

Journée sécurité des SI
dès la conception

ISACA-Québec
5 décembre 2018



The OWASP Foundation
www.owasp.org

Fondements de la sécurité applicative et prise en charge avec les outils OWASP

**UPDATED
NOV 2018**

Patrick Leclerc
Président du chapitre OWASP Ville de Québec
patrick.leclerc@owasp.org



OWASP

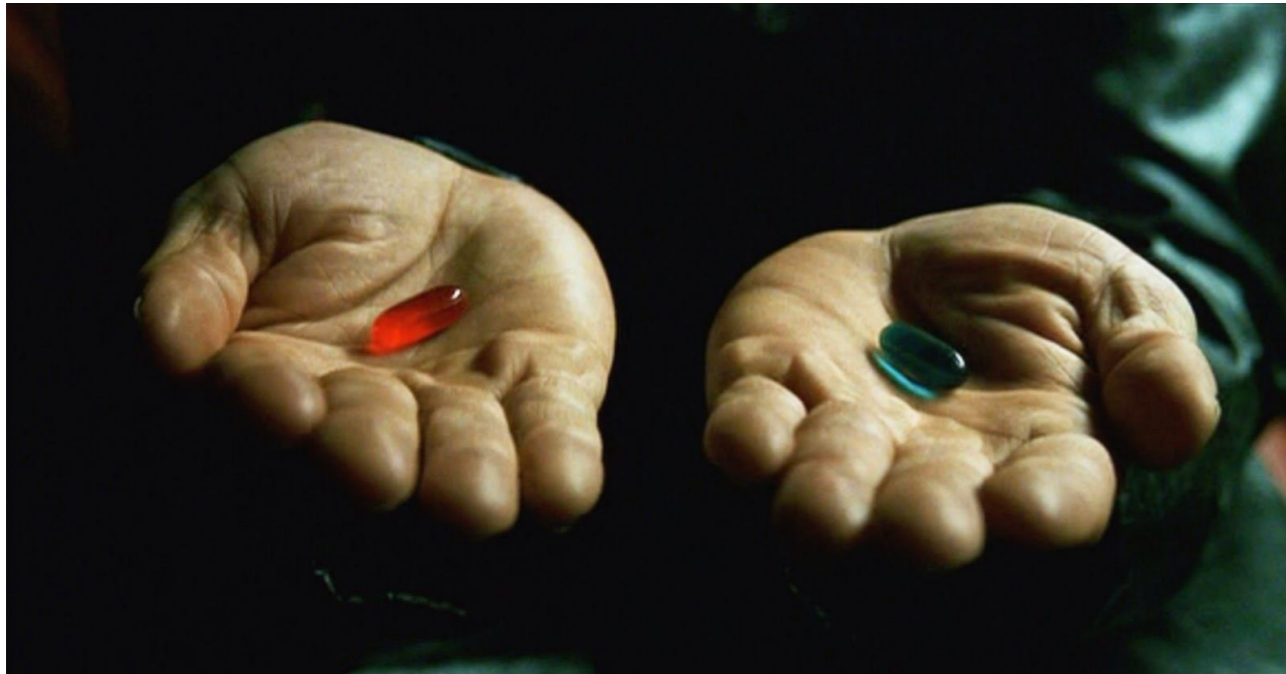
Open Web Application
Security Project

WWW.OWASP.ORG

Mondiale / Non-lucrative / Bénévole / « Open source » / Neutre / Indépendante

Mission : rendre la **sécurité applicative visible** + vous permettre de **prendre des décisions informées sur les risques de sécurité** des applications

Patrick Leclerc



24 ans d'expérience TI

Architecture logicielle et de sécurité

- Conseiller en architecture logicielle désormais dédié à la sécurité des applications

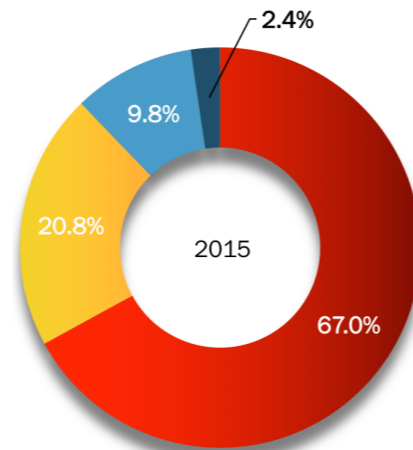
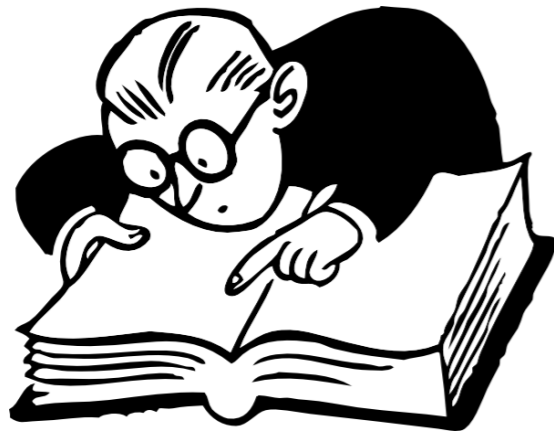


Leader du chapitre
OWASP Ville de Québec



Conseiller en sécurité des actifs
informationnels à La Capitale

Plan de la présentation



Qu'est ce que la sécurité applicative?



Qu'est ce que la sécurité applicative?



