Building Secure Software With OWASP

OWASP
The Open Web Application Security Project

GERMAN OWASP DAY 2017
Essen, 14. November 2017
About Me

Martin Knobloch
+10 years developer experience
+10 years information security experience
Dutch OWASP Chapter Leader since 2007
OWASP AppSec-Eu/Research 2017 PC Chair
Storyteller @ xebia.com
Email: martin.knobloch@owasp.org
Twitter: @knoblochmartin
https://www.linkedin.com/in/martin-knobloch
Welcome to OWASP
the free and open application security community

About · Searching · Editing · New Article · OWASP Categories

OWASP Overview

The Open Web Application Security Project (OWASP) is dedicated to finding and fighting the causes of insecure software. Everything here is free and open source. The OWASP Foundation is a 501c3 not-for-profit charitable organization that ensures the ongoing availability and support for our work. Participation in OWASP is free and open to all.

- Join webappsec! The OWASP mail list
- Get Started
- Contact OWASP
- Become a Member

Featured Story

Announcing the OWASP Sprajax Project - the first AJAX Security Scanner

OWASP thanks Denim Group for the donation of Sprajax, an open source security scanner for AJAX-enabled applications. Sprajax, a Microsoft .Net-based application is the first web security scanner developed specifically to scan AJAX web applications for security vulnerabilities.

"Denim Group is committed to furthering the field of application security,“ said Dan Cornell, principal of Denim Group, “and by donating Sprajax to OWASP, we intend to generate more discussion around security.

OWASP Conferences

Register for OWASP AppSec Conference in Seattle Oct. 16-17

The Open Web Application Security Project

Join us for our 5th AppSec Conference October 16-18 in Seattle. Microsoft’s Michael Howard will be giving the keynote and you’ll hear presentations on topics like Web Services Security, PCI status, Securing AJAX, the Microsoft Secure Development Lifecycle, all new OWASP projects, and much more. Check the full agenda online.

OWASP is a not-for-profit, and the OWASP AppSec Conference is an incredible bargain ($450, $400 for OWASP members, and $250 for students). You can attend one of 3 full-day training sessions on the 16th, and the main conference is two full days of presentations, panel discussion on the 17th and 18th. You can read all the details then register online.

OWASP Community (add)
A1 - Injection
A2 - Broken Authentication and Session Management
A3 - Cross Site Scripting (XSS)
A4 - Insecure Direct Object References
A5 - Security Misconfiguration
A6 - Sensitive Data Exposure
A7 - Missing Function Level Access Control
A8 - Cross-Site Request Forgery
A9 - Using Components with known Vulnerabilities
A10 - Unvalidated Redirects and Forwards

..but
Verify for Security Early and Often
Parameterize Queries
Encode Data
Validate All Inputs
Implement Identity and Authentication Controls
Implement Appropriate Access Controls
Protect data
Implement Logging and Intrusion Detection
Leverage Security Frameworks and Libraries
Error and Exception Handling

https://www.owasp.org/index.php/OWASP_Proactive Controls
open SAMM

OWASP
The Open Web Application Security Project

Governance
- Strategy & Metrics
- Policy & Compliance
- Education & Guidance

Construction
- Threat Assessment
- Security Requirements
- Secure Architecture

Verification
- Design Review
- Code Review
- Security Testing

Deployment
- Vulnerability Management
- Environment Hardening
- Operational Enablement

https://www.owasp.org/index.php/OWASP_SAMM_Project
Continuous development

OWASP
The Open Web Application Security Project

git

Gradle

Maven

sonarqube
Rugged DevOps - AppSec Pipeline Template

OWASP
The Open Web Application Security Project

https://www.owasp.org/index.php/OWASP_AppSec_Pipeline
https://www.appsecpipeline.org/index.html
Intake Tools:
The first stage of an AppSec Pipeline which handles inbound requests of the AppSec program. These can be new apps, existing apps that have never been assessed, apps which have been assessed before or retesting of previous security findings. These tools aim to tame the inflow of work into the AppSec Pipeline.

https://www.appsecpipeline.org/index.html
Triage Tools:
The second stage of an AppSec Pipeline which prioritizes inbound requests and assesses their testing needs based on the risk level. The more risky the app, the more activities are assigned. These tools aim to provide automation and orchestration to reduce the startup time of the testing stage.

https://www.appsecpipeline.org/index.html
Test Tools:
The forth and final stage of an AppSec Pipeline which collects and normalizes the data created during testing. Any duplicate findings should be removed so that the same issue found by multiple tools is only reported once. Here we link to issue tracking systems, produce reports, and otherwise provide data for stakeholders.

https://www.appsecpipeline.org/index.html
Delivery Tools:
The third stage of an AppSec Pipeline which runs one or more tests in parallel to assess the security posture of an application. Ideally, these testing or at least their setup should be automated. Priority should be given to tools that can be run programmatically and produce results with few false positives.

