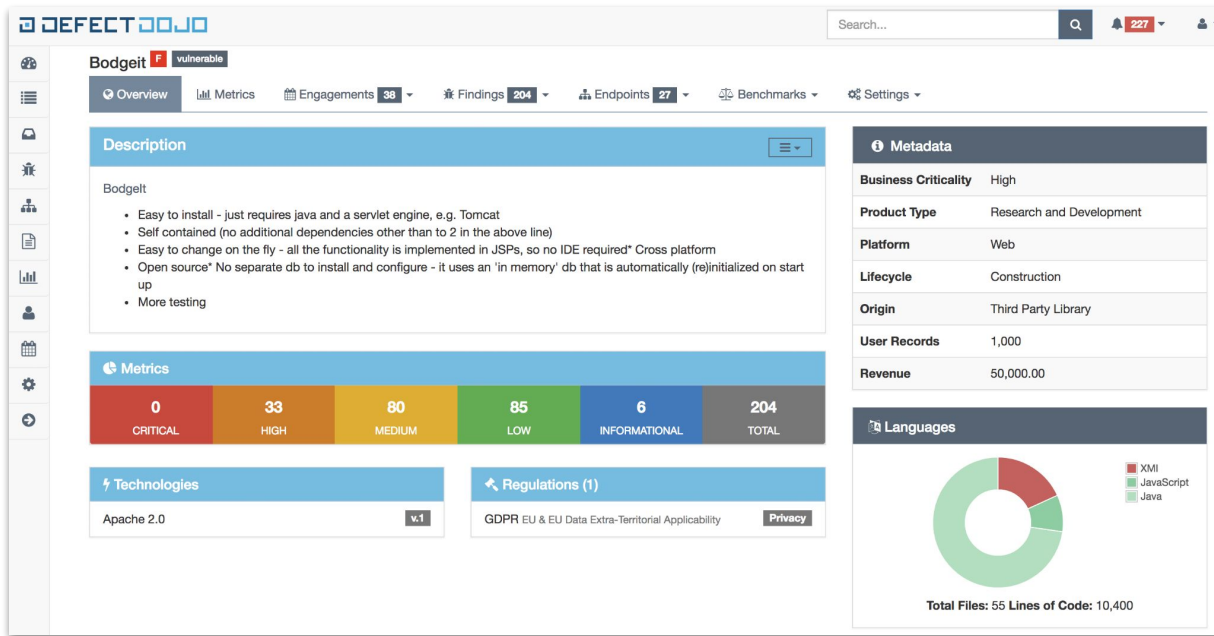
- ▶ See always
- ▶ DOSINPUTVALIDATION
- ▶ LIMITRESOURCES

