A baker’s dozen: application security on a limited budget

About Chris Romeo

SECURITY BACKGROUND

- CEO / Co-Founder @ Security Journey
- 22 years in the security world, CISSP, CSSLP
  - *10 years at Cisco, leading security education.*
- Co-Lead of the OWASP Triangle Chapter

LISTEN TO ME

The Application
Security Podcast

TALK TO ME

@edgeroute

@AppSecPodcast

Security Journey®

Agenda

1. Traditional application security programs
2. The importance of security community
3. Building a program based on OWASP
   - Awareness and education
   - Process and measurement
   - Tools
4. Final thoughts

Traditional AppSec programs

PEOPLE

PROCESS

TOOLS

Security Journey®

Goals of an AppSec Program

GOAL
1

Limit vulnerabilities in deployed code.

GOAL
2

Build secure software and teach developers to build secure software.

GOAL
3

Provide processes and tools for AppSec standardization.

GOAL
4

Demonstrate software security maturity through metrics and assessment.

What if I had to develop an application security program with a budget of $0?

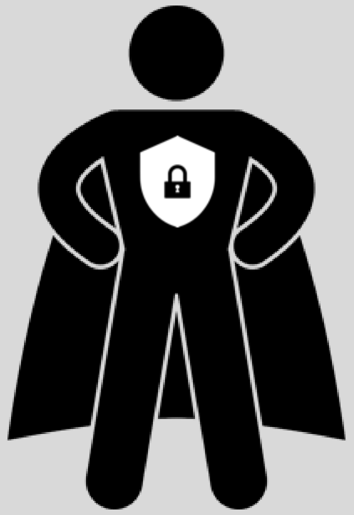
Enhance with OWASP Resources

Fill in missing areas of your program

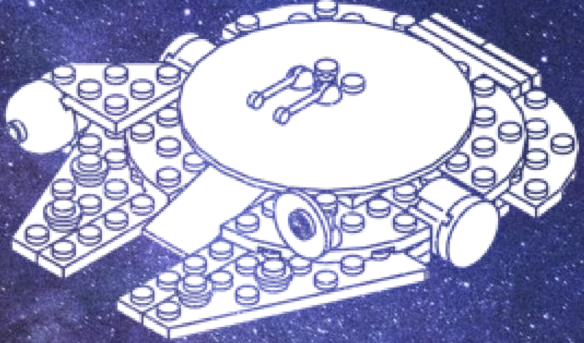
Security Champions

se · cu · ri · ty cham · pi · on [*sih · kyer · uh · tee cham · pee · uhn*], noun 1 a person passionate about security with a desire to educate those around them.

we all want to embed security champions in our companies.



OWASP



FLAGSHIP
PROJECTS
18

LAB
PROJECTS
24

INCUBATOR
PROJECTS
73

As of 6 September, 2019

Scale of project risk

Rating	Explanation
0	The only way this goes away is if owasp.org disappears off the Internet
1-3	Stable project, multiple releases, high likelihood of sustainability
4-6	Newer project, fewer releases
7-9	Older project with a lack of updates within the last year
10	If I added one of these to this project, I should have my head examined

NOTICE

Use OWASP projects with caution. There is no guarantee that a project will ever be updated again.

Security Journey®

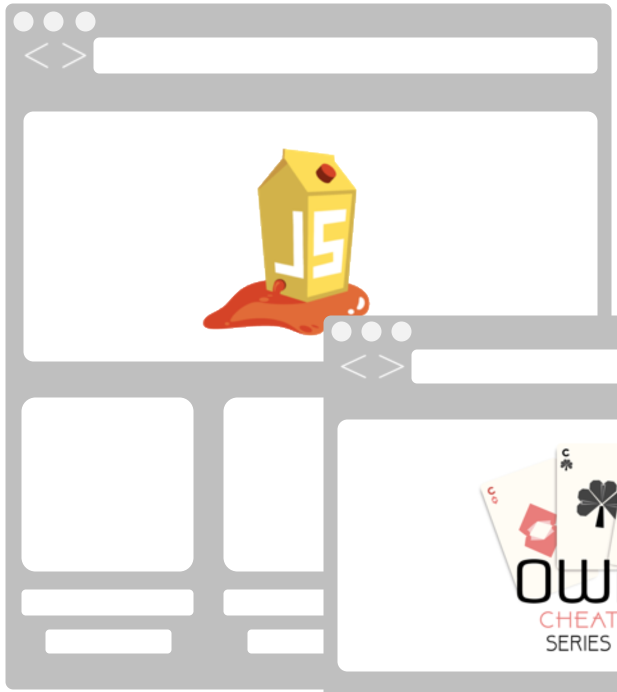
The categories

Awareness, knowledge, and education

Process and measurement

Tools

Awareness, knowledge and education



Security Journey®

Project Risk
0

A1:2017-Injection
A2:2017-Broken Authentication
A3:2017-Sensitive Data Exposure
A4:2017-XML External Entities (XXE)
A5:2017-Broken Access Control
A6:2017-Security Misconfiguration
A7:2017-Cross-Site Scripting (XSS)
A8:2017-Insecure Deserialization
A9:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring