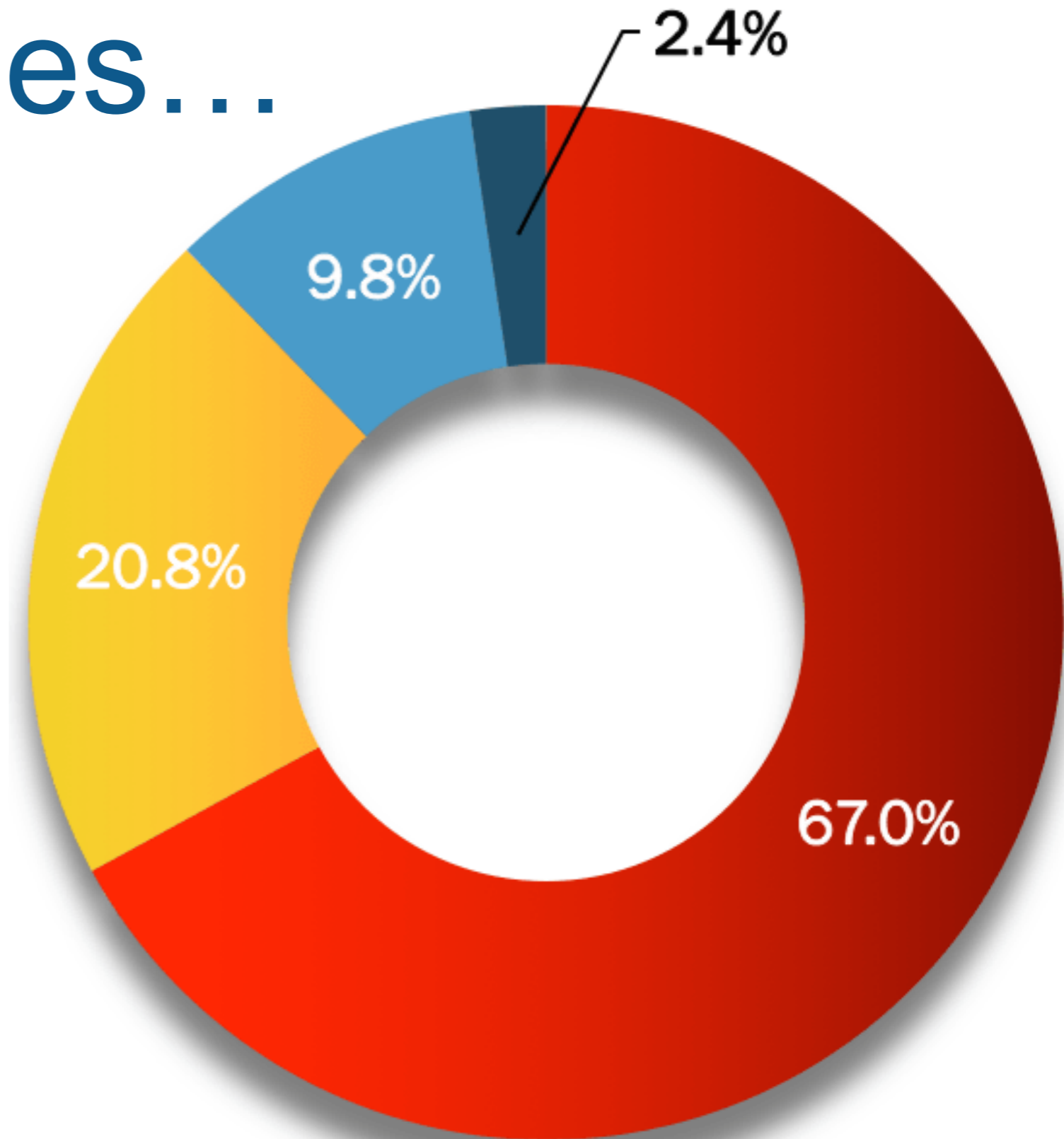
Définition ISO 27034 (Guide ISO sur la sécurité applicative):

« **La sécurité applicative est un processus effectué pour appliquer des contrôles et des mesures aux applications d'une organisation afin de gérer le risque** de leur utilisation ».

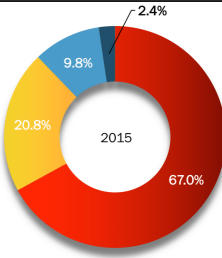
« **Les contrôles et les mesures peuvent être appliqués à l'application elle-même (ses processus, les composants, les logiciels et résultats), à ses données (données de configuration, les données de l'utilisateur, les données de l'organisation), et à toutes les technologies, les processus et acteurs impliqués dans le cycle de vie de l'application** ».

La sécurité applicative ne concerne pas seulement l'application, « l'gars d la sécurité pis les développeurs »!

Les statistiques...



Statistiques sur la sécurité applicative



77%
~~70%~~ des applications comportent au moins une vulnérabilité OWASP top 10
– Veracode 2018

20%
~~15%~~ des applications Web ont une vulnérabilité **critique** ou **élevée**
– Edgescan report 2018

96% des applications utilisent du code *open source*, et **90%** ~~2/3~~ comportent des vulnérabilités
– Black Duck OSSRA Report 2017

55% d'augmentation de brèches causées par du code *open source*
(Equifax avec Apache Struts, OpenSSL Heartbleed, etc)
– Sonatype 2018

Le nombre total d'attaques Web a bondi de **69%** ~~25%~~, année après année
– Akamai Q3 2017

Les applications Web sont responsables de plus de **21%** des brèches confirmées (au **1^{ier}** rang des patterns de brèches)
– Verizon DBIR 2018



Les impacts...