# OWASP DefectDojo

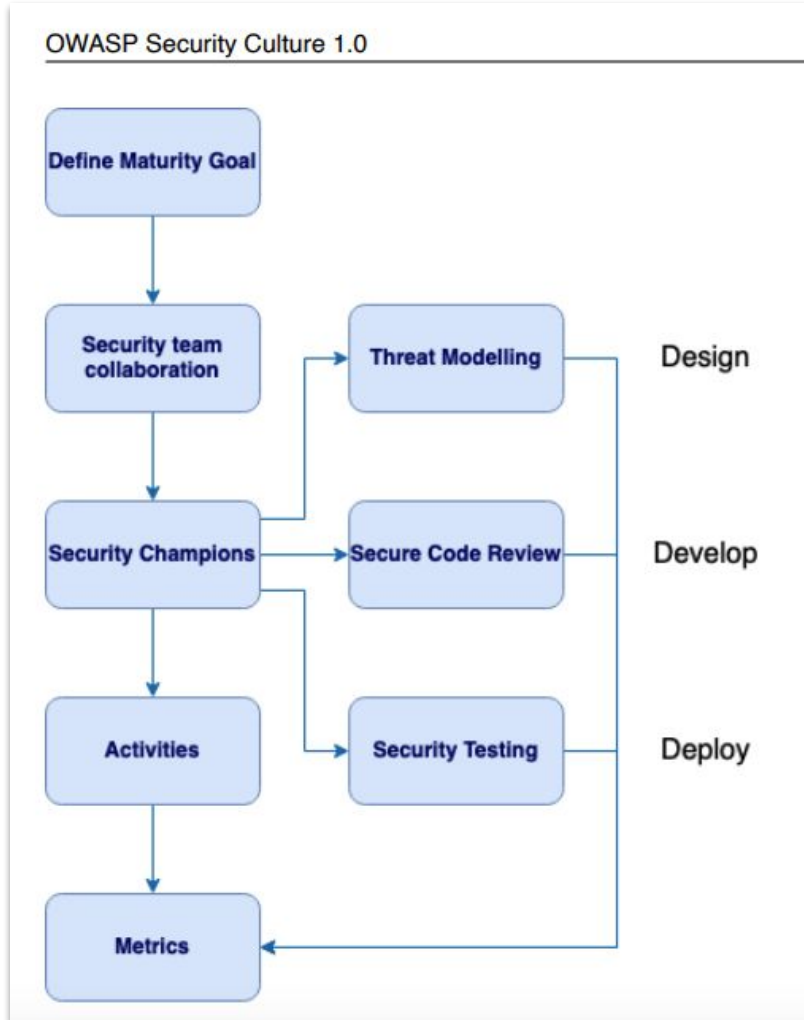


**Defect Dojo** is a security orchestration and vulnerability management platform. Aka the **“Single Pane of Glass”**.

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

0. [Frontispiece](#)
1. [Introduction](#)
2. [Why Add Security In Development Teams](#)
3. [Goal Setting and Security Team Collaboration](#)
4. [Security Champions](#)
5. [Activities](#)
6. [Threat Modelling](#)
7. [Security Testing](#)
8. [Metrics](#)





# Security Champions playbook



## Identify teams

- 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

## Define the role

- 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

## Nominate champions

- 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

## Comm channels

- 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-weekly should be fine to start with

## Knowledge base

- 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

## Maintain interest

- 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 WrongSecrets [Tweet](#)

Java checkstyle and testing **passing** Terraform FMT **passing** Test minikube script (k8s) **passing**  
Test minikube script (k8s&vault) **passing** Docker container test **passing** OWASP lab project discussions **2 total**

Welcome to the OWASP WrongSecrets p0wnable app. With this app, we have packed various ways of how to not store your secrets. These can help you to realize whether your secret management is ok. The challenge is to find all the different secrets by means of various tools and techniques.

Can you solve all the 24 challenges?

WrongSecrets Home Challenges ▾ Github 🌞 🌙 Environment:HEROKU\_DOCKER Version:1.3.4Dheroku

## Challenge 1

Welcome to challenge 1. You need to guess the secret that is hidden in [Java](#), [Docker](#), Kubernetes, Vault, AWS or GCP.

# OWASP Security Shepherd

## Security Shepherd



Admin

Scoreboard

Lessons

- XBroken Crypto
- XBroken Session Management
- XClient Side Injection
- XContent Provider Leakage
- XCross Site Request Forgery
- XCross Site Scripting
- XFailure to Restrict URL Access
- XInsecure Cryptographic Storage
- XInsecure Data Storage
- XInsecure Direct Object
- References
- XPoor Authentication
- XPoor Data Validation
- XReverse Engineering
- XSecurity Misconfiguration
- XSQL Injection

Submit Result Key Here...

## SQL Injection Lesson

Injection flaws, such as **SQL injection**, occur when hostile data is sent to an interpreter as part of a command or query. The hostile data can trick the interpreter into executing unintended commands or accessing unauthorized data. Injection attacks are of a high severity. Injection flaws can be exploited to remove a system's confidentiality by accessing any information held on the system. These security risks can then be extended to execute updates to existing data affecting the systems integrity and availability. These attacks are easily exploitable as they can be initiated by anyone who can interact with the system through any data they pass to the application.

The following form's parameters are concatenated to a string that will be passed to a SQL server. This means that the data can be interpreted as part of the code.

The objective here is to modify the result of the query with **SQL Injection** so that all of the table's rows are returned. This means you want to change the **boolean** result of the query's **WHERE** clause to return true for every row in the table. The easiest way to ensure the **boolean** result is always true is to inject a **boolean 'OR'** operator followed by a true statement like `1 = 1`.

If the parameter is been interpreted as a string, you can escape the string with an apostrophe. That means that everything after the apostrophe will be interpreted as SQL code.

