Building an AppSec Program with a Budget of $0: Beyond the OWASP Top 10

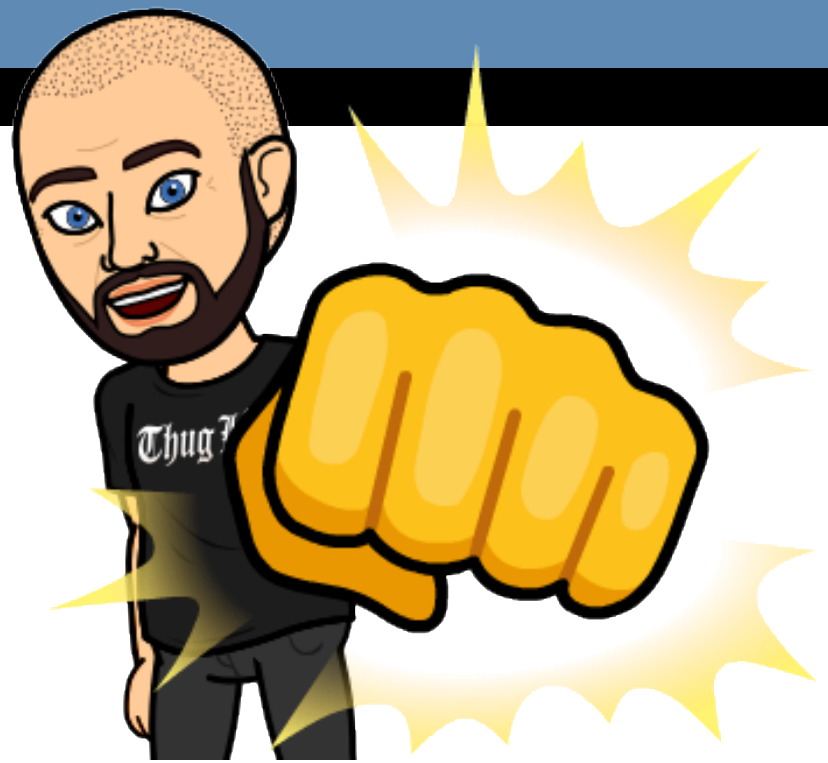
Chris Romeo, CEO, Security Journey

About Chris Romeo

- **CEO / Co-Founder @ Security Journey**
- **22 years in the security world, CISSP, CSSLP**
- **Co-host of the**

  APPLICATION SECURITY PODCAST

- **Co-Lead of the OWASP Triangle Chapter**



@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

Security Journey

Traditional AppSec programs

People

Process

Tools

Security Journey

Goals of an AppSec Program

1. **Limit vulnerabilities in deployed code**
2. **Build secure software and teach developers to build secure software**
3. **Provide processes and tools for AppSec standardization**
4. **Demonstrate software security maturity through metrics and assessment**

Security Journey


NOTE
NO
BUDGET

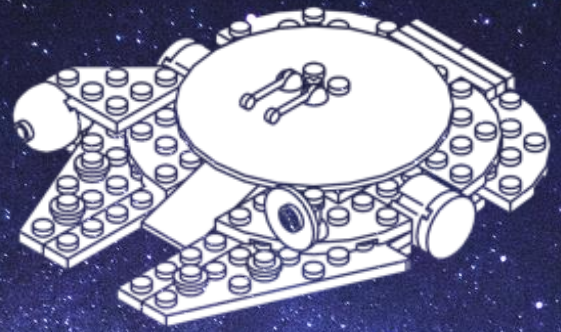
Large budget

Small budget

Security Champions

Goal: Educate about product security and embed expertise within every product team.