Impacts pour l'organisation

- Pertes de confidentialité
- Bris d'intégrité
- Fraudes
- Lourdes pertes financières
- Secrets d'entreprises dévoilés

Selon une étude de *Ponemon Institute (2018)*

Au **Canada**:

- Coût moyen d'une brèche : ~~5,78~~^{6,11} millions
- Coût moyen par identité volée: ~~255\$~~^{266\$}



Autres coûts...

« Le coût de la cybercriminalité comprend **beaucoup plus que la valeur de l'information volée**, il comprend aussi :

les coûts de l'interruption des activités,

les occasions perdues,

les frais juridiques,

les coûts des rapports,

les dommages à la réputation,

et les efforts de rétablissement. »





Que retenir de tout ça?

- Les **applications Web** sont **déjà au 1 rang** des causes des **brèches** de sécurité...
- Et la **tendance s'intensifie**:
 - + d'applications!
 - + de vulnérabilités dans les applications
 - + d'attaques sur les applications
- **Rappelez-vous**: Votre **budget** de sécurité informatique doit correspondre aux **priorités** et aux **risques** de votre entreprise...



Pourquoi les applications Web échappent-elles aux mesures de sécurité traditionnelles?



Le périmètre de sécurité du passé...



Autrefois:

- Sécurité **physique** et d'**infrastructure** *autour* des applications de missions (*internes*)
- **Utilisateurs internes**, **appareils internes** sous le **contrôle** de l'organisation

INTERNET + MONDIALISATION DES MARCHÉS

Hier:

- Applications **internes exposées** sur le Web avec **+/- les mêmes mesures**
 - Dorénavant accessibles à tous : usagers légitimes **et pirates!**
 - Souvent les applications ont été développées par des développeurs qui **en connaissaient peu sur la sécurité**
- **Perte d'étanchéité du périmètre de sécurité...** par l'exposition des applications **peu sécurisées** sur Internet

Le nouveau périmètre



{ Code / Composants / Librairies / Scripts } **EXTERNES**
Smartphones – IoT – BYOD – CLOUD

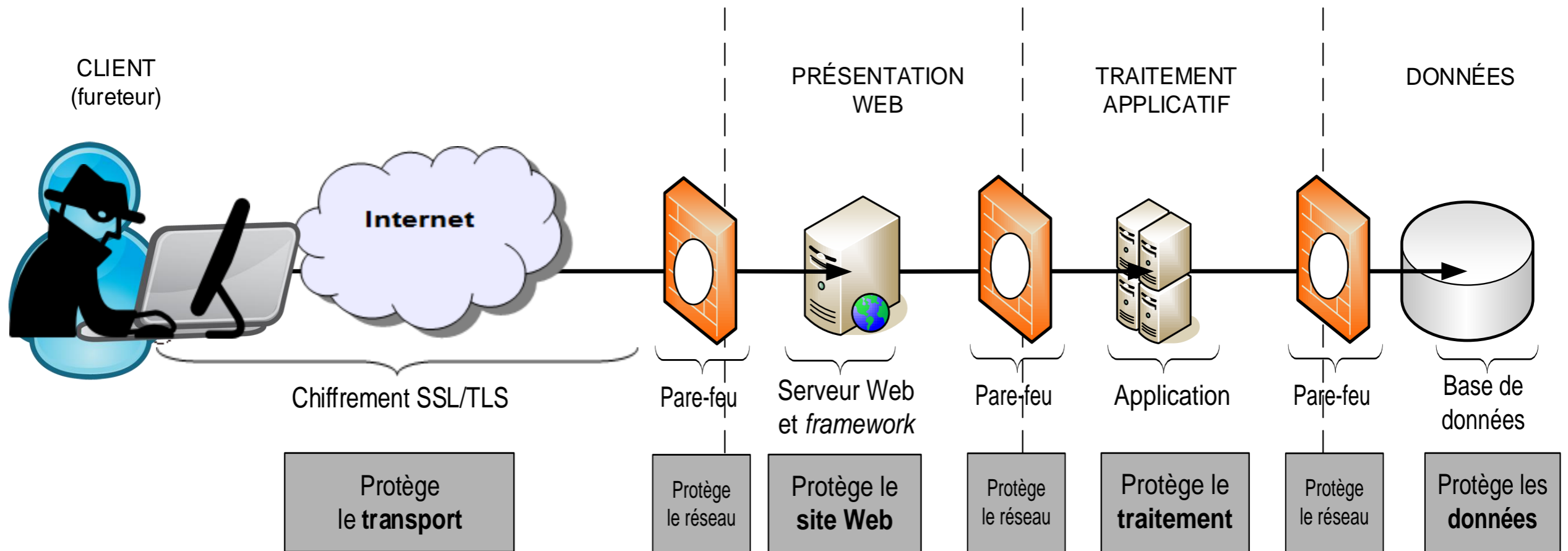
Aujourd'hui :

- Applications éparpillées sur le Cloud, chez plusieurs fournisseurs
- Les **applications connectées** sont partout: mobiles, voitures, « *wearable computers* », domotique, appareils électroniques...

Où est le périmètre?