Exploit the **SQL Injection** flaw in the following example to retrieve all of the rows in the table. The lesson's solution key will be found in one of these rows! The results will be posted beneath the search form.

Please enter the **user name** of the user that you want to look up

## Scoreboard


The OWASP Security Shepherd Project



1st:	dcua	36	15	6	3941
2nd:	NULL Life	18	12		3558
3rd:	arusell	2	2	3	3053
4th:	andro1de	1	3		2996
5th:	micaman	1	1	1	2966
6th:	Insanity	3	10	2	2909
7th:	longerthan5characters	1	2	3	2878
8th:	aiacobelli	1	1		2516
9th:	ottucsakj	3	3	2	2501
10th:	mfocuz	2	4		2084

# OWASP WebGoat





## Report card

- Introduction >
- General >
- (A1) Injection >
- (A2) Broken Authentication >
- (A3) Sensitive Data Exposure >
- (A4) XML External Entities (XXE) >
- (A5) Broken Access Control >
- (A7) Cross-Site Scripting (XSS) >
- (A8) Insecure Deserialization >
- (A9) Vulnerable Components >
- (A8:2013) Request Forgeries >**
- Cross-Site Request Forgeries
- Server-Side Request Forgery
- Client side >
- Challenges >

OVERVIEW	
Total number of lessons	32
Total number of lessons solved	0
Total number of assignments	85
Total number of assignments solved	0

LESSON OVERVIEW	
Lesson name	Solved
Without password	false
Admin password reset	false
Admin lost password	false
Without account	false
Bypass front-end restrictions	false
Client side filtering	false
Crypto Basics	false
Cross Site Scripting	false

### Learn in three steps



#### Explain the vulnerability

Teaching is now a first class citizen of WebGoat, we explain the vulnerability. Instead of 'just hacking' we now focus on explaining from the beginning what for example a SQL injection is.



#### Learn by doing

During the explanation of a vulnerability we build assignments which will help you understand how it works.




#### Explain mitigation

At the end of each lesson you will receive an overview of possible mitigations which will help you during your development work.

# OWASP JuiceShop

## OWASP Juice Shop

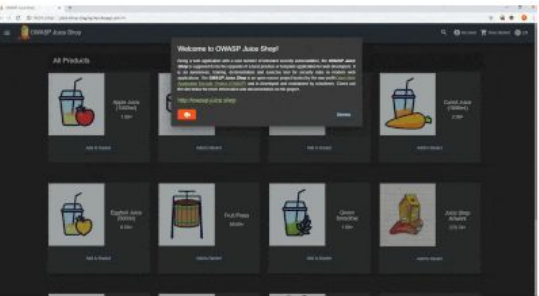
[Main](#) [Overview](#) [News](#) [Challenges](#) [Learning](#) [CTF](#) [Ecosystem](#) [Supporters](#)



owasp **flagship project** release v14.2.1 GitHub **7.2k** **Follow** **4.8k**

openssf **best practices** **gold** Contributor Covenant **v2.0 adopted**

OWASP Juice Shop is probably the most modern and sophisticated insecure web application! It can be used in security trainings, awareness demos, CTFs and as a guinea pig for security tools! Juice Shop encompasses vulnerabilities from the entire OWASP Top Ten along with many other security flaws found in real-world applications!



### Description



- XSS /CROSS/SITE/SCRIPTING
- Sensitive Data Exposure
- Improper Input Validation
- Broken Access Control
- Vulnerable Components
- Broken Authentication
- Broken Access Control
- Vulnerable Components
- BROKEN ACCESS CONTROL
- Vulnerable Components
- Security through Obscurity
- Insecure Deserialization
- Broken Anti Automation
- Security by Obscurity
- Insecure Deserialization
- BROKEN Anti-Automation
- Injection
- Security Misconfiguration
- Cryptographic Issues
- INJECTION
- Security Misconfiguration
- d R!t4gSx!l?s
- CRYPTOGRAPHY
- Unvalidated Redirects
- Miscellaneous
- XXE
- Unvalidated Redirects
- Miscellaneous Threats
- XML EXTERNAL ENTITY

# 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.

**WILD CARD**

## Joker

Alice can utilize the application to attack users' systems and data

*Have you thought about becoming an individual OWASP member? All tools, guidance and local meetings are free for everyone, but individual membership helps support OWASP's work*