Security Journey



Flagship
Projects: 13

Lab
Projects: 35

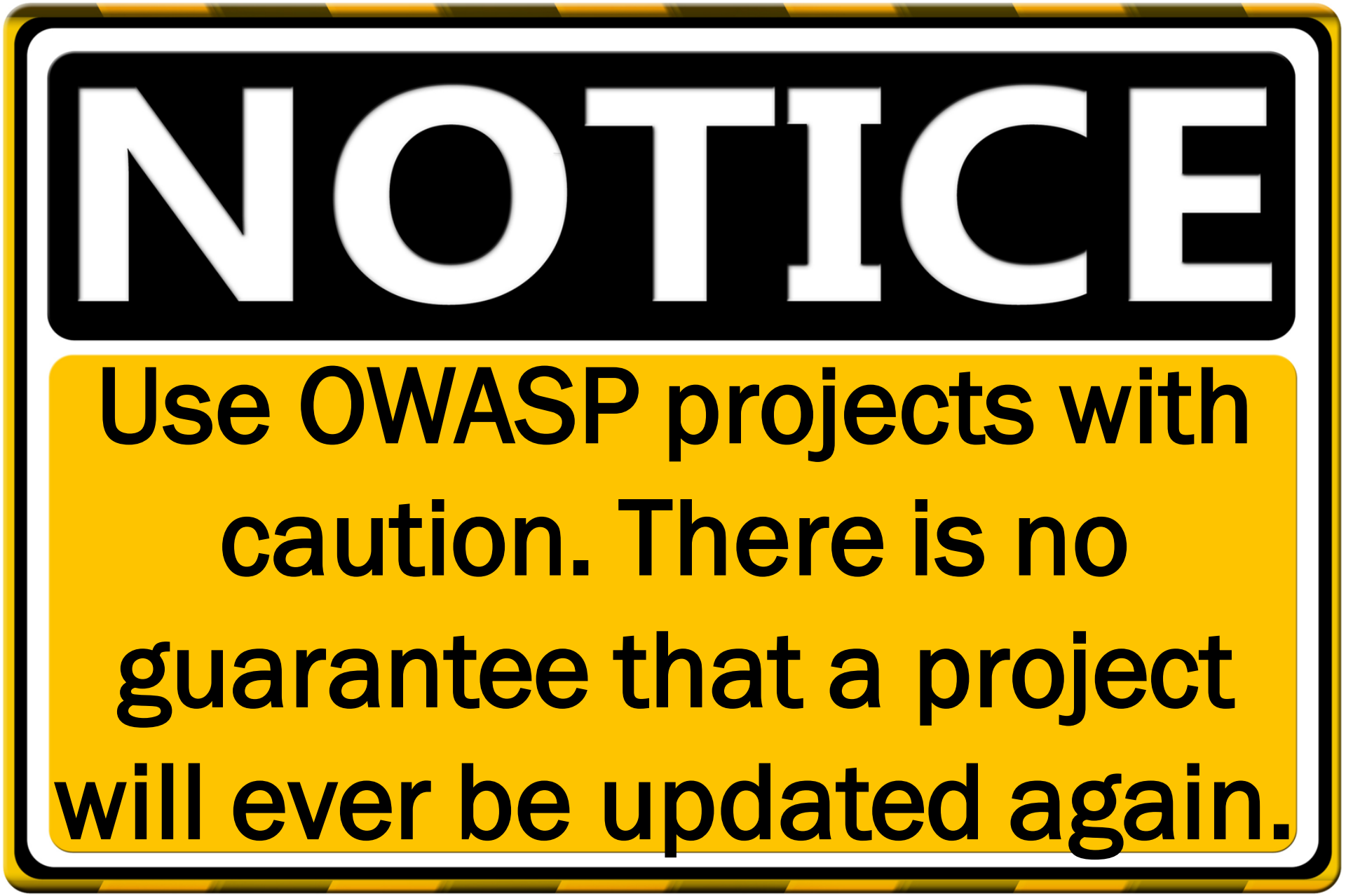
OWASP

Incubator
Projects: 49

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

Security Journey



Security Journey

The categories

Awareness, knowledge, education

Process and measurement

Tools

Security Journey

Awareness, knowledge and education

Security Journey

OWASP Top 10 - 2017

The Ten Most Critical Web Application Security Risks

Awareness

Risk of using project 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

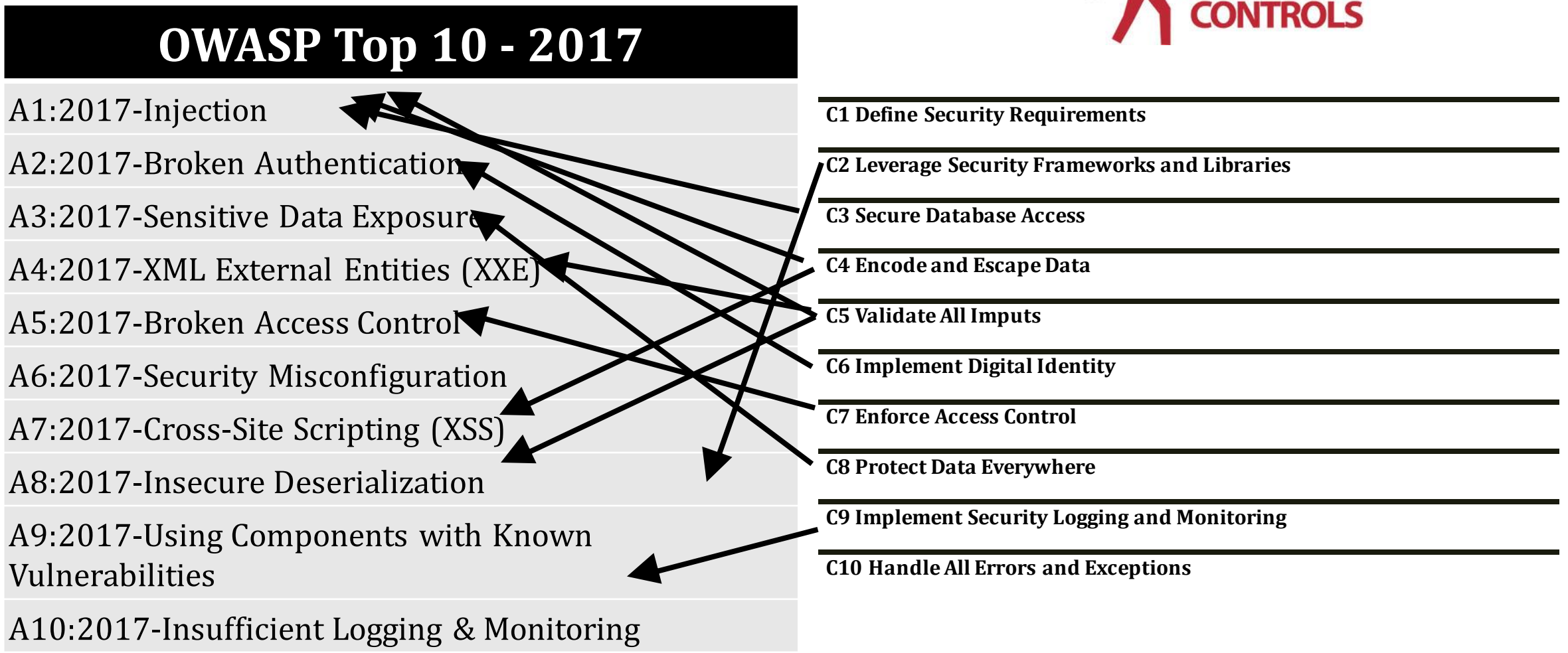
https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Security Journey

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://www.owasp.org/index.php/OWASP_Proactive_Controls

The intermingling



Security Journey

Account Aggregation	Account Creation	Ad Fraud	CAPTCHA defeat	Carding	Card Cracking	Cashing Out
Credential Cracking	Credential Stuffing	Denial of Inventory	Denial of Service	Expediting	Fingerprinting	Footprinting
Scalping	Scraping	Skewing	Sniping	Spamming	Token Cracking	Vulnerability Scanning

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

Knowledge

Risk of using project 2

V · T · E	**Cheat Sheets** [Collapse]
Developer / Builder	3rd Party Javascript Management · Access Control · AJAX Security Cheat Sheet · Authentication (ES) · Bean Validation Cheat Sheet · Choosing and Using Security Questions · Clickjacking Defense · Credential Stuffing Prevention Cheat Sheet · Cross-Site Request Forgery (CSRF) Prevention · Cryptographic Storage · C-Based Toolchain Hardening · Deserialization · DOM based XSS Prevention · Forgot Password · HTML5 Security · HTTP Strict Transport Security · Injection Prevention Cheat Sheet · Injection Prevention Cheat Sheet in Java · JSON Web Token (JWT) Cheat Sheet for Java · Input Validation · Insecure Direct Object Reference Prevention · JAAS · Key Management · LDAP Injection Prevention · Logging · Mass Assignment Cheat Sheet · .NET Security · OS Command Injection Defense Cheat Sheet · OWASP Top Ten · Password Storage · Pinning · Query Parameterization · REST Security · Ruby on Rails · 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
Assessment / Breaker	Attack Surface Analysis · REST Assessment · Web Application Security Testing · XML Security Cheat Sheet · XSS Filter Evasion
Mobile	Android Testing · IOS Developer · Mobile Jailbreaking
OpSec / Defender	Virtual Patching · Vulnerability Disclosure
Draft and Beta	Application Security Architecture · Business Logic Security · Content Security Policy · Denial of Service Cheat Sheet · Grails Secure Code Review · IOS Application Security Testing · PHP Security · Regular Expression Security Cheatsheet · Secure Coding · Secure SDLC · Threat Modeling

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

Security Journey

- Security knowledge reference
  - *Code example*
  - *Knowledge Base*

</> Code Language

PHP

C#/.net

JAVA

Py-Flask

Py-Django

Py-Django

Ruby on Rails

Go

```
package com.edw;

import org.owasp.esapi.ESAPI;
import org.jsoup.Jsoup;
import org.jsoup.safety.Whitelist;

public final class XssFilter {

        /**
        * Strips any potential XSS threats out of the value
        * @param value
        * @return
        */
        public String filter( String value ) {
            if( value == null )
                                        return null;

                // Use the ESAPI library to avoid encoded attacks.
                value = ESAPI.encoder().canonicalize( value );

                // Avoid null characters
                value = value.replaceAll("\0", "");

                // Clean out HTML
                value = Jsoup.clean( value, Whitelist.none() );

                return value;
        }
}
```

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

WEBGOAT

- Java based
- Version 8.0, long lasting
- Includes lessons and hacks

DevSlop

- Collection of DevOps-driven applications, specifically designed to showcase security catastrophes
- Micro services and containerization

- JavaScript based
- Intentionally insecure web app
- Encompasses the entire OWASP Top Ten and other severe security flaws

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

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

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

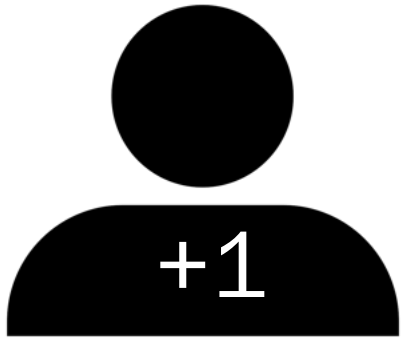
Delivery of awareness and education

Administration of the training platforms

Security Journey

Awareness and education: impact and headcount

Awareness	Knowledge	Hands-on training
• Foundational understanding of the most important concepts in AppSec	• A concise reference for solving the most difficult AppSec problems • Secure coding examples in multiple languages	• Assimilation of key concepts through activities that lock in knowledge and make it practical



Security Journey

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 the different training apps
- Schedule a hack-a-thon where teams gather together and work on the vulnerable apps in teams and learn from each other