La sécurité des applications est le
nouveau périmètre de défense

Architecture n-tiers commune



Autres défis de la sécurité applicative...



- **Single Page Application (SPA)**
 - Logique applicative déplacée vers le navigateur du client
 - Plusieurs portes d'accès... plusieurs services à sécuriser (API)
- **Développement Agile**
 - **Vue** souvent **partielle** du système final, difficulté d'avoir une vue d'ensemble des risques de sécurité du(des) système(s)
- **Cycles de livraisons accélérés (DevOps, CI/CD)**
 - Les tests de sécurité doivent être **automatisés** pour **suivre la cadence**
- **Microservices (containers, serverless functions, etc.)**
 - Architectures plus complexes à gérer d'un point de vue sécurité
- **Le Cloud**
 - Protection des données, gestion des identités, etc.



<http://>

Les 10 risques les plus critiques des applications Web



OWASP Top 10 - 2017

The Ten Most Critical Web Application Security Risks



OWASP Top 10 2017

Fiches explicatives



Vulnérabilités

- Menaces et vecteurs d'attaque
- Impacts

Niveaux de risques:

Threat Agents	Exploitability	Weakness Prevalence	Weakness Detectability	Technical Impacts	Business Impacts
Application Specific	Easy: 3	Widespread: 3	Easy: 3	Severe: 3	Business Specific
	Average: 2	Common: 2	Average: 2	Moderate: 2	
	Difficult: 1	Uncommon: 1	Difficult: 1	Minor: 1	

Explications:

- Application vulnérable?
- Comment prévenir
- Exemples d'attaque
- Références et compléments d'information

A1
:2017

Injection

7

App. Specific	Exploitability: 3	Prevalence: 2	Detectability: 3	Technical: 3	Business ?
Almost any source of data can be an injection vector, environment variables, parameters, external and internal web services, and all types of users. Injection flaws occur when an attacker can send hostile data to an interpreter.		Injection flaws are very prevalent, particularly in legacy code. Injection vulnerabilities are often found in SQL, LDAP, XPath, or NoSQL queries, OS commands, XML parsers, SMTP headers, expression languages, and ORM queries. Injection flaws are easy to discover when examining code. Scanners and fuzzers can help attackers find injection flaws.			Injection can result in data loss, corruption, or disclosure to unauthorized parties, loss of accountability, or denial of access. Injection can sometimes lead to complete host takeover. The business impact depends on the needs of the application and data.

Is the Application Vulnerable?

An application is vulnerable to attack when:

- User-supplied data is not validated, filtered, or sanitized by the application.
- Dynamic queries or non-parameterized calls without context-aware escaping are used directly in the interpreter.
- Hostile data is used within object-relational mapping (ORM) search parameters to extract additional, sensitive records.
- Hostile data is directly used or concatenated, such that the SQL or command contains both structure and hostile data in dynamic queries, commands, or stored procedures.

Some of the more common injections are SQL, NoSQL, OS command, Object Relational Mapping (ORM), LDAP, and Expression Language (EL) or Object Graph Navigation Library (OGNL) injection. The concept is identical among all interpreters. Source code review is the best method of detecting if applications are vulnerable to injections, closely followed by thorough automated testing of all parameters, headers, URL, cookies, JSON, SOAP, and XML data inputs. Organizations can include static source ([SAST](#)) and dynamic application test ([DAST](#)) tools into the CI/CD pipeline to identify newly introduced injection flaws prior to production deployment.

How to Prevent

Preventing injection requires keeping data separate from commands and queries.

- The preferred option is to use a safe API, which avoids the use of the interpreter entirely or provides a parameterized interface, or migrate to use Object Relational Mapping Tools (ORMs). **Note:** Even when parameterized, stored procedures can still introduce SQL injection if PL/SQL or T-SQL concatenates queries and data, or executes hostile data with EXECUTE IMMEDIATE or exec().
- Use positive or "whitelist" server-side input validation. This is not a complete defense as many applications require special characters, such as text areas or APIs for mobile applications.
- For any residual dynamic queries, escape special characters using the specific escape syntax for that interpreter. **Note:** SQL structure such as table names, column names, and so on cannot be escaped, and thus user-supplied structure names are dangerous. This is a common issue in report-writing software.
- Use LIMIT and other SQL controls within queries to prevent mass disclosure of records in case of SQL injection.

Example Attack Scenarios

Scenario #1: An application uses untrusted data in the construction of the following **vulnerable** SQL call:

```
String query = "SELECT * FROM accounts WHERE custID=" + request.getParameter("id") + "";
```

Scenario #2: Similarly, an application's blind trust in frameworks may result in queries that are still vulnerable, (e.g. Hibernate Query Language (HQL)):

```
Query HQLQuery = session.createQuery("FROM accounts WHERE custID=" + request.getParameter("id") + "");
```

In both cases, the attacker modifies the 'id' parameter value in their browser to send: ' or '1'=1. For example:

```
http://example.com/app/accountView?id=' or '1'=1
```

This changes the meaning of both queries to return all the records from the accounts table. More dangerous attacks could modify or delete data, or even invoke stored procedures.