**SESSION MANAGEMENT**

## 9

Ivan can steal session identifiers because they are sent over insecure channels, or are logged, or are revealed in error messages, or are included in URLs, or are accessible unnecessarily by code which the attacker can influence or alter

OWASP SCP	69, 75, 76, 119, 138
OWASP ASVS	3.5, 8.10, 11.4
OWASP AppSensor	SE4-6
CAPEC	31, 60
SAFECODE	28

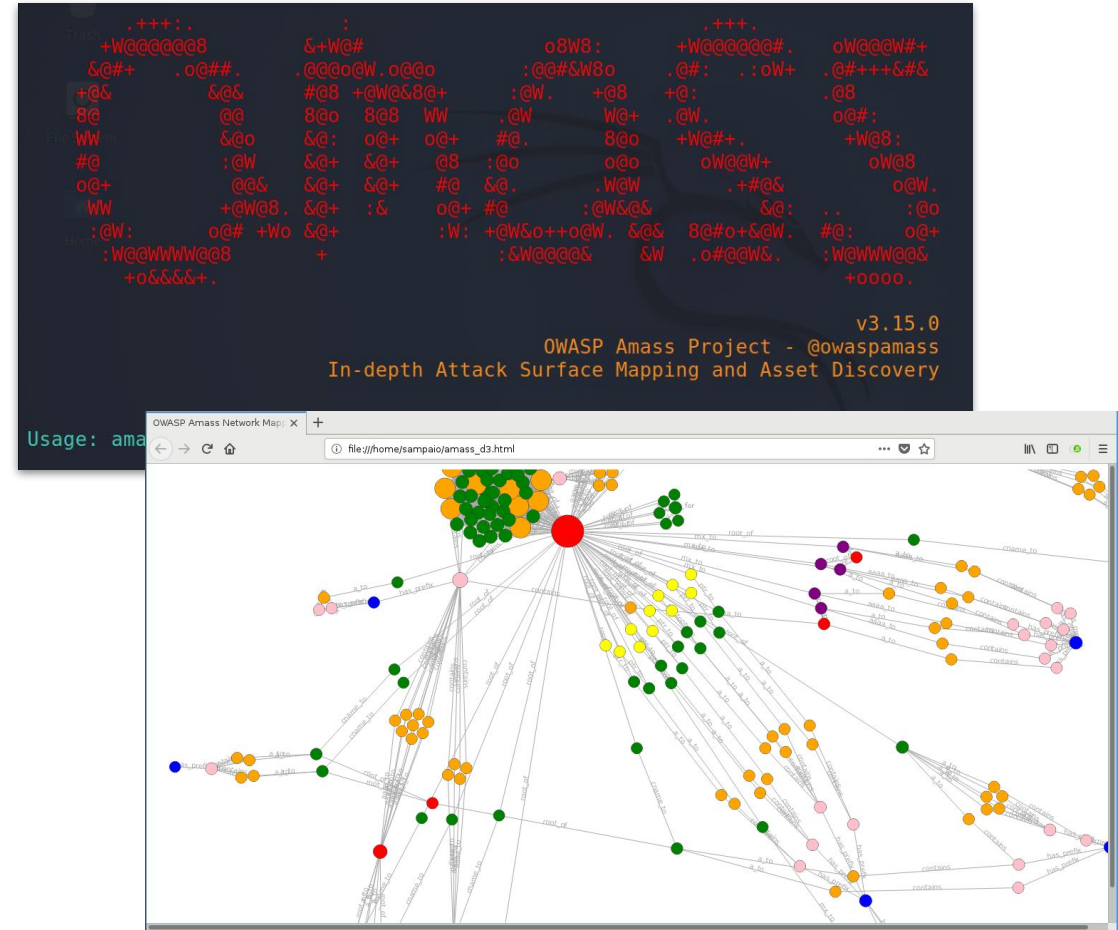
OWASP Cornucopia Ecommerce Website Edition v1.04