https://owasp.org/www-project-top-ten/

Project Risk
2

C1 Define Security Requirements

C2 Leverage Security Frameworks and Libraries

C3 Secure Database Access

C4 Encode and Escape Data

C5 Validate All Imputs

C6 Implement Digital Identity

C7 Enforce Access Control

C8 Protect Data Everywhere

C9 Implement Security Logging and Monitoring

C10 Handle All Errors and Exceptions

https://owasp.org/www-project-proactive-controls/

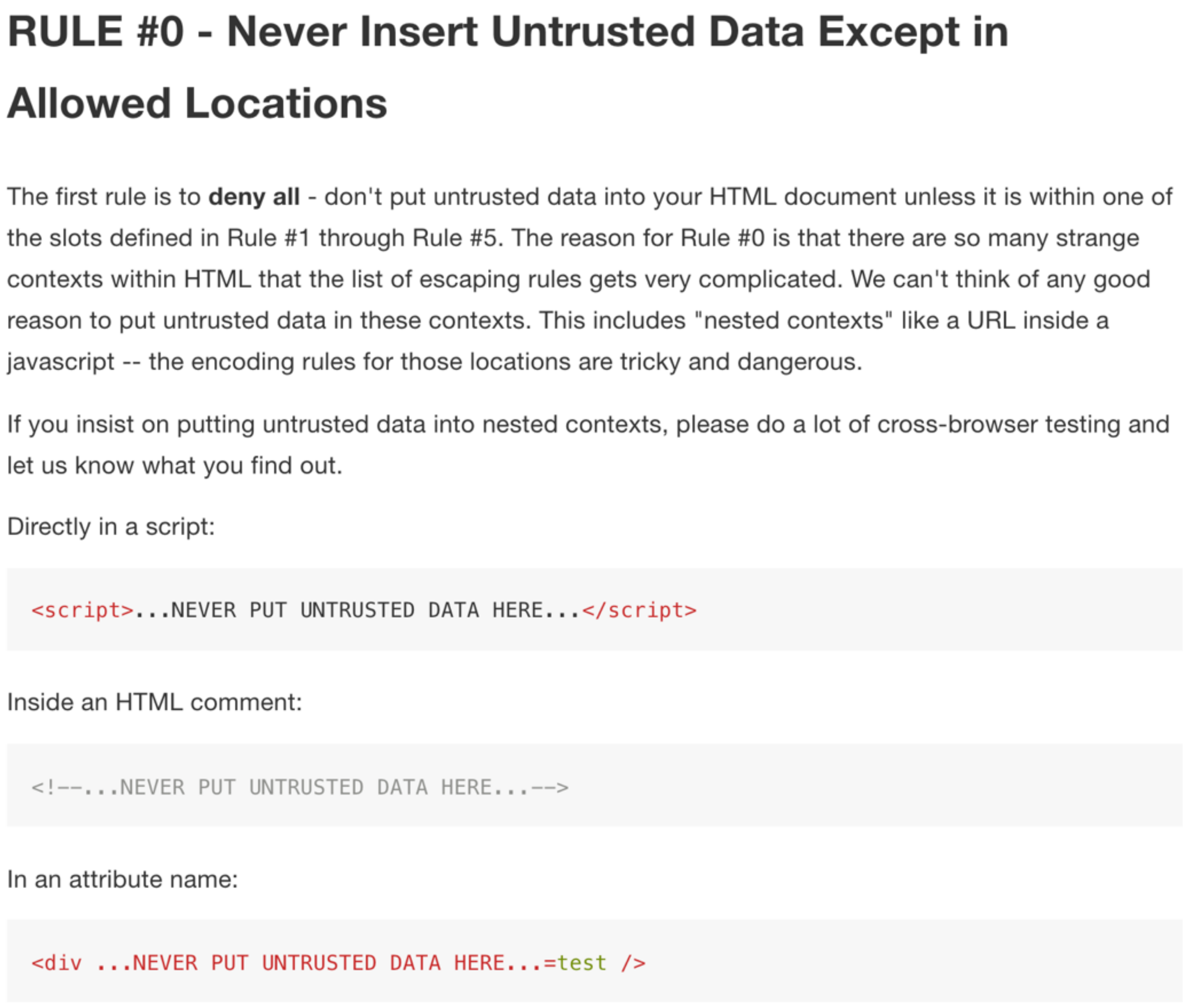
Security Journey®

The intermingling

OWASP Top 10 - 2017	OWASP ProActive CONTROLS
A1:2017-Injection	**C4 Encode and Escape Data, C5 Validate All Inputs**
A2:2017-Broken Authentication	**C6 Implement Digital Identity**
A3:2017-Sensitive Data Exposure	**C8 Protect Data Everywhere**
A4:2017-XML External Entities (XXE)	**C5 Validate All Inputs**
A5:2017-Broken Access Control	**C7 Enforce Access Control**
A6:2017-Security Misconfiguration	**None**
A7:2017-Cross-Site Scripting (XSS)	**C4 Encode and Escape Data, C5 Validate All Inputs**
A8:2017-Insecure Deserialization	**C5 Validate All Inputs**
A9:2017-Using Components with Known Vulnerabilities	**C2 Leverage Security Frameworks and Libraries**
A10:2017-Insufficient Logging & Monitoring	**C9 Implement Security Logging and Monitoring**

Project Risk
2

Cross Site Scripting Prevention



RULE #0 - Never Insert Untrusted Data Except in Allowed Locations

The first rule is to **deny all** - don't put untrusted data into your HTML document unless it is within one of the slots defined in Rule #1 through Rule #5. The reason for Rule #0 is that there are so many strange contexts within HTML that the list of escaping rules gets very complicated. We can't think of any good reason to put untrusted data in these contexts. This includes "nested contexts" like a URL inside a javascript -- the encoding rules for those locations are tricky and dangerous.

If you insist on putting untrusted data into nested contexts, please do a lot of cross-browser testing and let us know what you find out.

Directly in a script:

```
<script>...NEVER PUT UNTRUSTED DATA HERE...</script>
```

Inside an HTML comment:

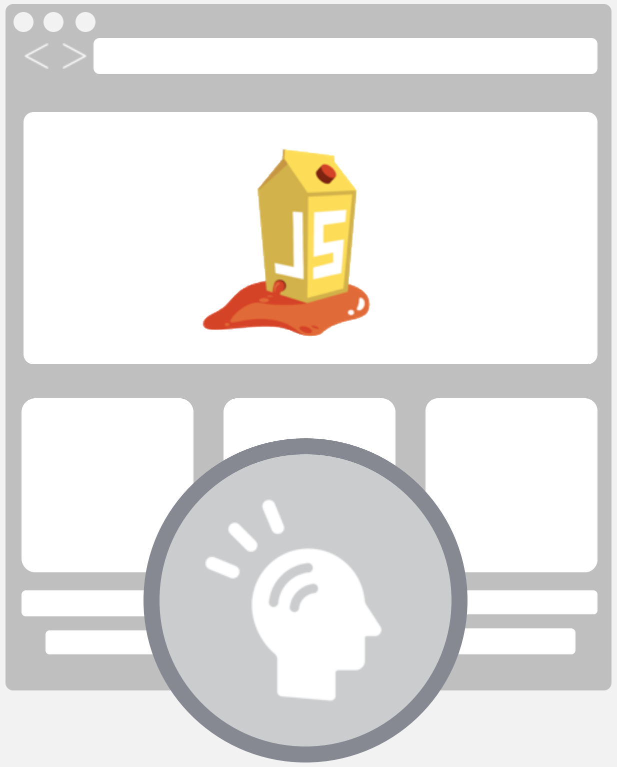
```
<!--...NEVER PUT UNTRUSTED DATA HERE...-->
```

In an attribute name:

```
<div ...NEVER PUT UNTRUSTED DATA HERE...=test />
```

https://cheatsheetseries.owasp.org/

Security Journey®



Project Risk
3

JavaScript-based

Intentionally insecure web app

Encompasses the entire OWASP Top Ten and other severe security flaws

https://owasp.org/www-project-juice-shop/

Security Journey®

Missing pieces in awareness, knowledge and education

Delivery of awareness and education

Administration of the training platforms

Awareness and education: impact and headcount

Awareness

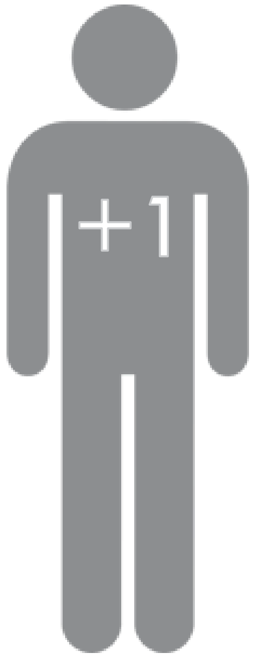
Foundational understanding of the most important concepts in AppSec

Knowledge

A concise reference for solving the most difficult AppSec problems

Hands-on training

Assimilation of key concepts through activities that lock in knowledge and make it practical



Awareness and education: getting started

Awareness

Lunch and learn sessions to teach the basics of all awareness documents

Knowledge

Teach developers about available cheat sheets

Host an internal copy of the cheat sheets

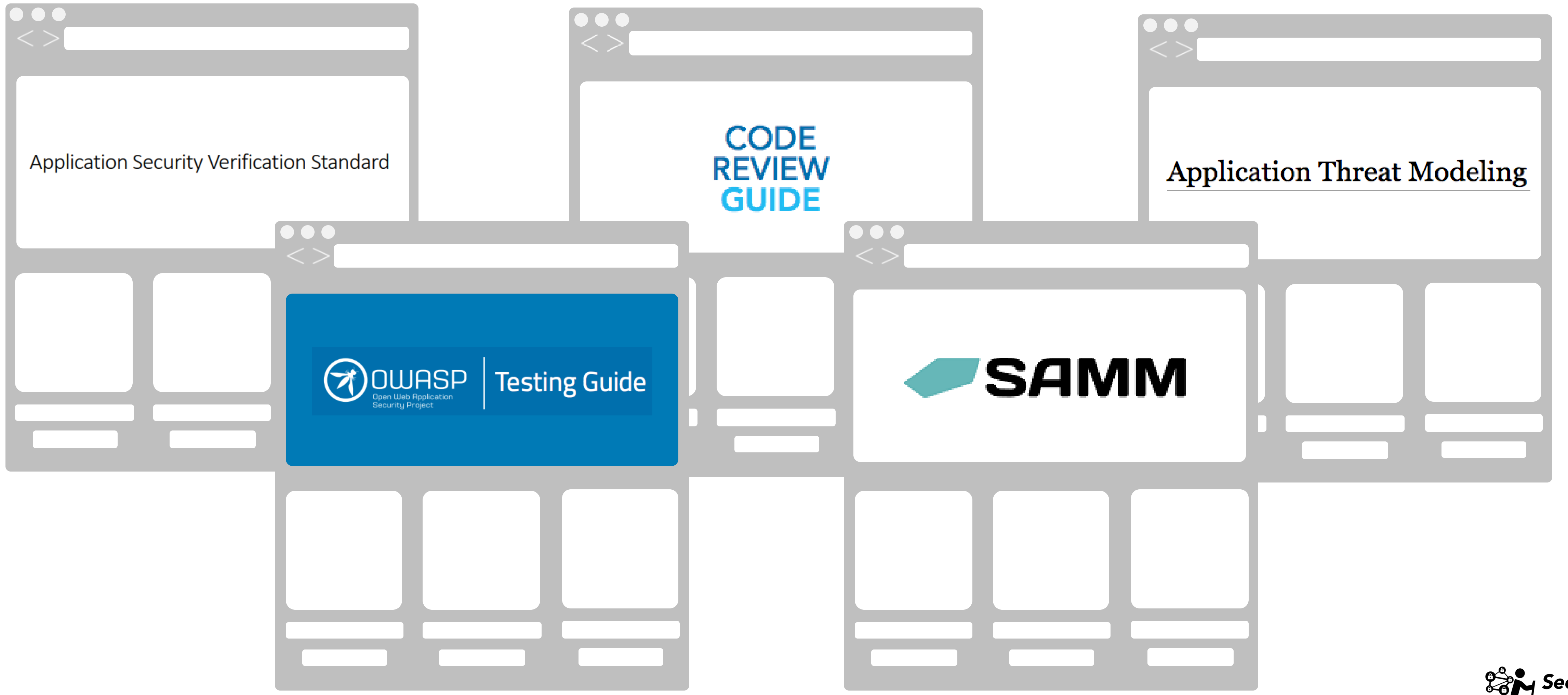
Lead a training session covering the three most crucial cheat sheets for your organization