Security Journey

Process and Measurement

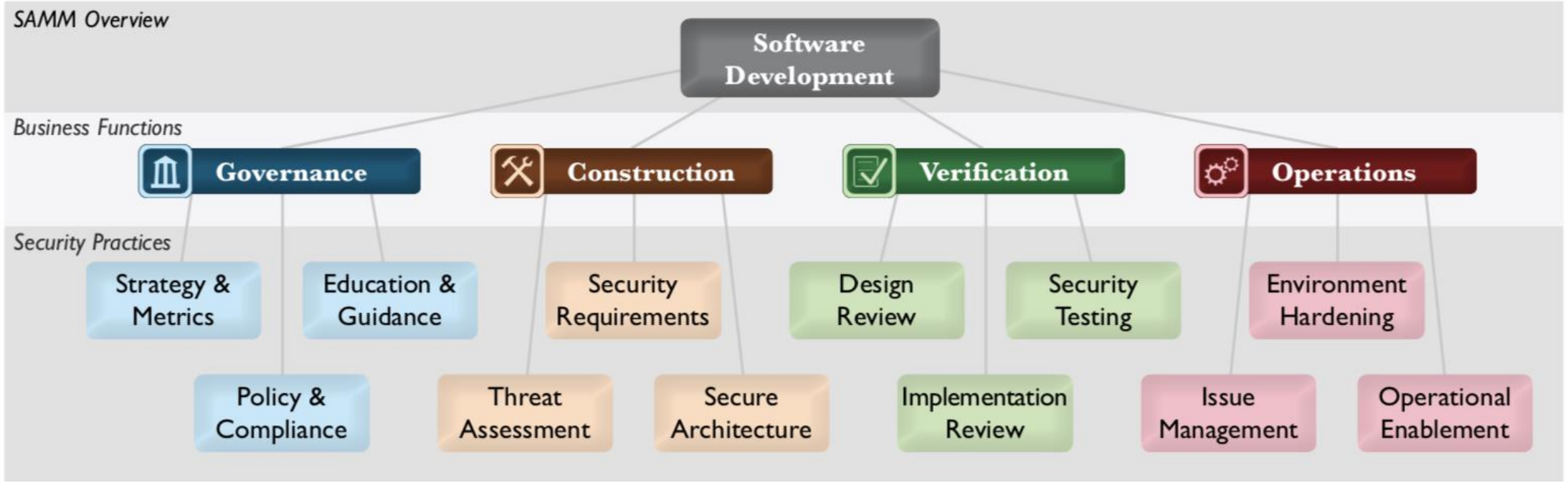
Application Security Verification Standard

DEFECTDOJO

CODE REVIEW GUIDE

Application Threat Modeling

Security Journey



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

0 Implicit starting point representing the activities in the practice being unfulfilled

1 Initial understanding and adhoc provision of security practice

2 Increase efficiency and/or effectiveness of the security practice

3 Comprehensive mastery of the security practice at scale

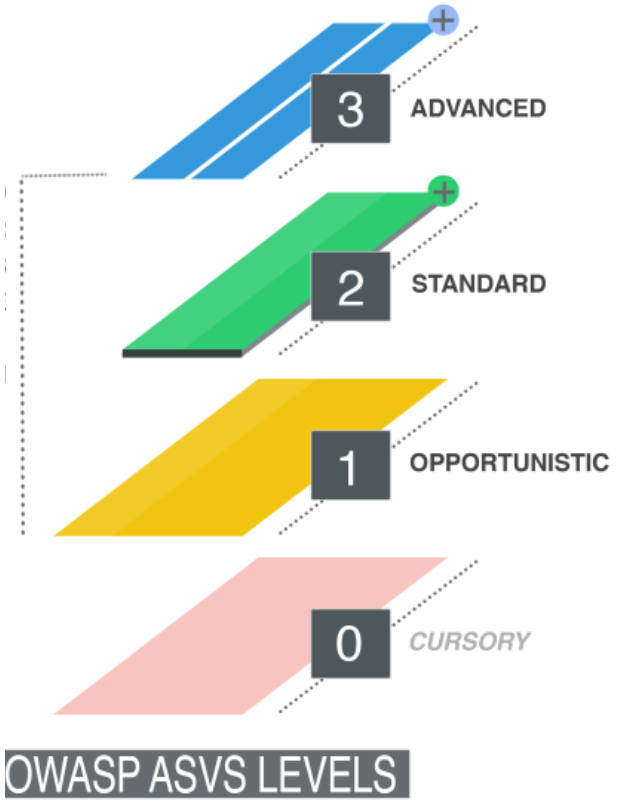
Application Security Verification Standard

Process

Knowledge

Risk of using project **0**

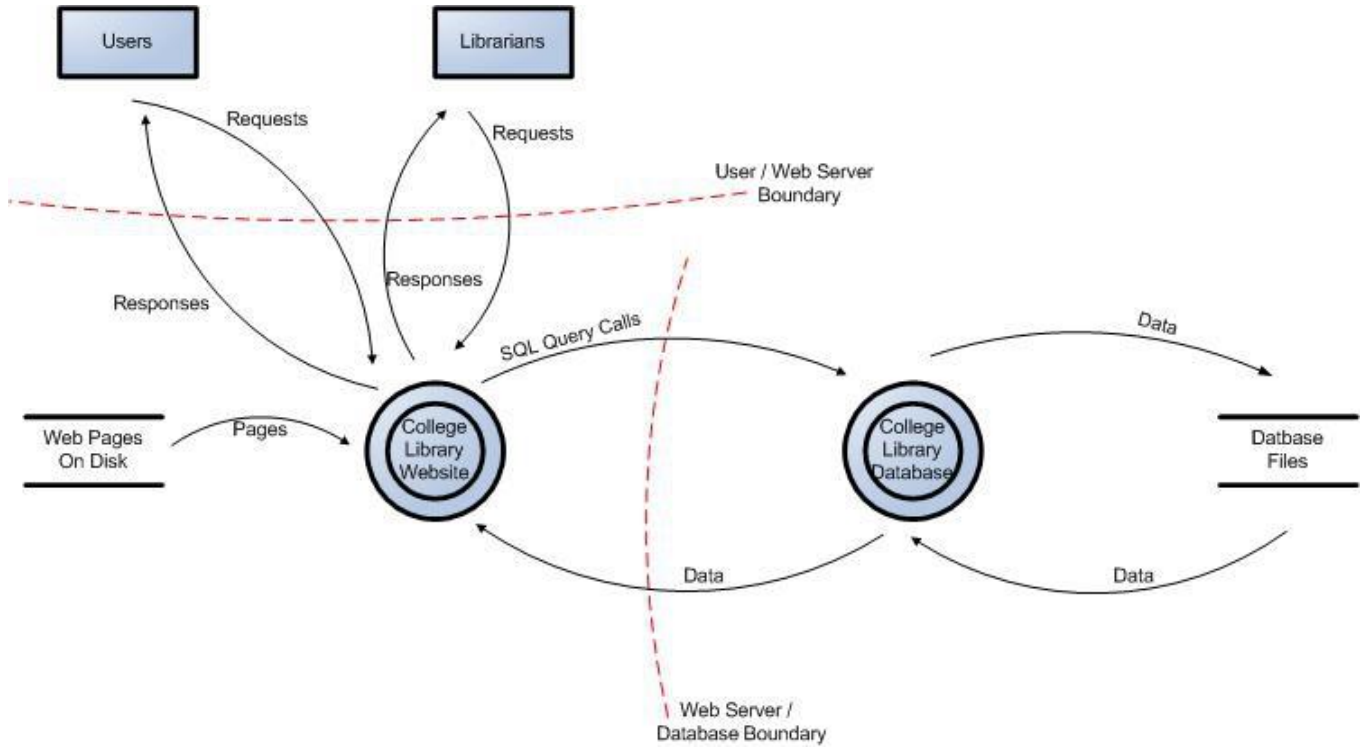
Requirement	
V1. Architecture, design and threat modelling	V11. HTTP security configuration
V2. Authentication	V13. Malicious controls
V3. Session management	V15. Business logic
V4. Access control	V16. File and resources
V5. Malicious input handling	V17. Mobile
V7. Cryptography at rest	V18. Web services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