# 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



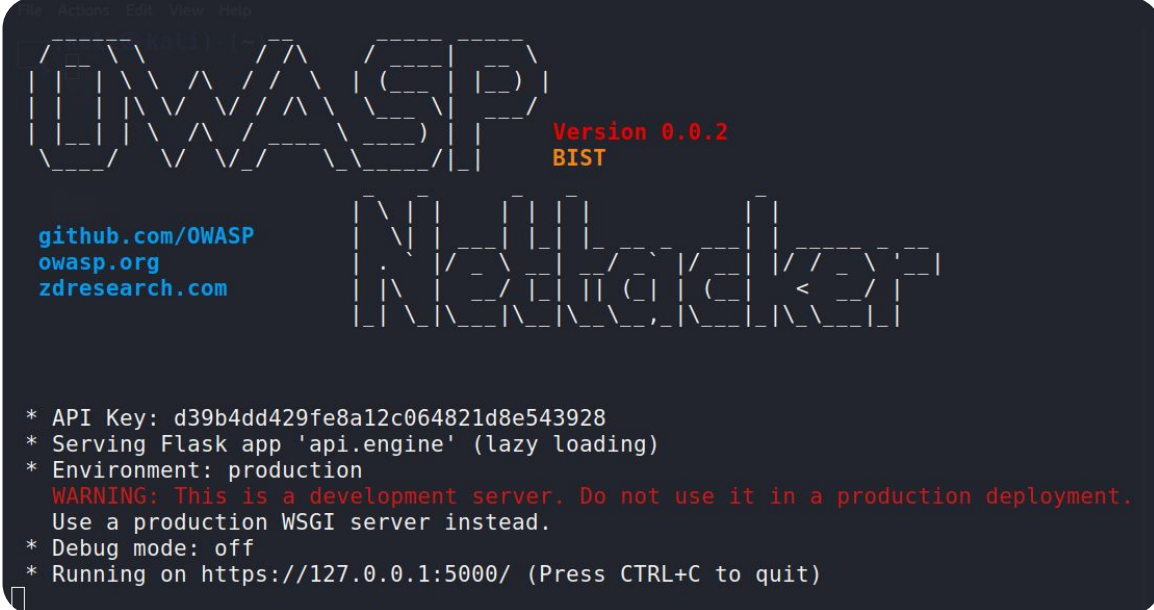
# OWASP Nettacker

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.



```
OWASP Version 0.0.2
BIST

github.com/OWASP
owasp.org
zdresearch.com

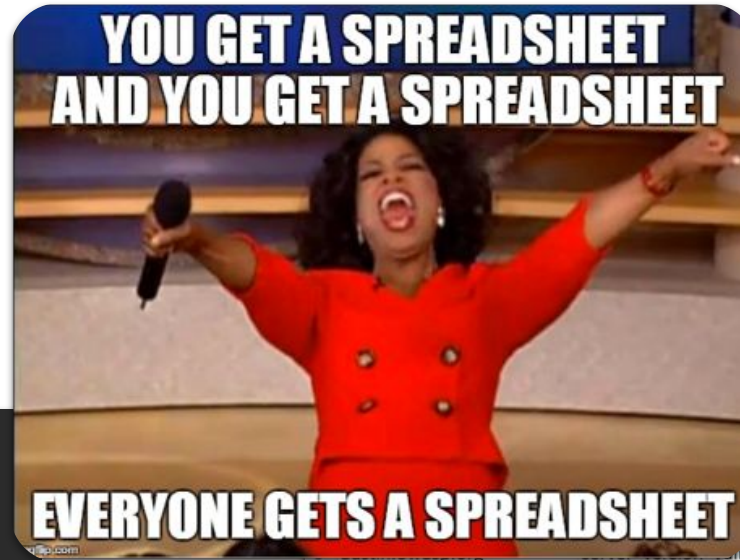
Nettacker

* API Key: d39b4dd429fe8a12c064821d8e543928
* Serving Flask app 'api.engine' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on https://127.0.0.1:5000/ (Press CTRL+C to quit)
```



# 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



```
OWASP Version 0.3.3 [TRENT]
github.com/OWASP
owasp.org
z3r0d4y.com
Nettacker

[2024-01-21 01:37:20][+] Nettacker engine started ...
[2024-01-21 01:37:20][+] 85 modules loaded ...
[2024-01-21 01:37:20][+] regrouping targets based on hardware resources!
[2024-01-21 01:37:20][+] Removing old database record for selected targets and modules.
[2024-01-21 01:37:20][+] imported 1 targets in 1 process(es).
[2024-01-21 01:37:20][+] process-1[ivanti_ics_cve_2023_46885_vuln] [module-thread 1/1]request-thread 1/2|allow_redirects: fal
gent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36 method: get ssl:
0 url: https://[redacted]:443/api/v1/configuration/users/user-roles/user-role/rest-userrole1/web-bookmarks/bookmark
success_condition(s):
conditions: content: true status_code: - '403'
[2024-01-21 01:37:20][+] building graph ...
[2024-01-21 01:37:20][+] finish building graph!
```