Hands-on Training

Build an environment that hosts JuiceShop

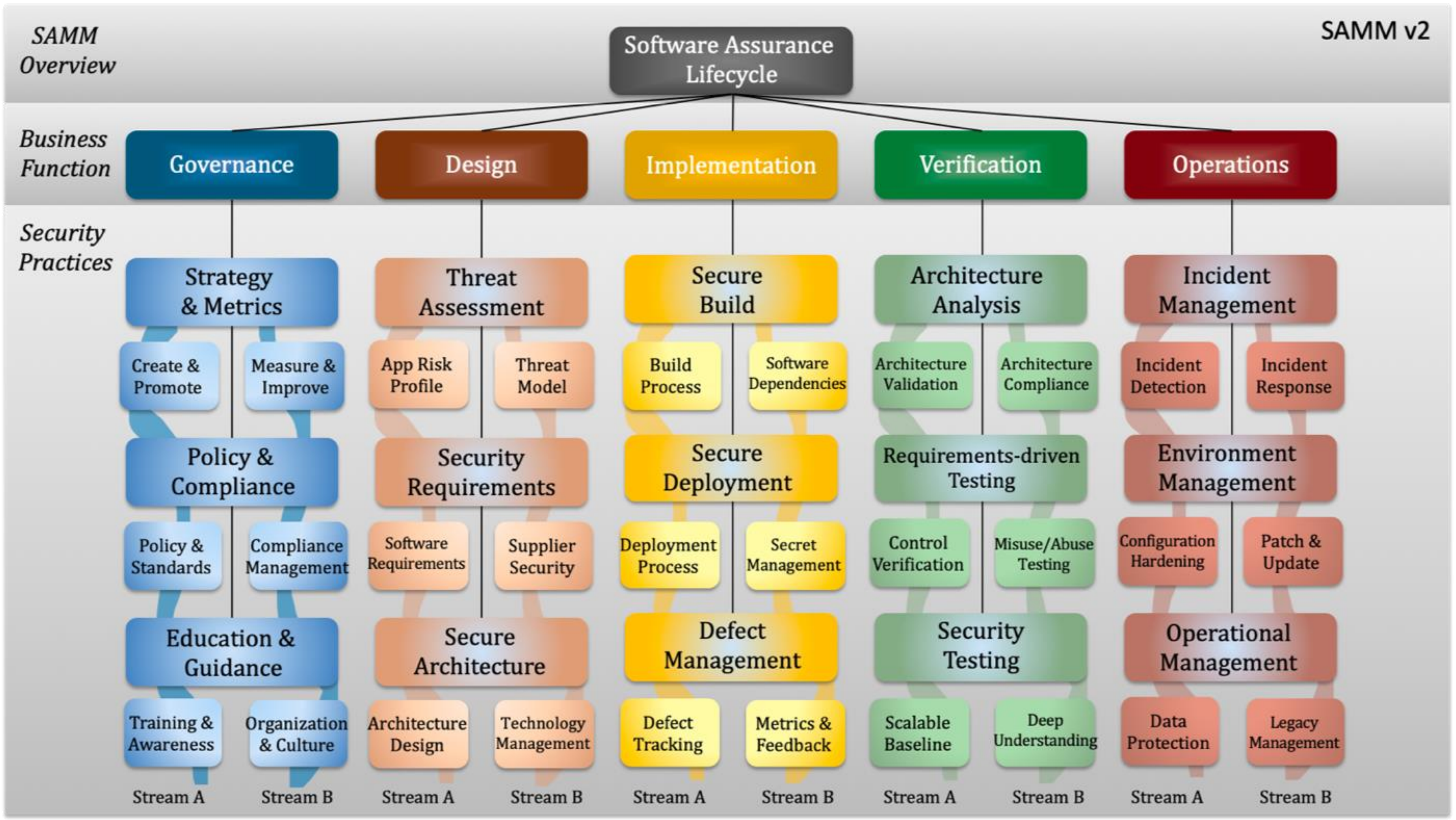
Schedule a hack-a-thon where teams gather and work on JuiceShop in teams and learn from each other

Process and Measurement


Application Security Verification Standard
CODE REVIEW GUIDE
Application Threat Modeling
OWASP Open Web Application Security Project
Testing Guide
SAMM

SAMM

Project Risk
1



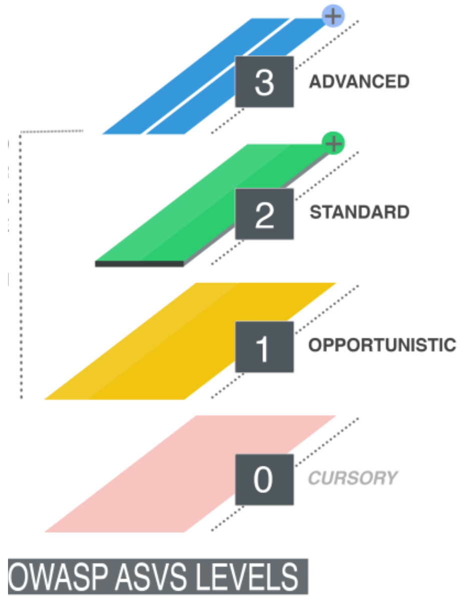
https://owasp.org/www-project-samm/

Security Journey®

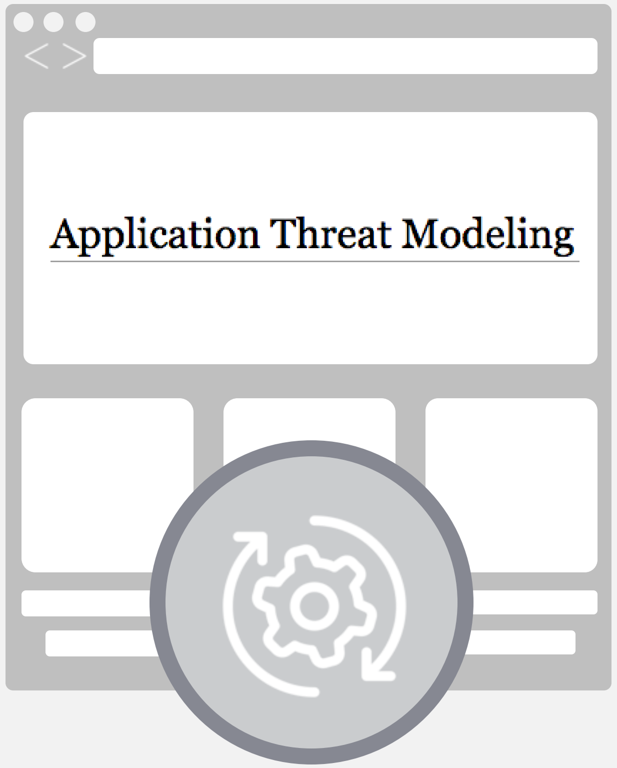
Application Security Verification Standard