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

Security Journey

Application Threat Modeling



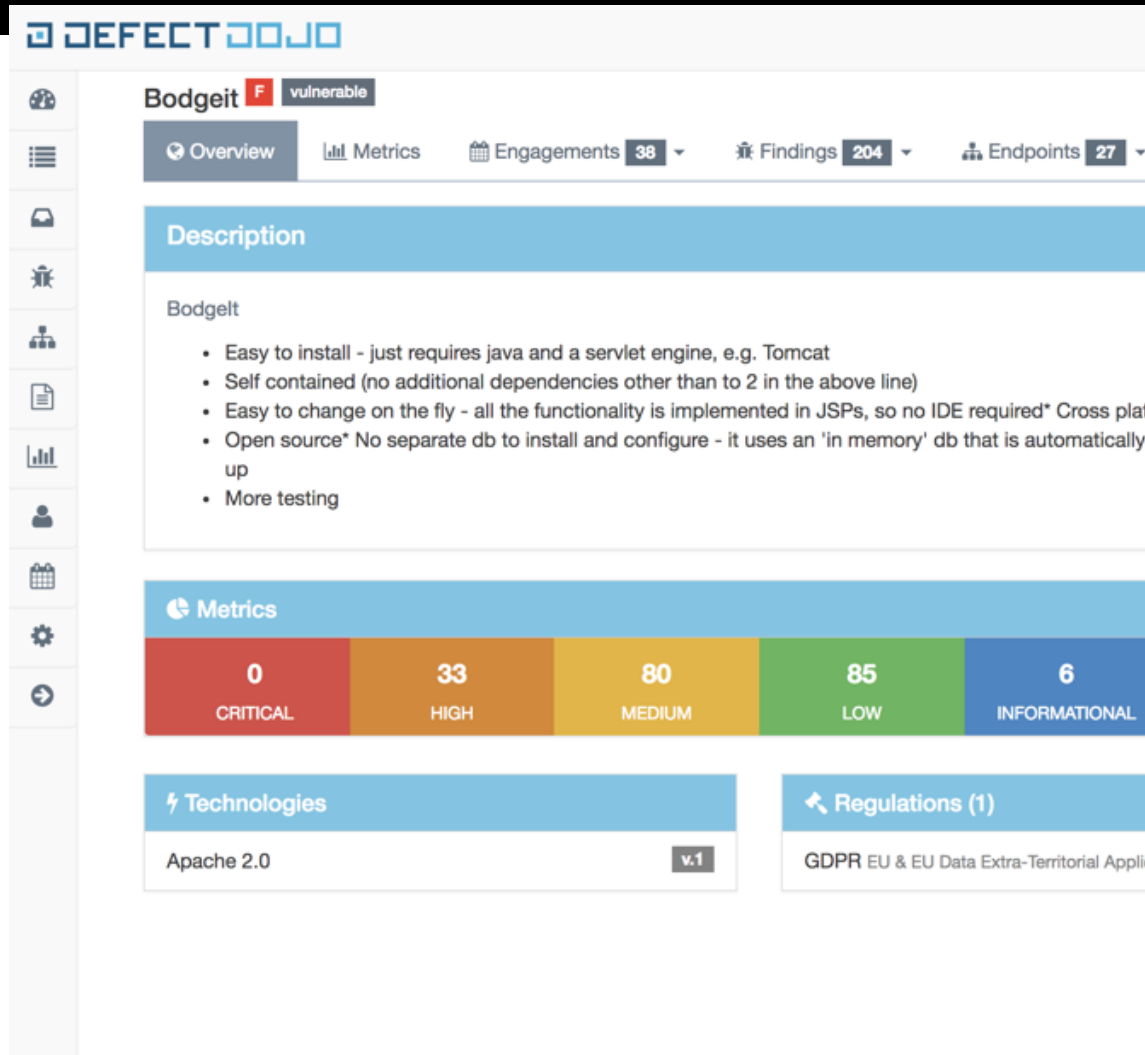
https://www.owasp.org/index.php/Application_Threat_Modeling

Security Journey

- 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

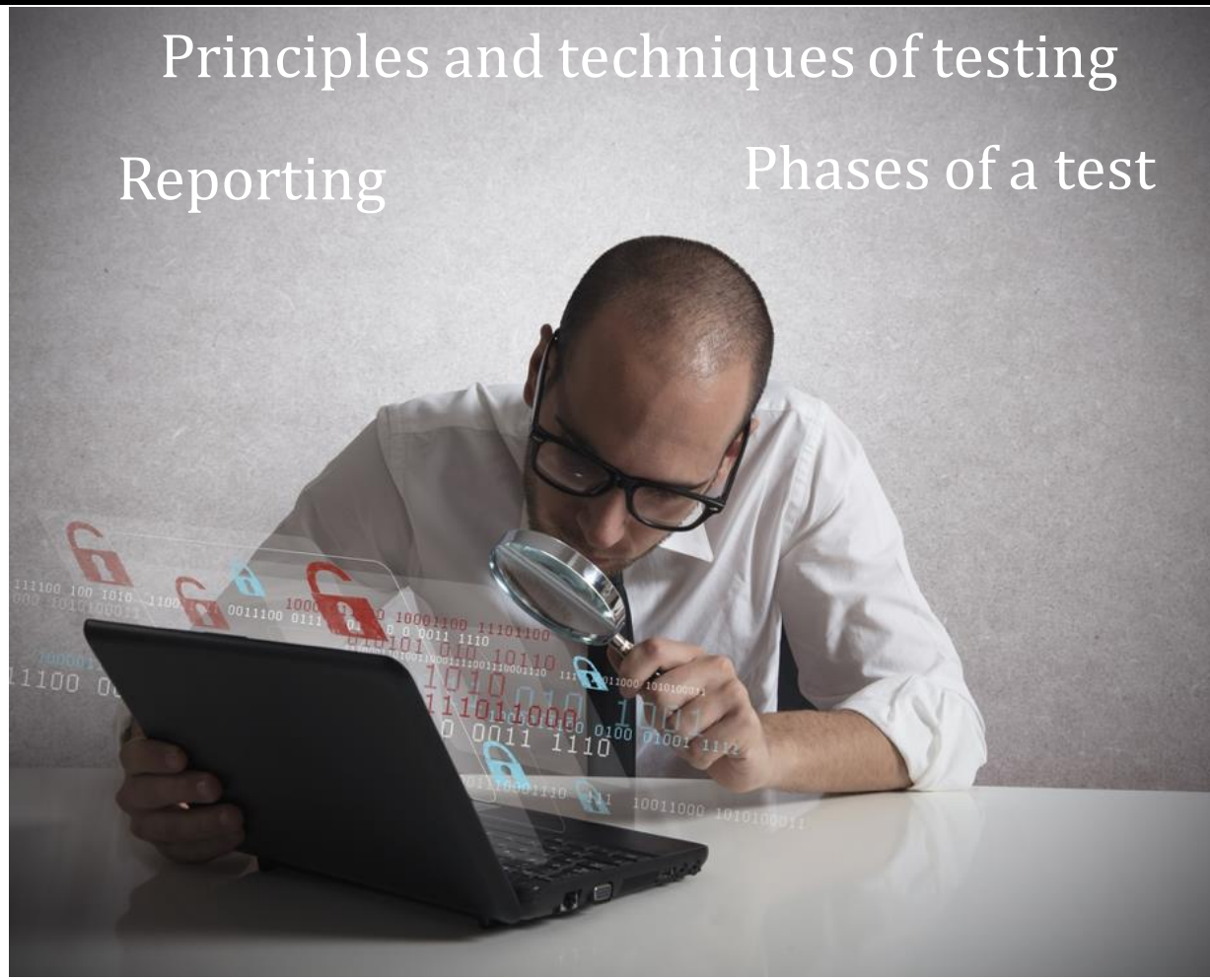
- One-stop source of truth for vuln findings.
  - Open source vulnerability management tool that streamlines the testing process.
- Goal: make optimizing vulnerability tracking less painful and reduce the amount of time security professionals spend logging vulnerabilities.
- AppSec Programs, QA, Pen Testers
  - Imports for common vuln scanners.
  - Custom report generation.
  - Metrics and dashboards.
  - App & infra findings supported.