date	target	module_name	port	logs
2024-01-21 01:37:20.550545	[redacted]	ivanti_ics_cve_2023_46885_vuln	443	Detected

```
Software Details: OWASP Nettacker version 0.3.3 [TRENT] in 2024-01-21 01:37:20
[2024-01-21 01:37:20][+] report saved in /sec/root/venv/Nettacker/.data/results/results_2024_01_21_01_37_16_endmxcsczc.html and database
[2024-01-21 01:37:20][+] done!
```

PORT	DESCRIPTION	USERNAM	PASSWOI	TIME
80	AtlassianProxy/1.15.8.1			#####
443	AtlassianProxy/1.15.8.1			#####
443	cloudflare			#####
80	cloudflare			#####
80	cloudflare			#####
443	cloudflare			#####
443	cloudflare			#####
80	cloudflare			#####
443	nginx			#####
80	nginx			#####
443	cloudflare			#####
80	cloudflare			#####
443	cloudflare			#####
443	cloudflare			#####
80	cloudflare			#####
443	cloudflare			#####
80	cloudflare			#####
443	cloudflare			#####
443	cloudflare			#####
10	secureflag.owasp.org server_version_vuln			
11	secureflag.owasp.org server_version_vuln			
12	kerala.owasp.org server_version_vuln			
13	kerala.owasp.org server_version_vuln			
14	lists.owasp.org server_version_vuln			
15	lists.owasp.org server_version_vuln			
16	name-virt-host.owasp.org server_version_vuln			
17	name-virt-host.owasp.org server_version_vuln			
18	cheatsheetseries.owasp.org server_version_vuln			

# (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....



The graphic features the OWASP logo (a fly) and the text "OWASP® BECOME A MEMBER". It displays two membership options in circular callouts. The first callout, in a dark blue circle, lists "1 YEAR \$50.00" with sub-options: "Two Year \$95", "Lifetime \$500", and "Full-time Student \$20". The second callout, in a multi-colored circle, lists "1 YEAR \$20.00" with sub-options: "Two years \$35", "Lifetime \$200", and "Full-time Student \$8". A note on the right states "\* PRICES CAN VARY DEPENDING ON YOUR LOCATION". The URL "owasp.org/membership" is at the bottom left.

owasp.org/membership

# Secure Flag Training










secureflag.owasp.org/user/index.html#/exercises/paths

SecureFlag

Assigned Activities 🕒 Running Labs ▶ 🔔 S

- Dashboard
- Labs
- Learning Paths
- Tournaments
- Achievements
- Team
- Settings
- Help
- Log Out

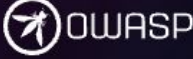
 <span>Not Completed</span> <b>Intermediate Secure Coding in Java</b> Number of Activities: <b>18</b> Level <span>●</span> <span>●</span> <span>●</span> Intermediate <a href="#">Browse Learning Path</a>	 <span>Not Completed</span> <b>Secure Authentication in Java</b> Number of Activities: <b>12</b> Level <span>●</span> <span>●</span> <span>●</span> Intermediate <a href="#">Browse Learning Path</a>	 <span>Not Completed</span> <b>OWASP API Security Top 10:2019 in Java</b> Number of Activities: <b>24</b> Level <span>●</span> <span>●</span> <span>●</span> Intermediate <a href="#">Browse Learning Path</a>
 <span>Not Completed</span> <b>Intermediate Secure Coding With Spring Boot</b> Number of Activities: <b>14</b> Level <span>●</span> <span>●</span> <span>●</span> Intermediate <a href="#">Browse Learning Path</a>	 <span>Not Completed</span> <b>Injections in Java</b> Number of Activities: <b>16</b> Level <span>●</span> <span>●</span> <span>●</span> Intermediate <a href="#">Browse Learning Path</a>	 <span>Not Completed</span> <b>Advanced Secure Coding in Java</b> Number of Activities: <b>18</b> Level <span>●</span> <span>●</span> <span>●</span> <span>●</span> Advanced <a href="#">Browse Learning Path</a>

powered by  SecureFlag


# Checkmarx Codebashing



← → ↻ [owasp.codebashing.com/app/login](https://owasp.codebashing.com/app/login)

Checkmarx 

## Welcome to



# CODEBASHING

Powered by Checkmarx

CodeBashing™ is redefining developer secure code training by bringing security and coding together fast. With bite-sized secure code workouts, we're empowering developers to secure code as they write it.