Project Risk 1

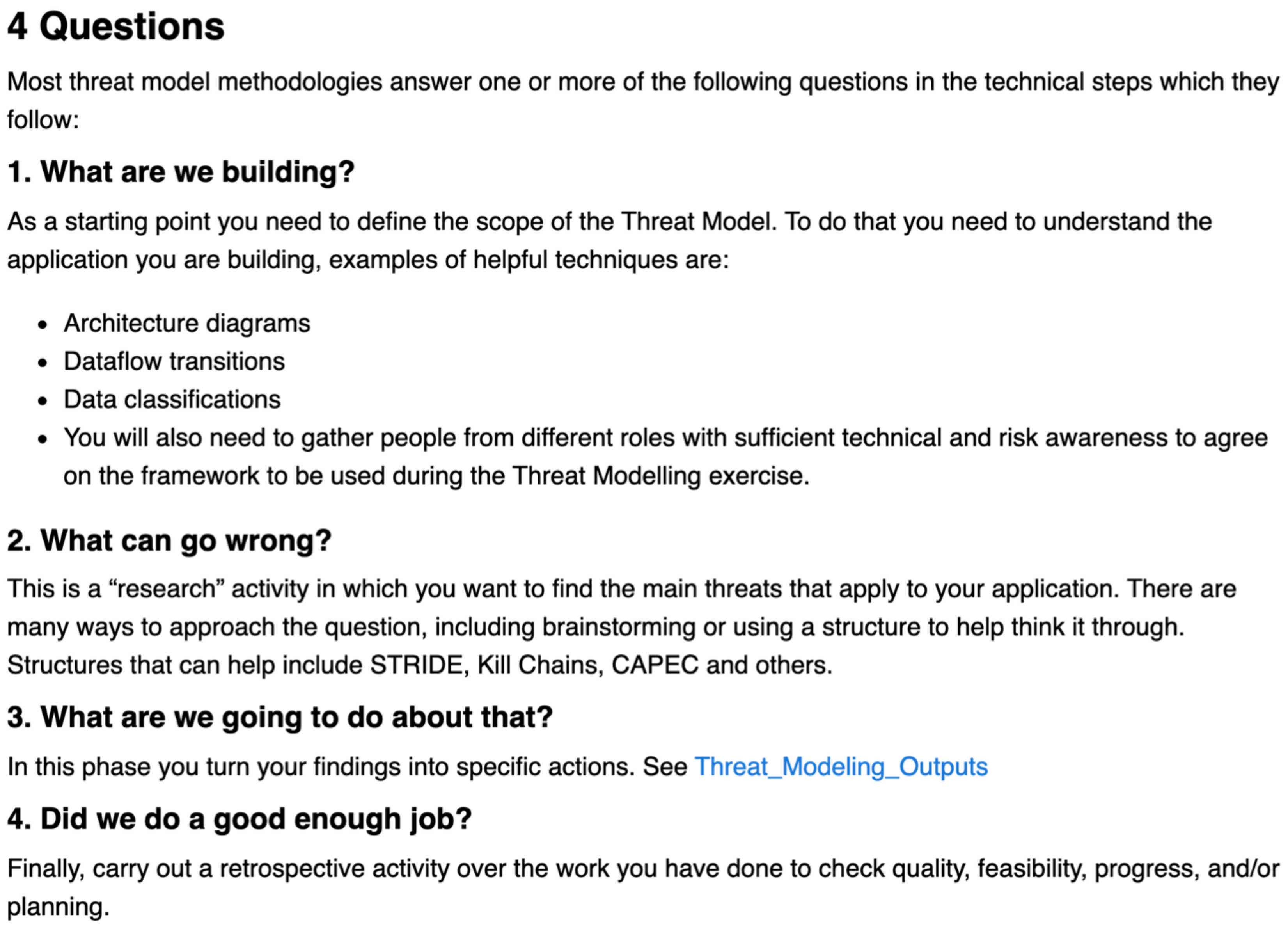
Requirement	
V1. Architecture, design and threat modelling	V11. HTTP security configuration
V2. Authentication	V13. Malicious controls
V3. Session management	V15. Business logic
V4. Access control	V16. File and resources
V5. Malicious input handling	V17. Mobile
V7. Cryptography at rest	V18. Web services
V8. Error handling and logging	V19. Configuration
V9. Data protection	V11. HTTP security configuration
V10. Communications	



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



Project Risk
5



4 Questions

Most threat model methodologies answer one or more of the following questions in the technical steps which they follow:

1. What are we building?

As a starting point you need to define the scope of the Threat Model. To do that you need to understand the application you are building, examples of helpful techniques are:

- Architecture diagrams
- Dataflow transitions
- Data classifications
- You will also need to gather people from different roles with sufficient technical and risk awareness to agree on the framework to be used during the Threat Modelling exercise.

2. What can go wrong?

This is a "research" activity in which you want to find the main threats that apply to your application. There are many ways to approach the question, including brainstorming or using a structure to help think it through. Structures that can help include STRIDE, Kill Chains, CAPEC and others.

3. What are we going to do about that?

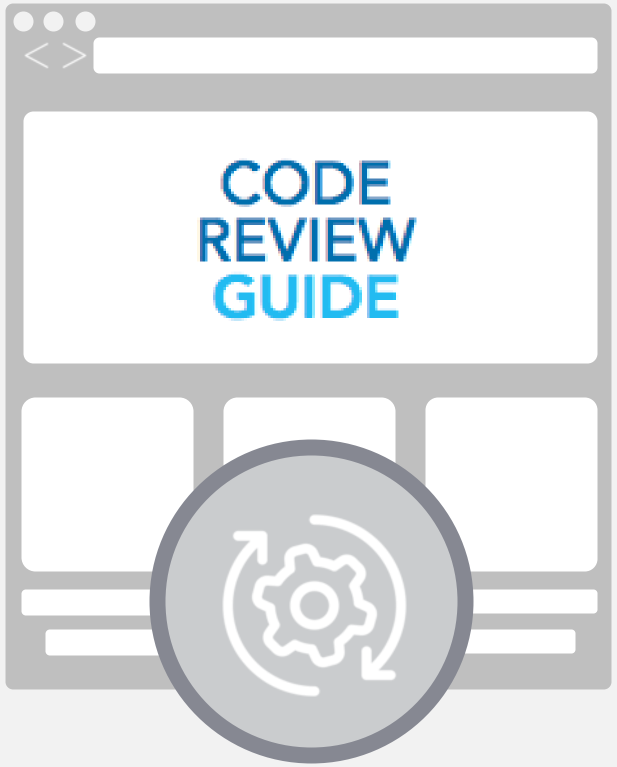
In this phase you turn your findings into specific actions. See Threat_Modeling_Outputs

4. Did we do a good enough job?

Finally, carry out a retrospective activity over the work you have done to check quality, feasibility, progress, and/or planning.

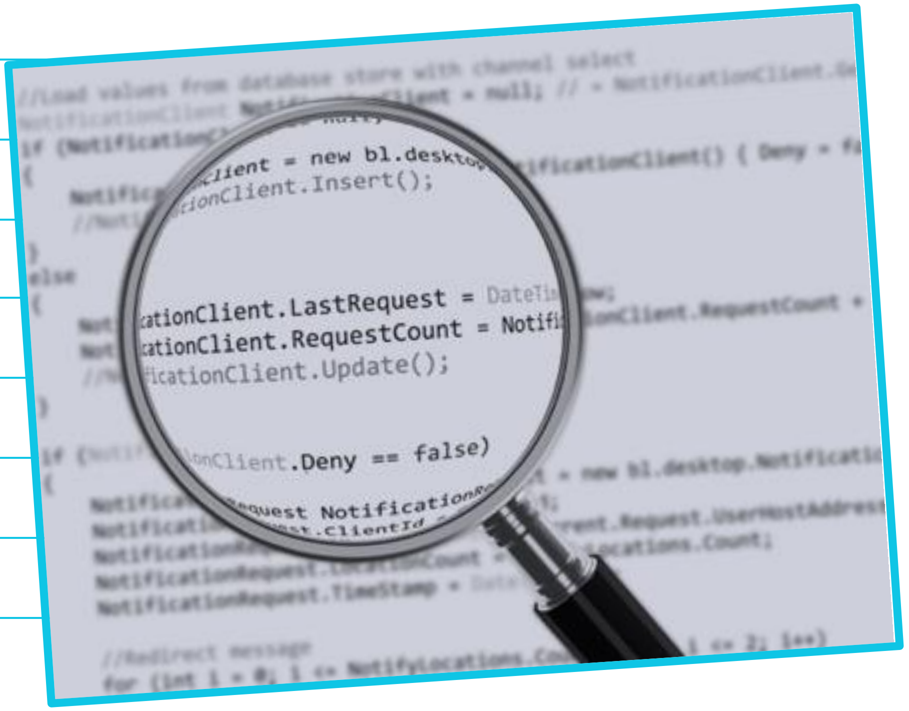
https://www.owasp.org/index.php/Application_Threat_Modeling