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

Security Journey

- 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://www.owasp.org/index.php/OWASP_Testing_Project

OWASP Open Web Application Security Project | Testing Guide

Testing for Reflected Cross site scripting (OTG-INPVAL-001)

How to Test

Black Box testing

A black-box test will include at least three phases:

[1] Detect input vectors. For each web page, the tester must determine all the web application's user-defined variables and how to input them. This includes hidden or non-obvious inputs such as HTTP parameters, POST data, hidden form field values, and predefined radio or selection values. Typically in-browser HTML editors or web proxies are used to view these hidden variables. See the example below.

[2] Analyze each input vector to detect potential vulnerabilities. To detect an XSS vulnerability, the tester will typically use specially crafted input data with each input vector. Such input data is typically harmless, but trigger responses from the web browser that manifests the vulnerability. Testing data can be generated by using a web application fuzzer, an automated predefined list of known attack strings, or manually.

Some example of such input data are the following:

```
<script>alert(123)</script>
```

```
"><script>alert(document.cookie)</script>
```

For a comprehensive list of potential test strings, see the XSS Filter Evasion Cheat Sheet.

[3] For each test input attempted in the previous phase, the tester will analyze the result and determine if it represents a vulnerability that has a realistic impact on the web application's security. This requires examining the resulting web page HTML and searching for the test input. Once found, the tester identifies any special characters that were not properly encoded, replaced, or filtered out. The set of vulnerable unfiltered special characters will depend on the context of that section of HTML.

Ideally all HTML special characters will be replaced with HTML entities. The key HTML entities to identify are:

```
> (greater than)
< (less than)
& (ampersand)
' (apostrophe or single quote)
" (double quote)
```

However, a full list of entities is defined by the HTML and XML specifications. Wikipedia has a complete reference [1].

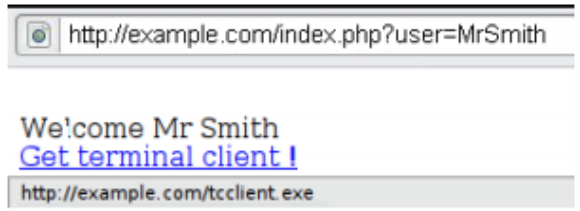
Within the context of an HTML action or JavaScript code, a different set of special characters will need to be escaped, encoded, replaced, or filtered out. These characters include:

```
\n (new line)
\r (carriage return)
\' (apostrophe or single quote)
\" (double quote)
\\ (backslash)
\uXXXX (unicode values)
```

For a more complete reference, see the Mozilla JavaScript guide. [2]

Example 1

For example, consider a site that has a welcome notice " Welcome %username% " and a download link.

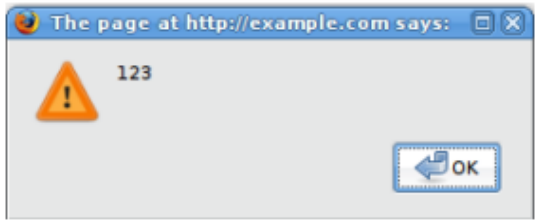


The tester must suspect that every data entry point can result in an XSS attack. To analyze it, the tester will play with the user variable and try to trigger the vulnerability.

Let's try to click on the following link and see what happens:

```
http://example.com/index.php?user=<script>alert(123)</
script>
```

If no sanitization is applied this will result in the following popup:



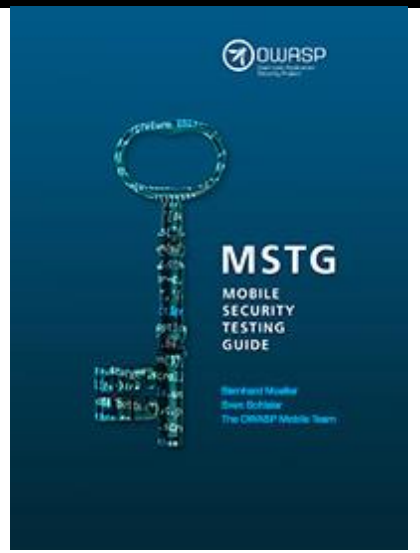
Security Journey

OWASP Mobile Security Requirements and Testing Guide

Process

Risk of using project 3

- Why mobile application security?
  - Different attack surface
    - Local storage
    - Local authentication
    - OS interaction
  - Different vulnerabilities
    - Reverse engineering
    - Secret storage
    - Fewer to NO XSS or CSRF



- Mobile security testing guide
- Maps directly to MASVS
- Native Android and iOS applications
- Uses OWASP testing guide for server side

- Mobile Application Security Verification
- 3 levels of requirements
  - Baseline
  - Defense-in-depth
  - Advanced
- Fork of ASVS dedicated to mobile

Security Journey

Missing pieces in process and measurement

End-to end SDL or Secure SDLC

Program metrics

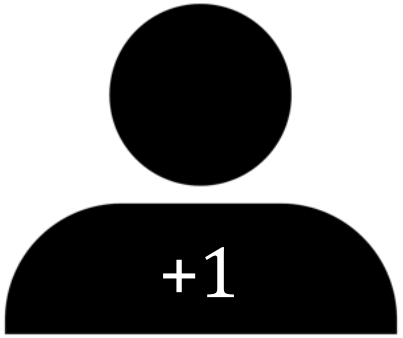
Deployment advice/experience on how to be successful

Security Journey

Process and measurement: impact and headcount

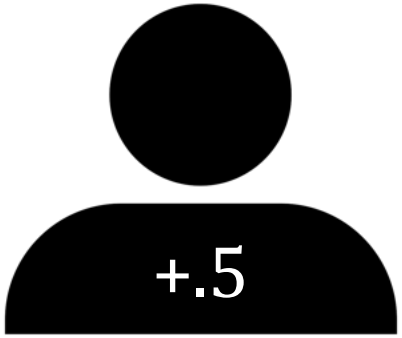
Process

- 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



Security Journey

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 a future plan for where you want to get to
- 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