[Learn more about Codebashing and how it helps developers write secure code the first time.](#)

By clicking Login you agree to the [Codebashing Terms of Service](#) and [Privacy Policy](#)

### Login

Email Address

Password

[Forgot password?](#)

Login

# Codebashing Training Courses



In partnership with

Home | **Learn** | Play | Test | Resources | Search

← Back

0%  
0/18  
0

 1,000 5-8min LESSON SQL Injection CWE-...   A3: Inject...   SANS-...	 1,000 5-8min LESSON XXE Processing CWE-611   A5: Security Mi...	 1,000 5-8min LESSON Command Injection CWE-77   A3: Injection	 1,000 5-8min LESSON Session Fixation CWE-384   A7: Identificatio...	 1,000 5-8min LESSON Use of Insufficiently Ran... CWE-330   A1: Broken Acc...
 1,000 5-8min LESSON Reflected XSS CWE-...   A3: Inject...   SANS-...	 1,000 5-8min LESSON Stored (Persistent) XSS A3: Injection	 1,000 5-8min LESSON DOM XSS CWE-...   A3: Inject...   SANS-...	 1,000 5-8min LESSON Directory (Path) Traversal CWE...   A1: Broke...   SANS-...	 1,000 5-8min LESSON Privileged Interface Expo... CWE-265   A5: Security Mi...
 1,000 5-8min LESSON Leftover Debug Code CWE-489   A1: Broken Acc...	 1,000 5-8min LESSON Authentication Credentia... CWE-522   A4: Insecure D...	 1,000 5-8min LESSON Session Exposure Within ... CWE-613   A4: Insecure D...	 1,000 5-8min LESSON User Enumeration CWE-203   A4: Insecure D...	 1,000 5-8min LESSON Horizontal Privilege Escal... CWE-265   A1: Broken Acc...

[https://owasp.codebashing.com/courses/java/lessons/reflected\\_xss](https://owasp.codebashing.com/courses/java/lessons/reflected_xss)

# 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)

OWASP Membership Information x +

owasp.org/membership/

Would your business like to become a [Corporate Member](#)?

## Join or Renew Now

United Kingdom Postal Code

One Year \$50 Two Year \$95 Lifetime \$500

Join the OWASP Mailing List (See details below)

I am requesting [Complimentary Membership for OWASP Leaders](#)

# Open Source Project Appeal



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

=>OWASP Gives Your OSS Projects:

<b>A Home</b>	Move your Source Code repo to <a href="https://github.com/OWASP">GitHub.com/OWASP</a> organisation repository
<b>Visibility</b>	OWASP Projects webpage gets over 6 million visitors a year.
<b>Credibility</b>	OWASP is well known in the AppSec/DevSecOps community and the industry
<b>Resources</b>	Funding and Project Summits are available for qualifying Programs & Grants
<b>Community</b>	Our Conferences and Local Chapters connect OWASP Projects with many users & collaborators
<b>Longevity</b>	The OWASP Community can continue to collaborate and develop your project even if you no longer have time/ability to maintain the project yourself
<b>Neutrality</b>	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](https://owasp.org/projects) -> click [Start a New Project](#) link on the right ->

This creates a New Project Request JIRA ticket:

The screenshot shows a web browser window with the URL [owasporg.atlassian.net/servicedesk/customer/portal/7/create/70](https://owasporg.atlassian.net/servicedesk/customer/portal/7/create/70). The page title is "OWASP Services / Non-funding Request Service Desk". The main heading is "New Project Request". The form contains the following fields:

- Project Name\***: A text input field.
- Project Type\***: A dropdown menu with the following options: Code Project, Documentation Project, and Tool Project.
- Leader Name (First Last)\***: A text input field.
- Leader Email\***: A text input field.

## 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

Q 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 too 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**