Security Journey®



Project Risk
4

Secure code review methodology
Technical reference for secure code review: OWASP Top 10
HTML5
Same origin policy
Reviewing logging code
Error handling
Buffer overruns
Client-side JavaScript
Code review do's and don'ts
Code review checklist
Code crawling

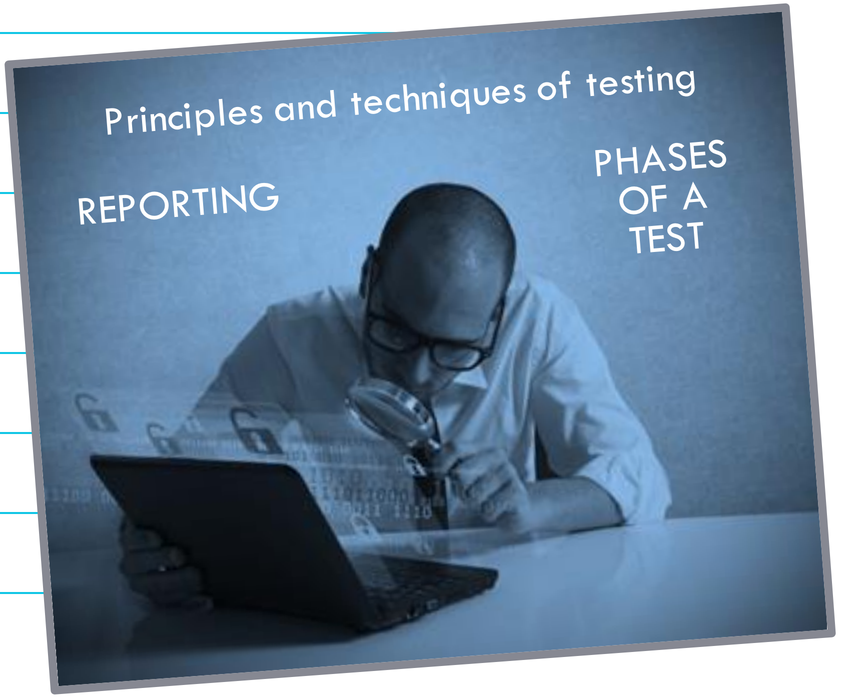


https://www.owasp.org/index.php/Category:OWASP_Code_Review_Project

OWASP Open Web Application Security Project | Testing Guide

Project Risk
1

- Information gathering
- Configuration and deployment management testing
- Identity management testing
- Authentication testing
- Authorization testing
- Session management testing
- Input validation testing
- Testing for error handling
- Testing for weak crypto
- Business logic testing
- Client-side testing



https://owasp.org/www-project-web-security-testing-guide/

Security Journey®

Missing pieces in process and measurement

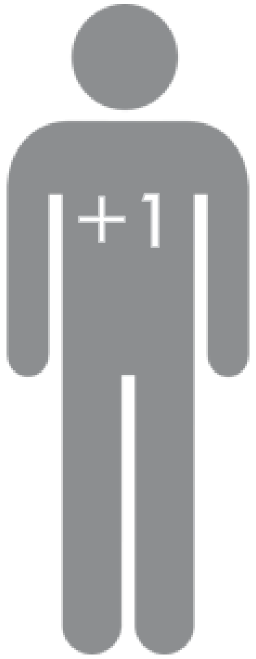
End-to end SDL or Secure SDLC

Program metrics

Deployment advice/experience on how to be successful

Security Journey®

Process and measurement: impact and headcount



Process

ASVS provides important requirements

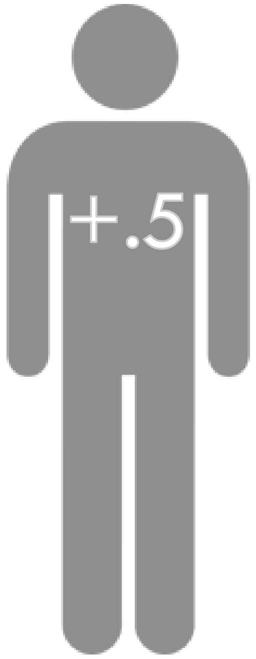
App threat modeling defines the process with examples

Code review guide describes how to perform a code review and what to look for

Testing guide provides how to test and a knowledge base of how to exploit vulnerabilities

Measurement

A roadmap to where you are today, and a plan for where you want to go with your AppSec program



Process and measurement: getting started

Process

Choose one of the process areas to start with (threat modeling) and build out this activity as your first

Early wins are key!

Measurement

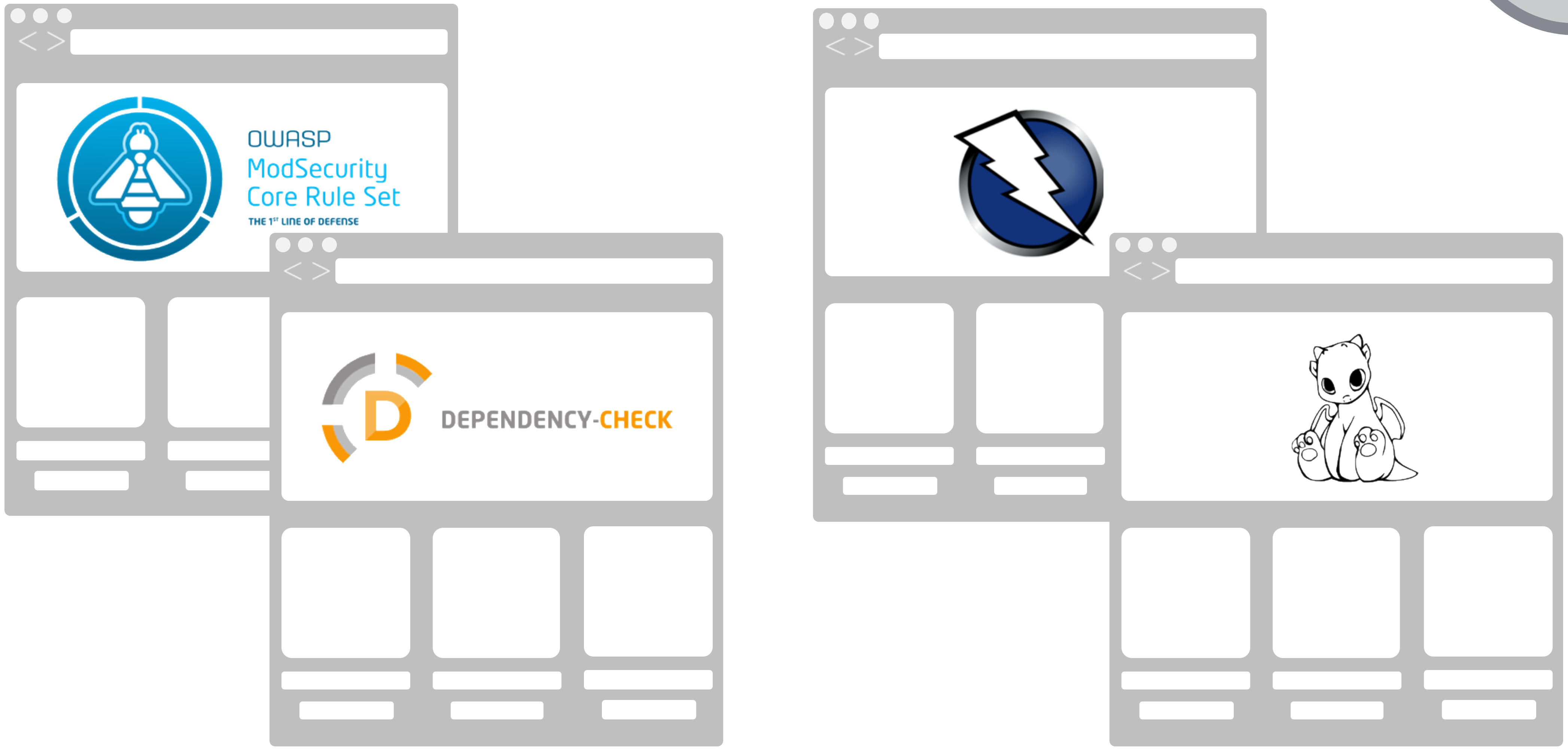
Perform an early assessment to determine where you are