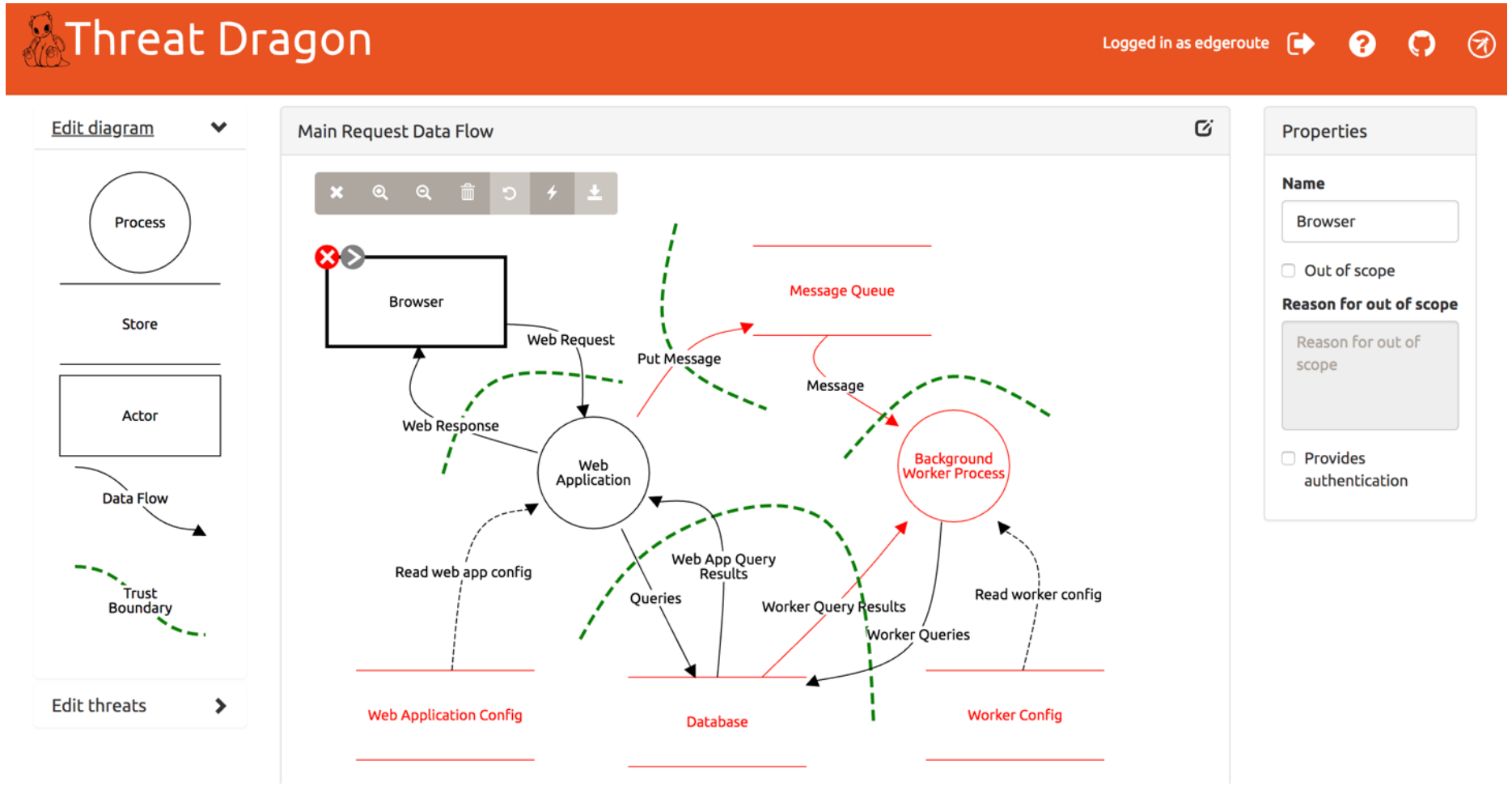
Tools

OWASP
ModSecurity
Core Rule Set
THE 1ST LINE OF DEFENSE

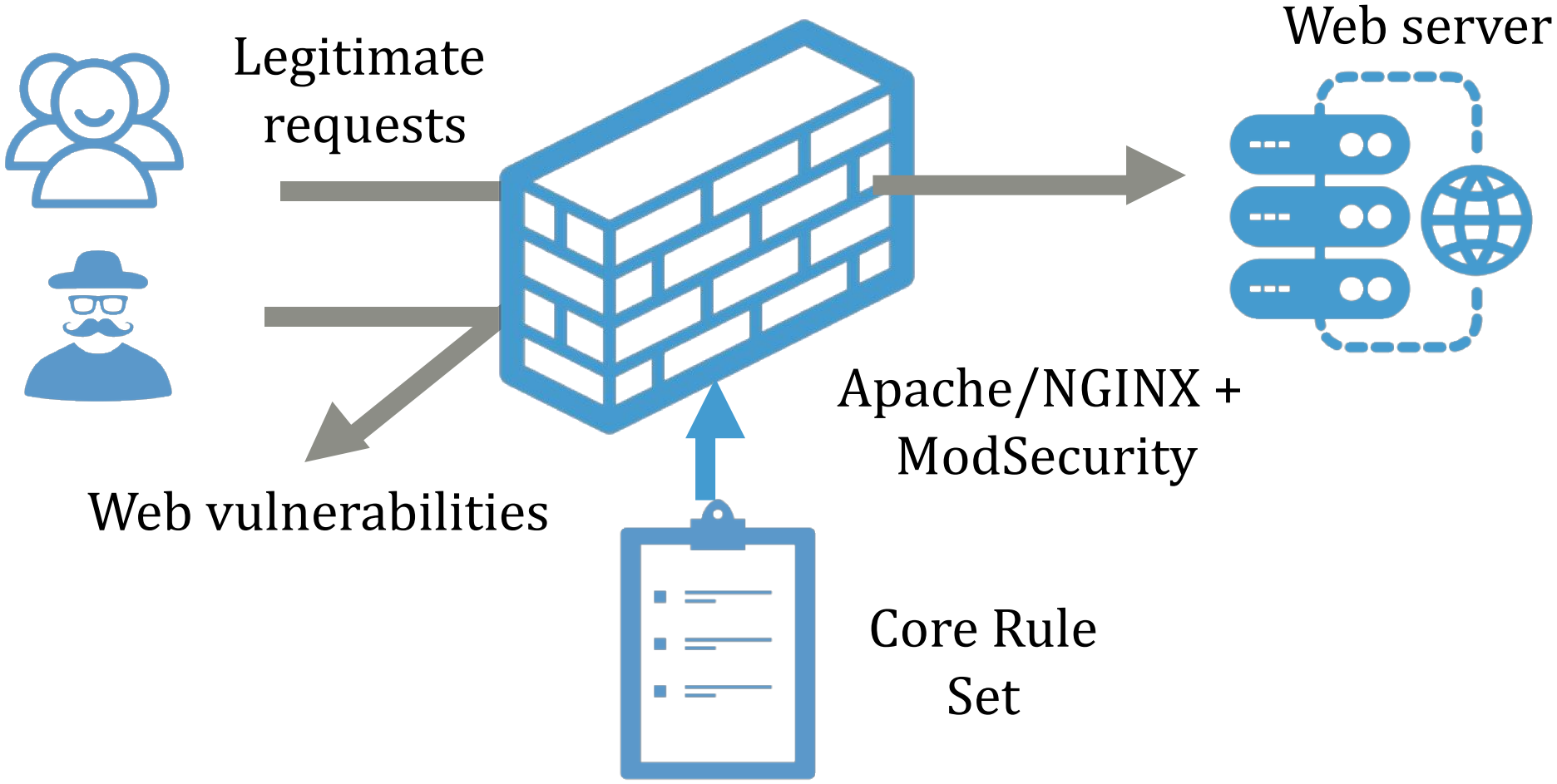
DEPENDENCY-CHECK