References

OWASP

- [OWASP Proactive Controls: Parameterize Queries](#)
- [OWASP ASVS: V5 Input Validation and Encoding](#)
- [OWASP Testing Guide: SQL Injection, Command Injection, ORM Injection](#)
- [OWASP Cheat Sheet: Injection Prevention](#)
- [OWASP Cheat Sheet: SQL Injection Prevention](#)
- [OWASP Cheat Sheet: Injection Prevention in Java](#)
- [OWASP Cheat Sheet: Query Parameterization](#)
- [OWASP Automated Threats to Web Applications – OAT-014](#)

External

- [CWE-77: Command Injection](#)
- [CWE-89: SQL Injection](#)
- [CWE-564: Hibernate Injection](#)
- [CWE-917: Expression Language Injection](#)
- [PortSwigger: Server-side template injection](#)

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

20

OWASP Top 10

Risques des applications Web



OWASP Top 10 - 2013	➔	OWASP Top 10 - 2017
A1 – Injection	➔	A1:2017-Injection
A2 – Broken Authentication and Session Management	➔	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	➡	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	➡	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	➔	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	⊗	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	➔	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	⊗	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

OWASP Top 10 2017

Risques des applications Web



A1	Injection	A6	Mauvaise configuration de sécurité
A2	Contournement des mécanismes de gestion d'authentification et de session	A7	Cross-Site Scripting (XSS)
A3	Exposition de données sensibles	A8	Désérialisation non sécuritaire
A4	Entités XML eXternes (XXE)	A9	Utilisation de composants avec des vulnérabilités connues
A5	Contournement des contrôles d'accès	A10	Journalisation et surveillance insuffisantes