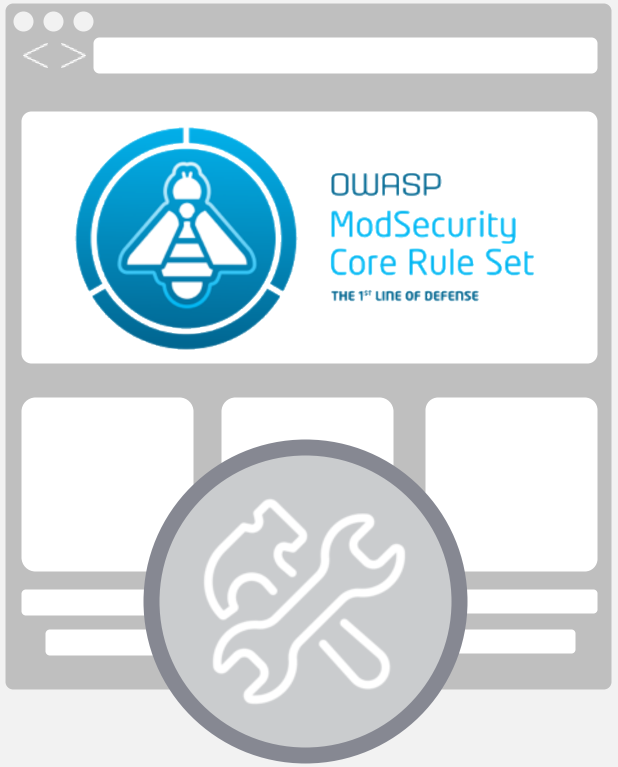
Map out your future
Share these assessments with Executives and Security Champions (and anyone else that will listen)