DEPENDENCY-TRACK

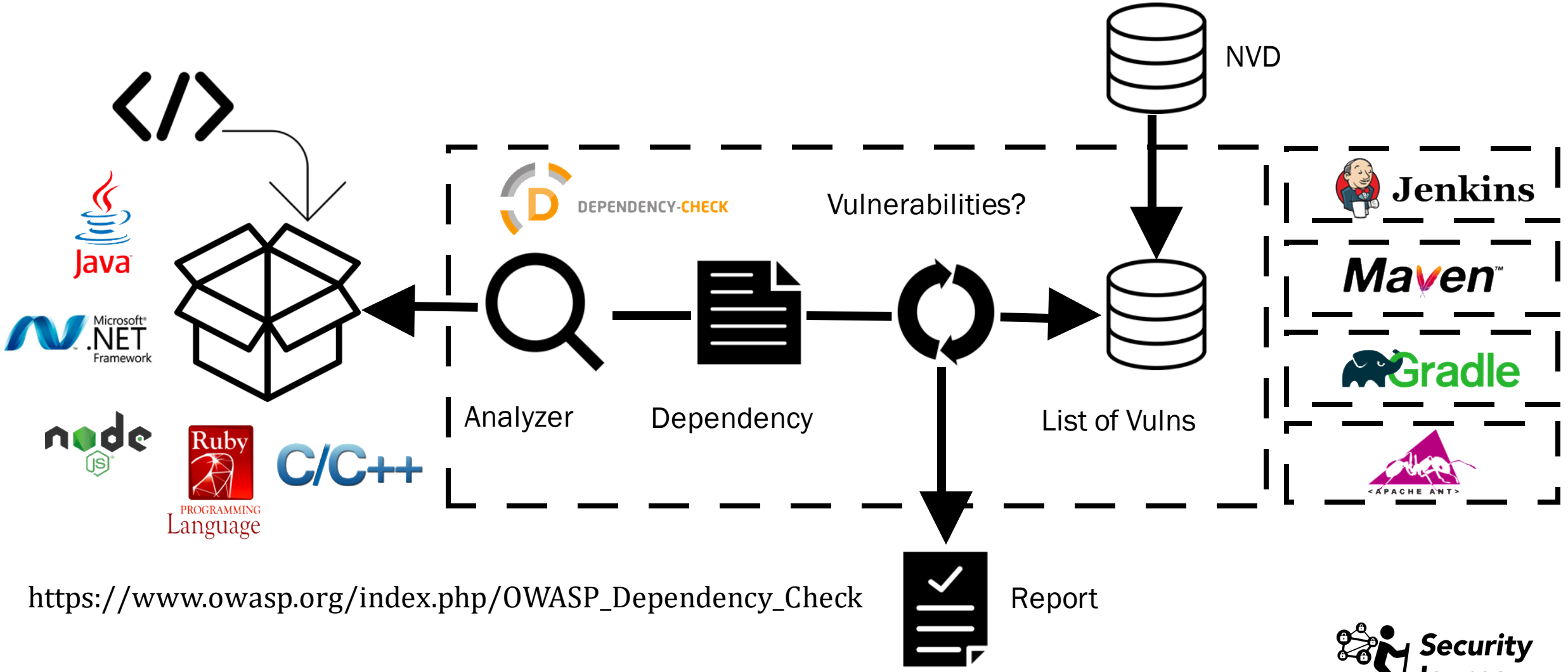
Security Journey



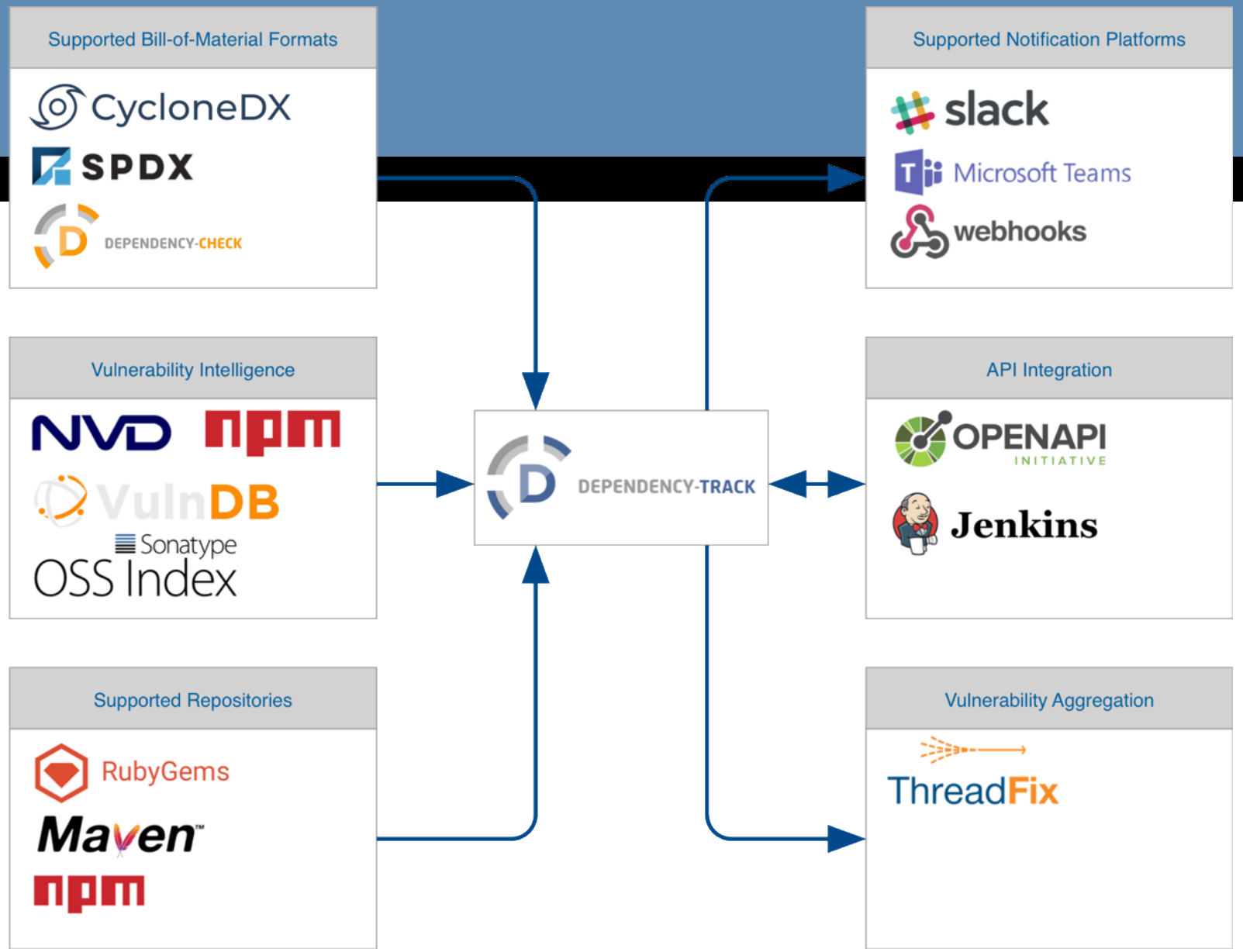
https://www.owasp.org/index.php/OWASP_Threat_Dragon