**2 semaines
d'hacking éthique**

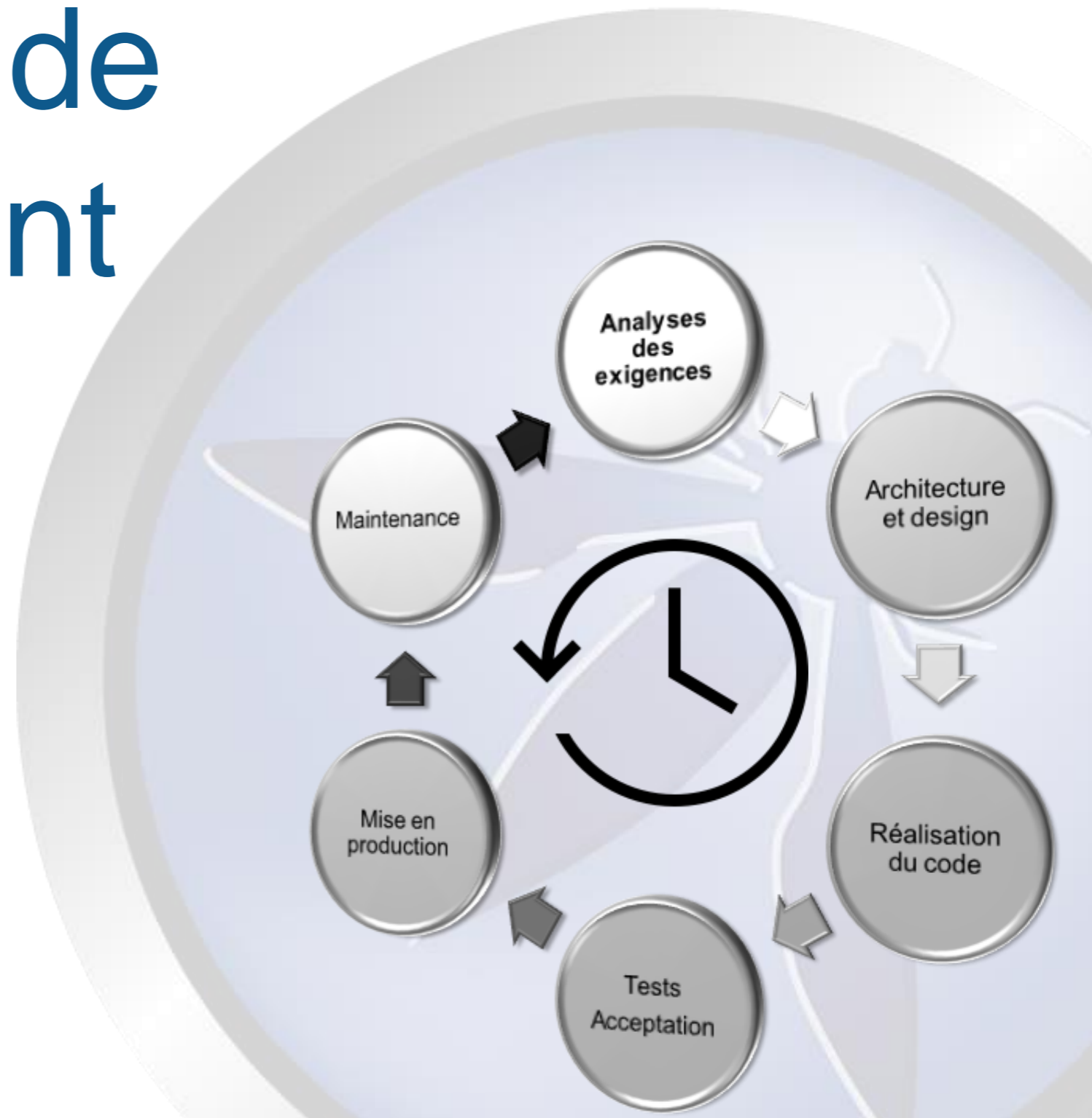
**Lacunes dans
la logique
d'affaire**

**Erreurs de
sécurité**

**Lacunes
dans le
code**

**10 années-personnes
de développement**

Intégrer la sécurité dans le cycle de développement

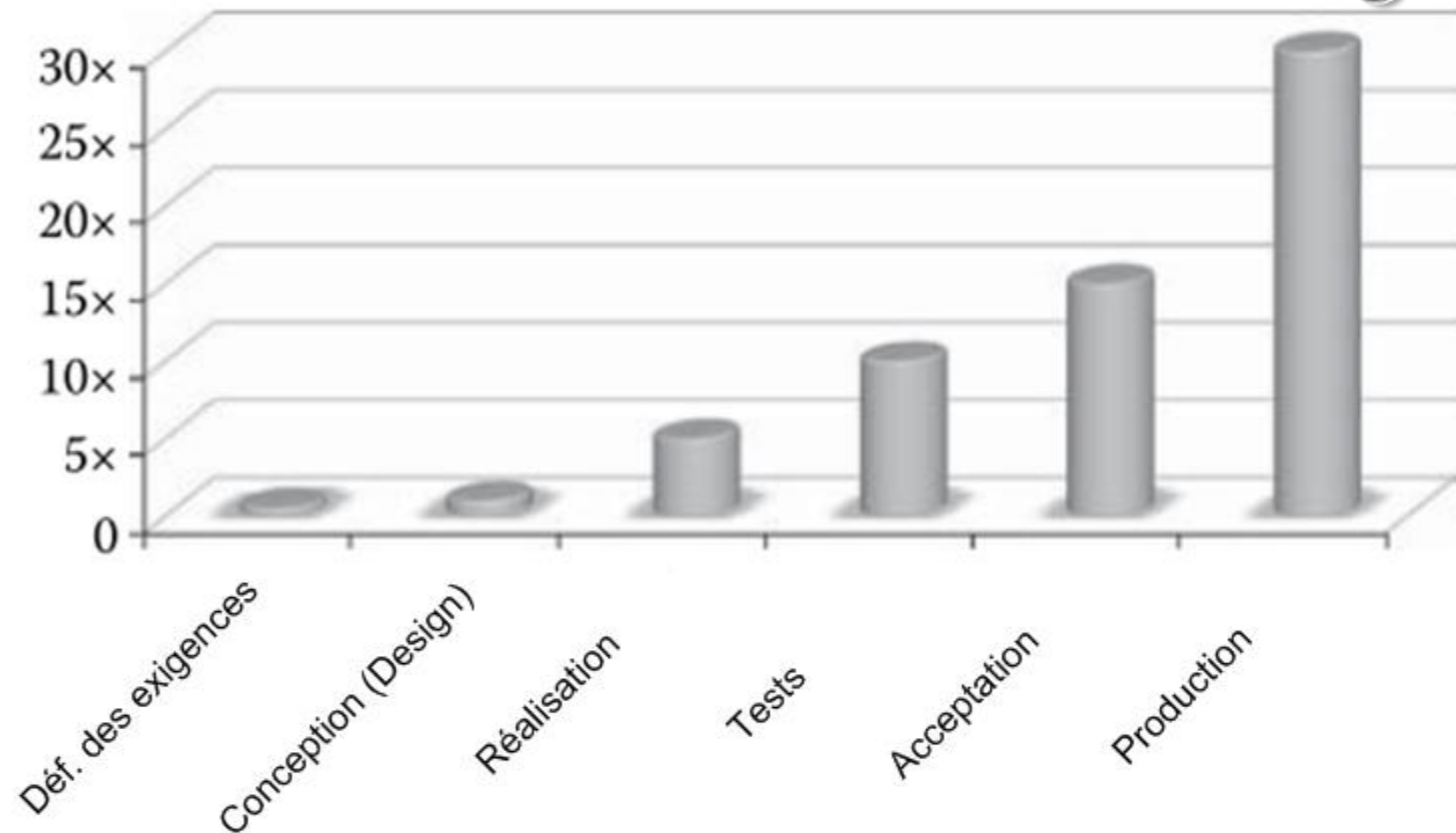


Sécurité dans le cycle de développement



Observation:

Les **coûts** liés aux **corrections** des vulnérabilités **augmentent** de façon **exponentielle** plus les vulnérabilités sont découvertes tardivement dans le cycle de développement...