Advocate for Executive support on your plan to build a stronger AppSec program

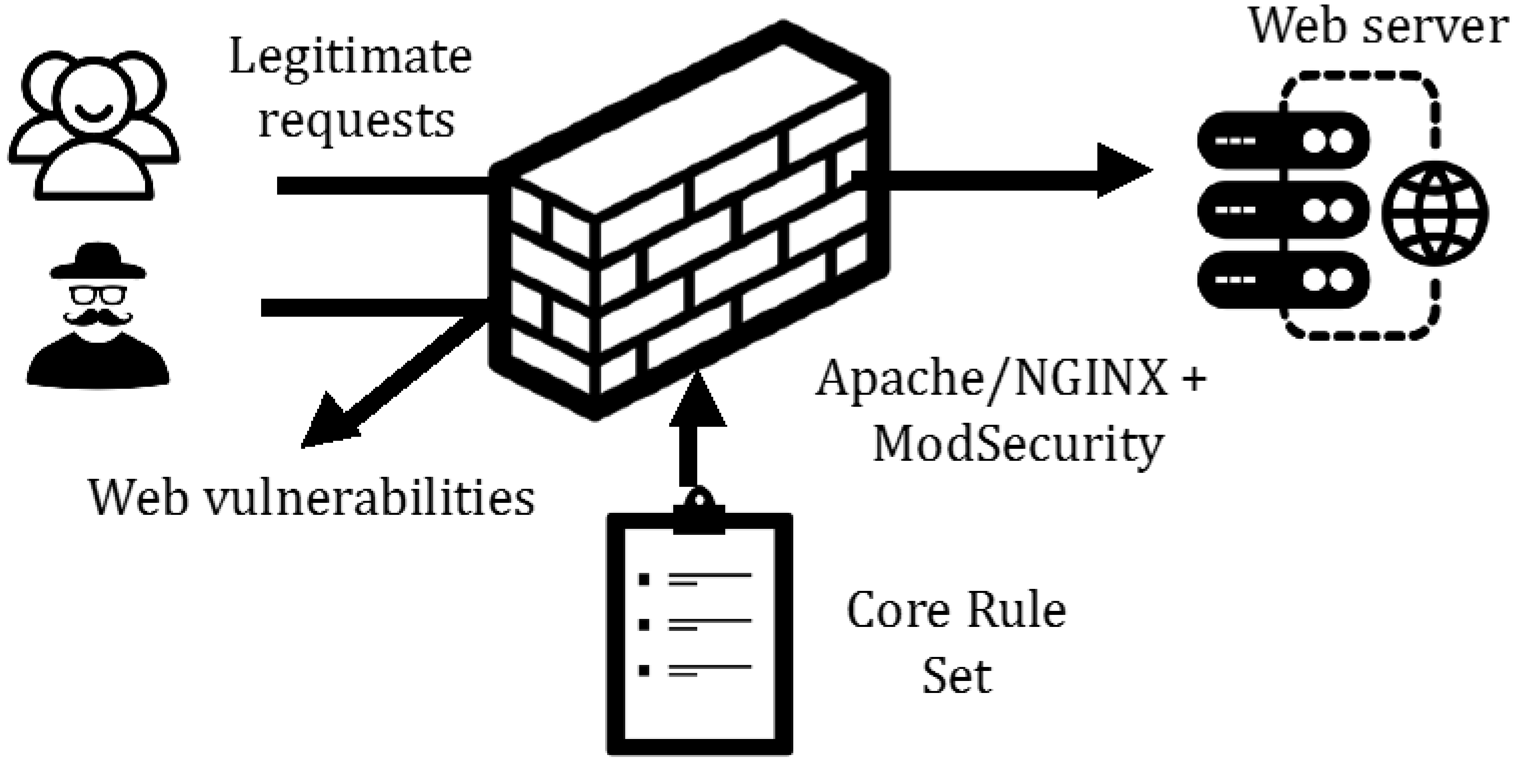
Security Journey®

Tools


OWASP
ModSecurity
Core Rule Set
THE 1ST LINE OF DEFENSE
DEPENDENCY-CHECK

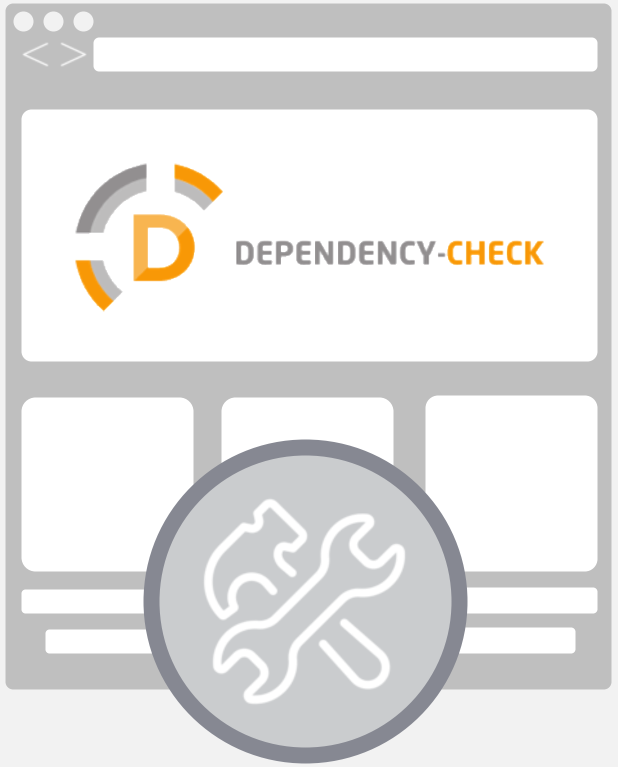


Project Risk 1

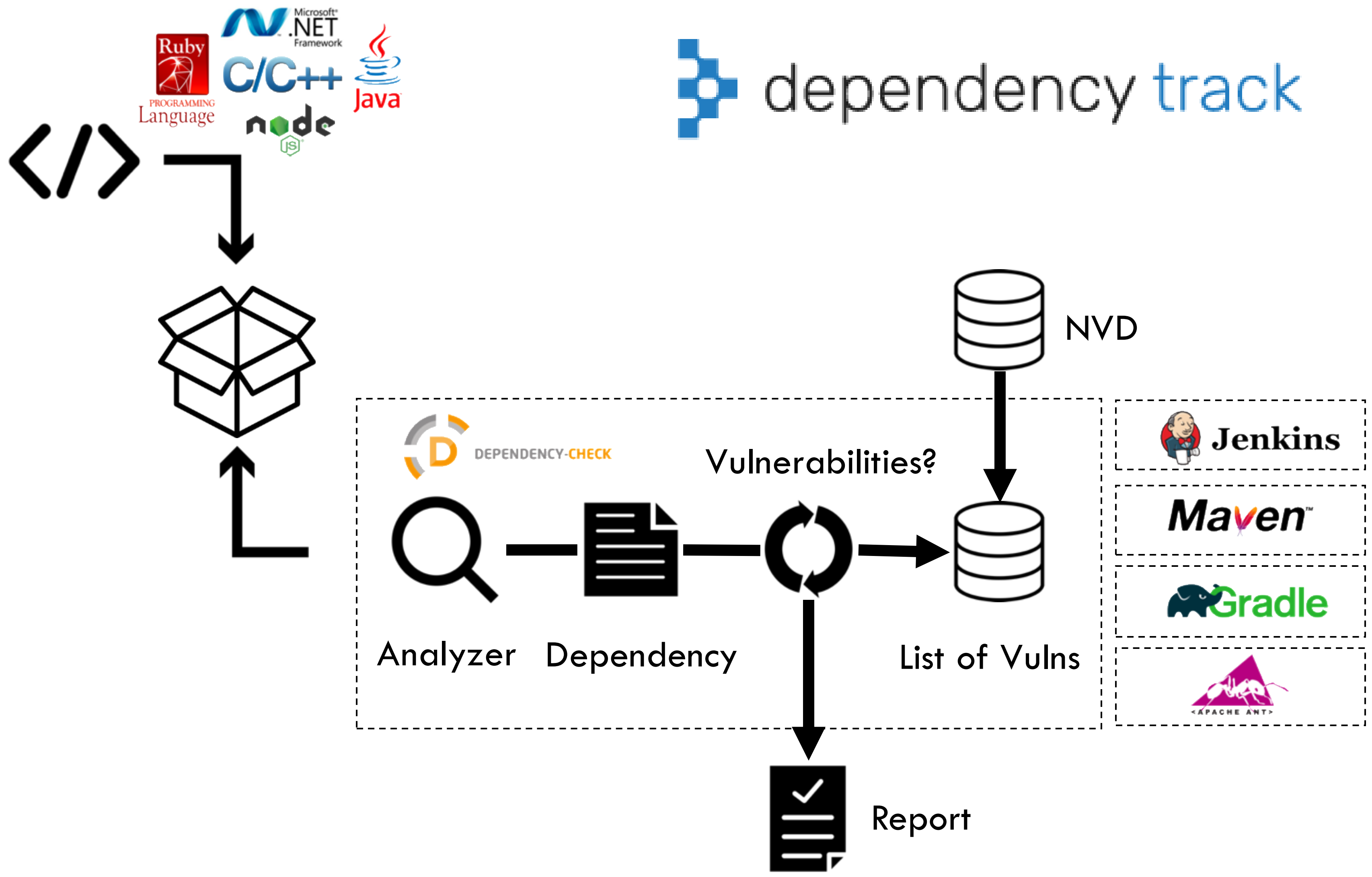


https://owasp.org/www-project-modsecurity-core-rule-set/

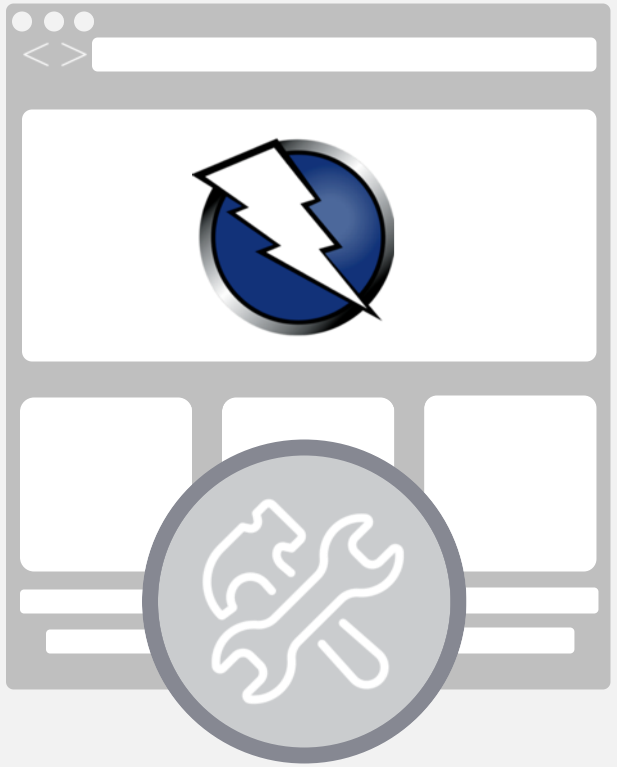
Security Journey®



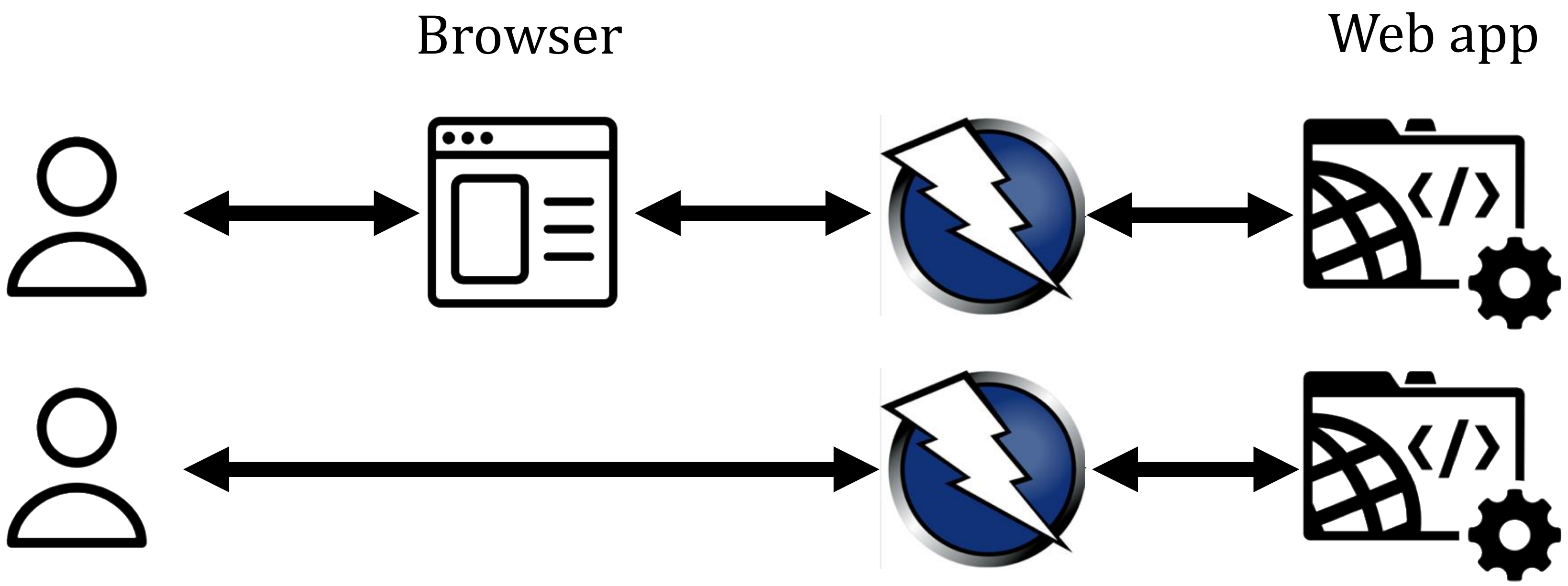
Project Risk
3



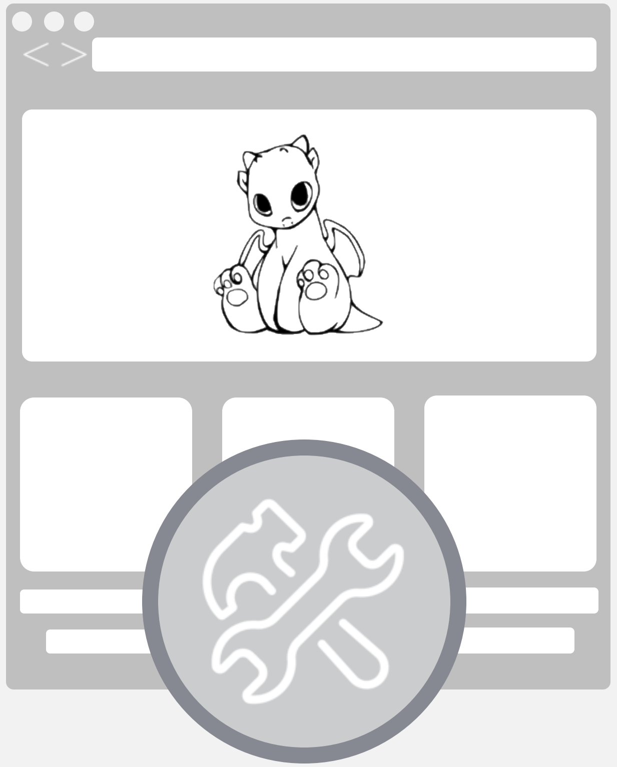
https://owasp.org/www-project-dependency-check/