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



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



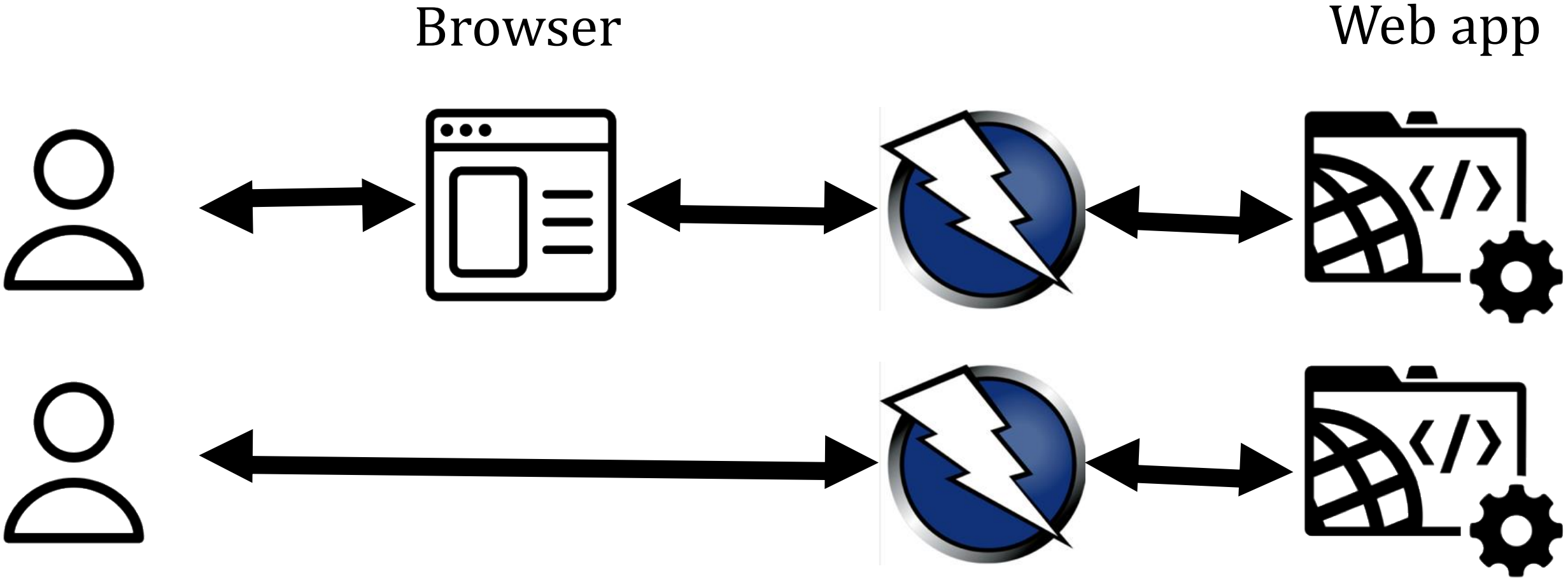
DEPENDENCY-TRACK

Infrastructure

Risk of using project 4

Source: https://docs.dependencytrack.org/integrations/ecosystem/

Security Journey



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

Missing pieces in tools

No options for SAST or IAST

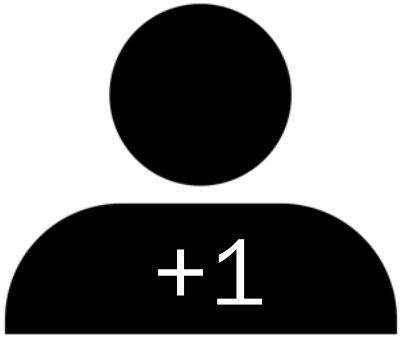
A dashboard to track everything (requirements management, activities, releases, metrics)

Security Journey

Tools: impact and headcount

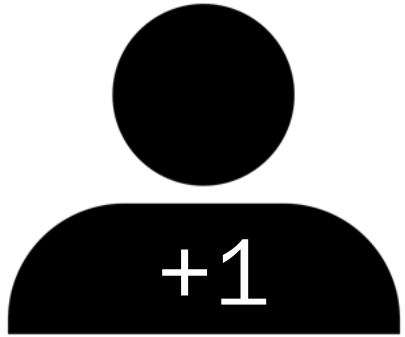
Design

- Threat dragon provides a new, web based approach to capturing threats that will reach Enterprise status if it delivers on the roadmap



Infrastructure

- CRS provides a true WAF solution
- Dependency check identifies vulnerable 3rd party software
- Dependency track provides Enterprise 3rd party software tracking
- ZAP provides DAST, and plugs in to any dev methodology



Security Journey

Tools: getting started

Design

- Use threat dragon as the tool to teach threat modeling and scale it across your development teams
  - Partner with application threat modeling knowledge

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

Security Journey

Headcount summary

Awareness and education

- Awareness
- Knowledge
- Hands-on training

+1

Process and measurement

- Knowledge
- Hands-on training

+1.5

Tools

- Design
- Infrastructure

+2

Security Journey

The OWASP stack as an AppSec program

Tools

Process and measurement

Awareness and education

Design

Infrastructure

Process

Measurement

Awareness

Knowledge

Hands-on training

DEPENDENCY-CHECK

DEPENDENCY-TRACK

OWASP ModSecurity Core Rule Set

Application Security Verification Standard

Application Threat Modeling

OWASP Testing Guide

CODE REVIEW GUIDE

OWASP Mobile Security Requirements and Testing Guide

OWASP Top 10 - 2017

The Ten Most Critical Web Application Security Risks

DevSlop

Security Knowledge Framework

OWASP Proactive Controls

OWASP Cheat Sheets

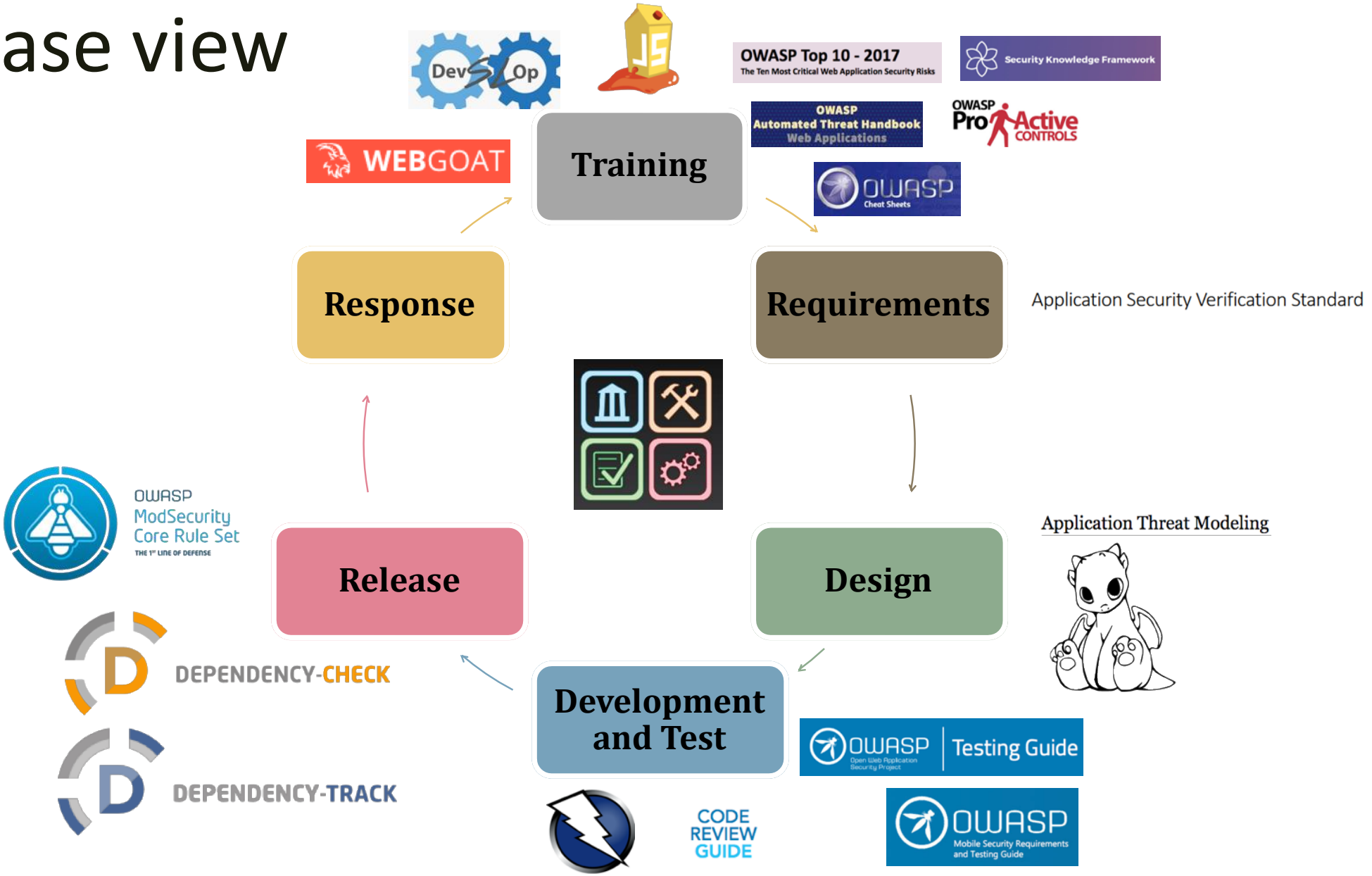
OWASP Automated Threat Handbook Web Applications

WEBGOAT

Security Community

Security Journey

SDL phase view

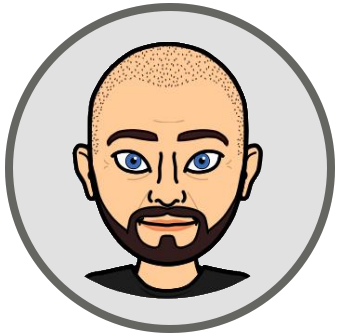


Final thoughts for an AppSec program on the cheap

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.

Security Journey

Q+A and Thank you!



Chris Romeo, CEO / Co-Founder

chris_romeo@securityjourney.com

www.securityjourney.com

@edgeroute, @SecurityJourney, @AppSecPodcast

Security Journey