https://www.appsecpipeline.org/index.html
ASVS defines detailed verification requirements for levels 1 and above; whereas level 0 is meant to be flexible and is customized by each organization.
V1: Architecture, design and threat modeling
V2: Authentication Verification Requirements
V3: Session Management Verification Requirements
V4: Access Control Verification Requirements
V5: Malicious input handling verification requirements
V7: Cryptography at rest verification requirements
V8: Error handling and logging verification requirements
V9: Data protection verification requirements
V10: Communications security verification requirements
V11: HTTP security configuration verification requirements
V13: Malicious controls verification requirements
V15: Business logic verification requirements
V16: Files and resources verification requirements
V17: Mobile verification requirements
V18: Web services verification requirements
V19. Configuration

https://www.owasp.org/index.php/OWASP_Application_Security_Verification_Standard_Project
Security RAT (Requirement Automation Tool) is a tool supposed to assist with the problem of addressing security requirements during application development. 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
https://www.owasp.org/index.php/OWASP_Threat_Dragon
### OWASP Cheat Sheets

#### Cheat Sheets
- 3rd Party Javascript Management
- Access Control
- AJAX Security Cheat Sheet
- Authentication (ES)
- Bean Validation Cheat Sheet
- Choosing and Using Security Questions
- Clickjacking Defense
- C-Based Toolchain Hardening
- Cross-Site Request Forgery (CSRF) Prevention
- Cryptographic Storage
- Deserialization
- DOM based XSS Prevention
- Forgot Password
- HTML5 Security
- HTTP Strict Transport Security
- Injection Prevention Cheat Sheet
- Input Validation
- JAAS
- LDAP Injection Prevention
- Logging
- Mass Assignment Cheat Sheet
- .NET Security
- OWASP Top Ten
- Password Storage
- Pinning
- Query Parameterization
- Ruby on Rails
- REST Security
- Session Management
- SAML Security
- SQL Injection Prevention
- Transaction Authorization
- Transport Layer Protection
- Unvalidated Redirects and Forwards
- User Privacy Protection
- Web Service Security
- XSS (Cross Site Scripting) Prevention
- XML External Entity (XXE) Prevention Cheat Sheet

#### Developer / Builder
- Attack Surface Analysis
- XSS Filter Evasion
- REST Assessment
- Web Application Security Testing

#### Mobile
- Android Testing
- iOS Developer
- Mobile Jailbreaking

#### OpSec / Defender
- Application Security Architecture
- Business Logic Security
- Command Injection Defense Cheat Sheet
- Credential Stuffing Prevention Cheat Sheet
- PHP Security
- Regular Expression Security Cheat Sheet
- Secure Coding
- Secure S0LCE
- Threat Modeling
- Grails Secure Code Review
- IOS Application Security Testing
- Key Management
- Insecure Direct Object Reference Prevention
- Content Security Policy

[https://www.owasp.org/index.php/OWASP_Cheat_Sheet_Series](https://www.owasp.org/index.php/OWASP_Cheat_Sheet_Series)
The security knowledge framework is here to support developers create secure applications. By analysing processing techniques in which the developers use to edit their data the application can link these techniques to different known vulnerabilities and give the developer feedback regarding descriptions and solutions on how to properly implement these techniques in a safe manner.
Dependency-Check is a utility that attempts to detect publicly disclosed vulnerabilities contained within project dependencies. 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 entries.

https://www.owasp.org/index.php/OWASP_Dependency_Check
Defect Dojo

OWASP
The Open Web Application Security Project

DEFECT dojo

Dashboard for Bobby Tables

Historical Finding Severity

Reported Finding Severity by Month

https://www.owasp.org/index.php/OWASP_DefectDojo_Project
OWASP & AppSec Pipelines

1) Pull Project Code
2) Launch Docker(s)
3) Push Findings

Weaponized Jenkins
Project Repo

Tests run in Docker containers

Source of Truth
Validated Findings
Optional Notifications

Project Jira
Project Slack Channel

Defect Dojo

OWASP Projects
AppSec Pipeline
OWASP Testing Guide

Information Gathering

Configuration and Deploy Management Testing

Identity Management Testing

Authentication Testing

Authorization Testing

Session Management Testing

Input Validation Testing

Error Handling

Cryptography

Business Logic Testing

Client Side Testing

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>O-Shepherd</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Attack surface analysis: <a href="#">O-Cheat sheets</a> on the topic</td>
<td>Static code analysis to detect vulnerabilities:</td>
<td>Fuzz testing &amp; Penetration testing (ZAP)</td>
<td>Correct any security issues</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><a href="#">SCA tools</a> <a href="#">OWASP review guide</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Controls:</td>
<td></td>
<td></td>
<td></td>
<td>Some controls in requirements form, esp <a href="#">ASVS</a></td>
<td>Some controls in implementation + O-Coding libraries (eg HTML sanitizer)</td>
<td>Some controls to verify (test, review) esp. <a href="#">O-Testing guide</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><a href="#">O-Cheat sheets</a></td>
<td><a href="#">SKF</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><a href="#">O-SKFs</a> [ASVS]</td>
<td><a href="#">O-Coding libraries</a> <a href="#">O-Testing guide</a></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Work in progress**