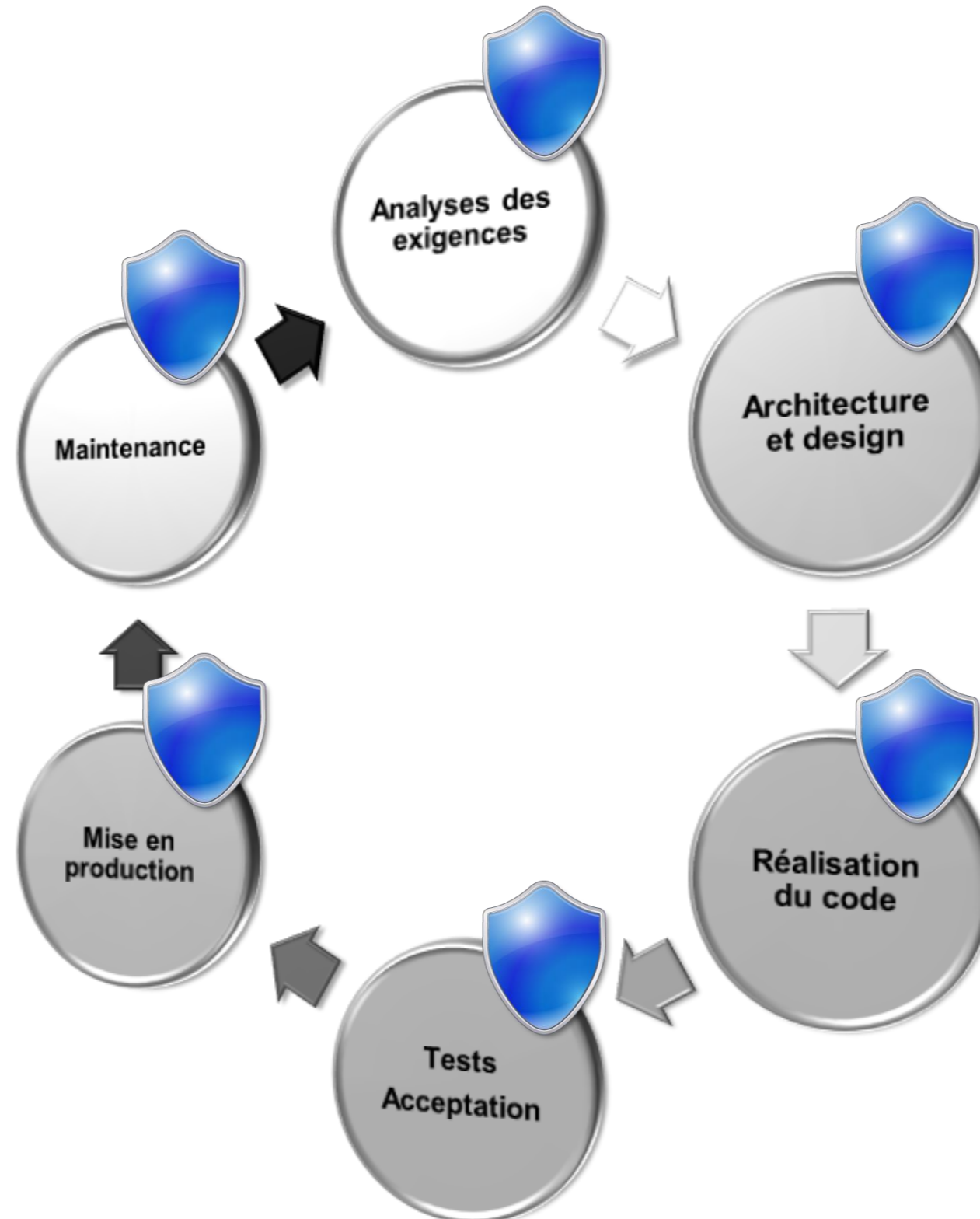


Évidence:

- **Prise en charge des enjeux de sécurité plus tôt et tout au long du cycle de développement**

*Approche prônée par OWASP, NIST, Microsoft et **plusieurs** autres organisations*

Phases d'un cycle simplifié...



OWASP SAMM (Software Assurance Maturity Model)



- **Méthodologie** aidant à **définir une approche adaptée** de prise en charge de la sécurité applicative **en fonction du contexte particulier** et des **risques** de l'organisation
- Couvre **le cycle de vie** des applications
- **Modèles par type d'industrie**
- **Très logique, très pragmatique**

4 domaines



SAMM – Disciplines de sécurité

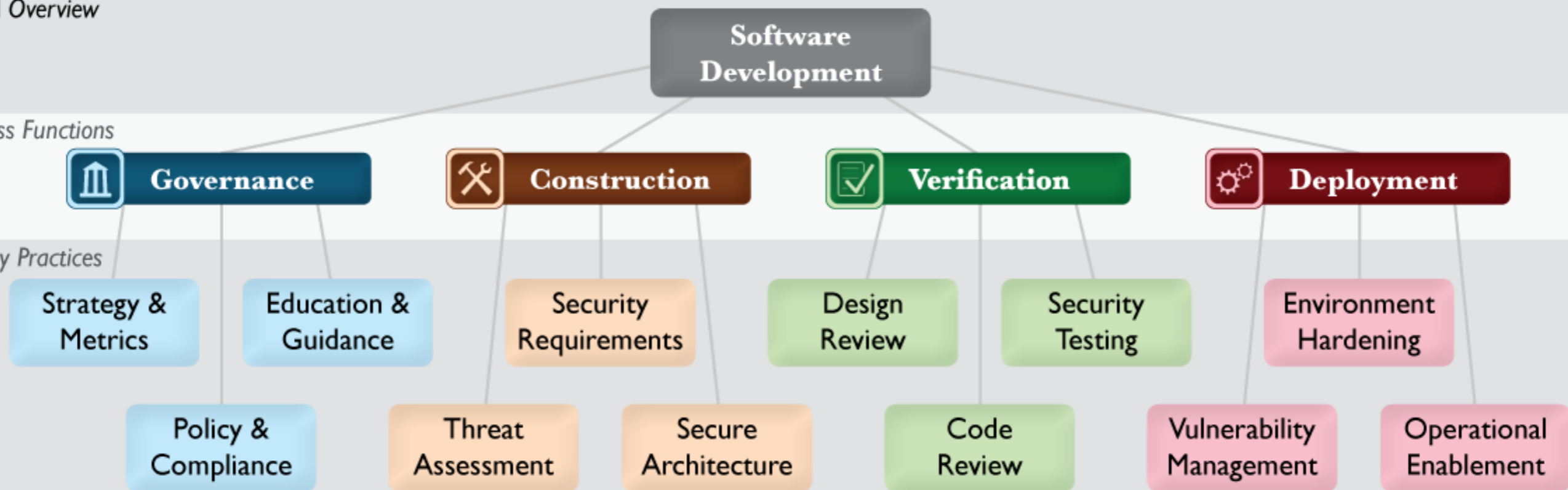


- **3 disciplines** par domaines de sécurité, chaque discipline incorpore des **activités de sécurité** (environ de 5-7 chacune)
- Les disciplines **couvrent toute la surface** de l'assurance sécurité des applications