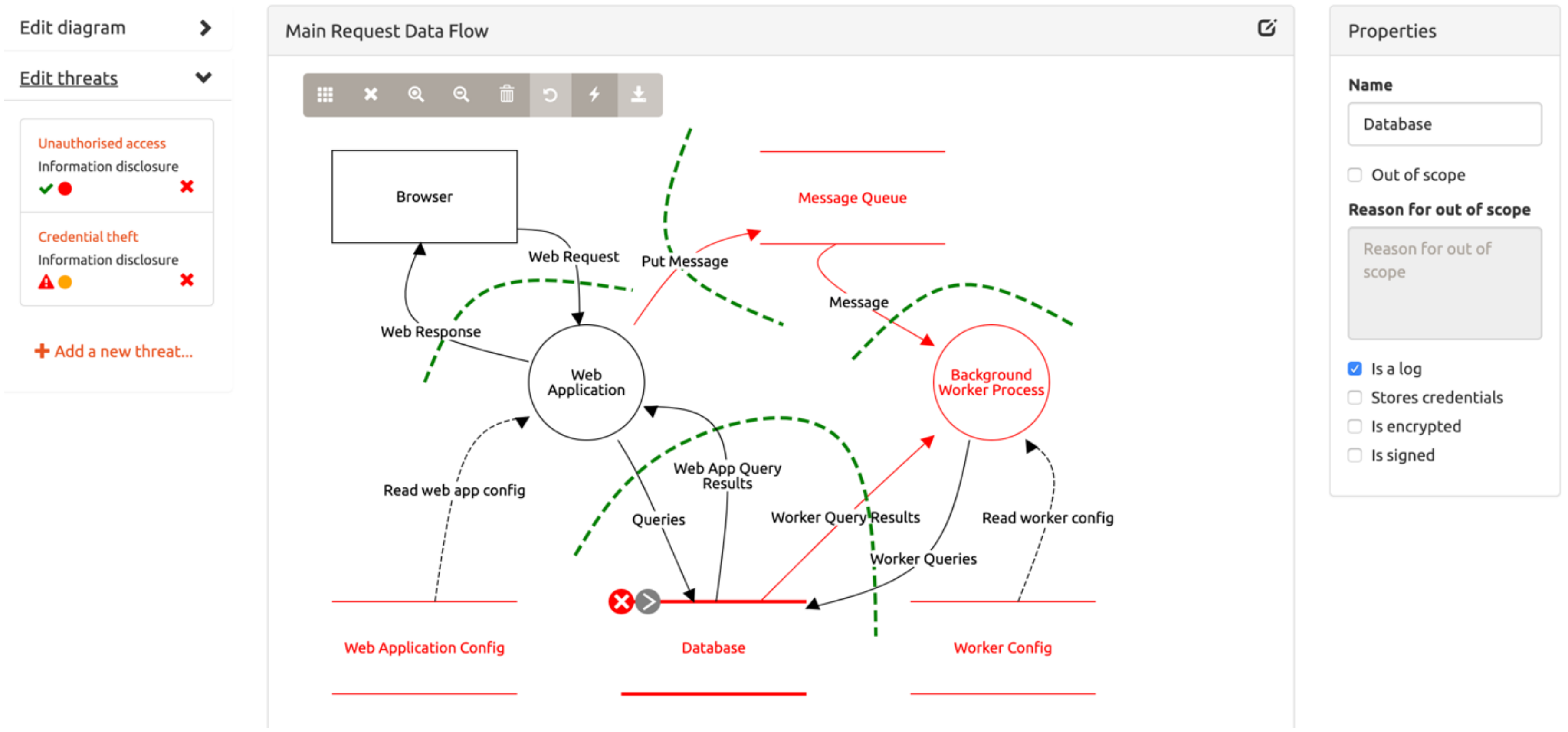
Project Risk
2



https://owasp.org/www-project-zap/



Project Risk
7



https://owasp.org/www-project-threat-dragon/

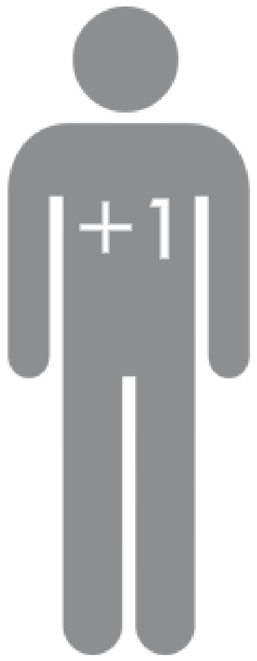
Missing pieces in tools

No options for SAST or IAST

A dashboard to track everything (requirements management, activities, releases, metrics)

Security Journey®

Tools: impact and headcount

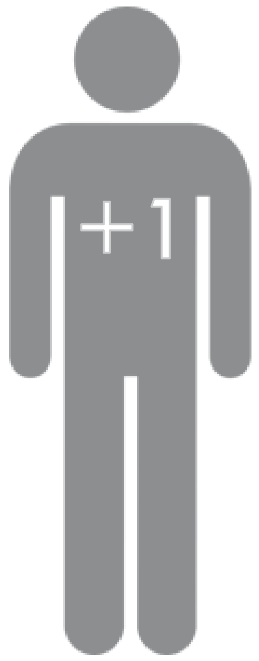


Infrastructure

CRS provides a true WAF solution

Dependency check identifies vulnerable 3rd party software

ZAP provides DAST, and plugs in to any dev methodology



Tools: getting started

Infrastructure

Add Dependency Check to your build pipeline tomorrow

Teach ZAP to Security Champions and interested testers

Work with your infra owner to deploy a test of ModSecurity + CRS

ThreatDragon POC

Headcount summary

Awareness, knowledge, and educ

+1

Process and measurement

+1.5

Tools

+2

The 13 OWASP projects as an AppSec program

Tools

- Design
- Infrastructure

DEPENDENCY-CHECK | OWASP ModSecurity Core Rule Set

Process and measurement

- Process
- Measurement

Application Security Verification Standard | OWASP Testing Guide | CODE REVIEW GUIDE | Application Threat Modeling | SAMM

Awareness and education

- Awareness
- Knowledge
- Hands-on training

OWASP Top 10 - 2017 The Ten Most Critical Web Application Security Risks | OWASP ProActive CONTROLS | OWASP CHEAT SHEET SERIES PROJECT

Security Community

Security Journey®

Apply What You Have Learned Today

- Next week you should:
  - Assess a high-level current state of your application security program and determine if you have visible gaps
- In the first three months following this presentation you should:
  - Perform a deeper assessment using OpenSAMM
  - Choose one of the dozen to implement
- Within six months you should:
  - Measure the impact of your first project implementation
  - Plan and execute on one or two additional pieces, resources permitting

Final thoughts for an AppSec program on the cheap

1. Use Open SAMM to assess current program and future goals.
2. There is no OWASP SDL; build/tailor required.
3. Start small; choose one item for awareness and education to launch your program.
4. Build security community early; it is the support structure.
5. Evaluate available projects in each category and build a 1-2-year plan to roll each effort out.
6. While OWASP is free, head count is not; plan for head count to support your “free” program.

How to engage with Security Journey

LEARN	Free trial of the Security Belt Program https://app.securityjourney.com
LISTEN	The Application Security Podcast
READ	<hi/5> five security articles that are worth your time powered by Security Journey https://www.securityjourney.com/hi5
EMAIL	chris_romeo@securityjourney.com
SOCIALS	@edgeroute @AppSecPodcast

Security Journey®