SAMM Overview

Business Functions

Security Practices



SAMM – Questionnaire de maturité



0
1
2
3

Fonction		Pratiques	Activités	Réponse	Cote	Cible
Stratégie	GSM1-1		Existe-il un programme d'assurance de la sécurité des applications/systèmes déjà en place?			
	GSM1-1.1		Est-ce que la plupart des gens de développement connaissent les plans d'avenir pour le programme d'assurance de la sécurité?			
	GSM1-2		Est-ce que la plupart des parties prenantes ont connaissance des risques de sécurité de l'organisation?			

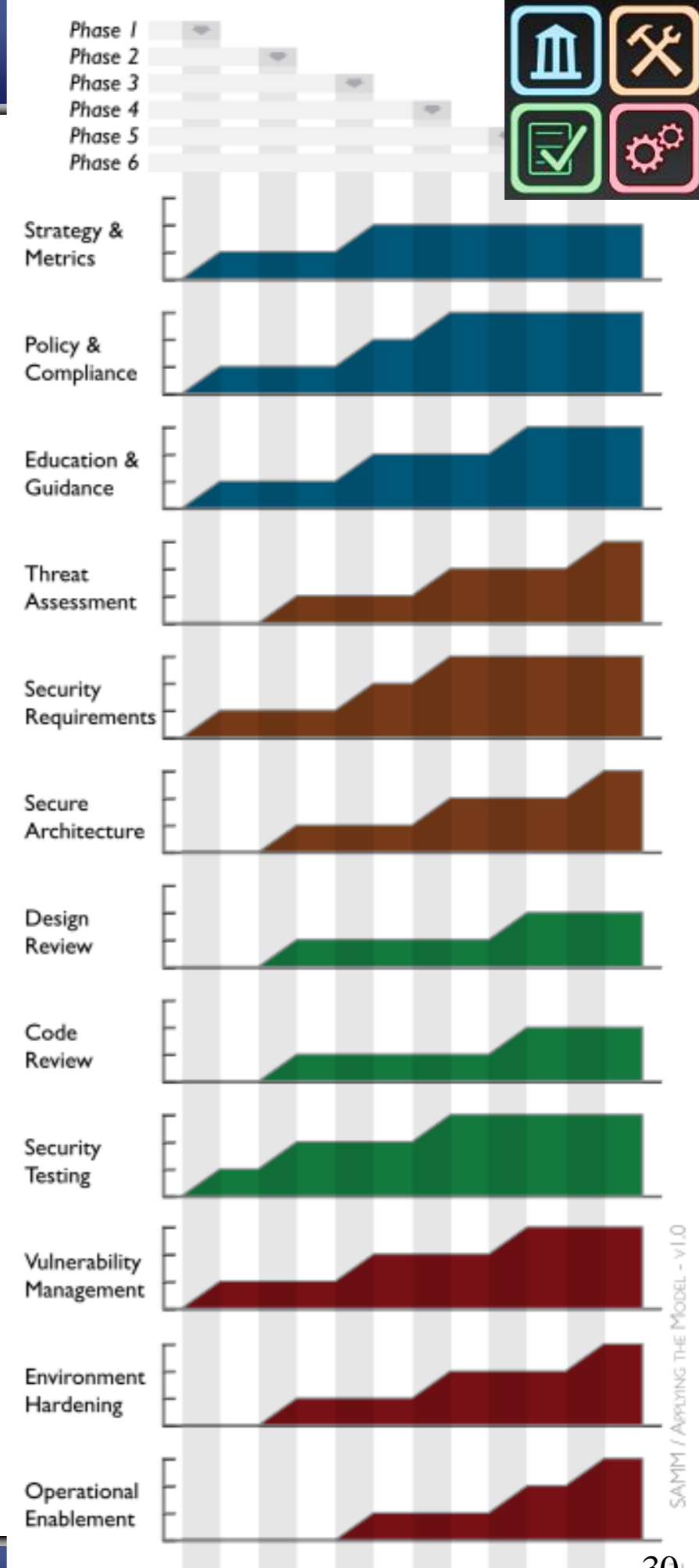
Fonction		Pratiques	Activités	Réponse	Cote	Cible
Construction	Modélisation des menaces	CTA1-1	Est-ce que la plupart des projets considèrent et documentent les menaces probables?			
		CTA1-1.1	Est-ce que l'organisation comprend et documente les types d'attaques auxquelles elle est confrontée?			
		CTA2-1	Est-ce que les projets analysent les exigences fonctionnelles afin d'identifier les abus probables ?			
		CTA2-2	Est-ce que les équipes de projet utilisent une méthode de notation pour comparer les menaces?			
		CTA2-3	Est-ce que les parties prenantes sont conscientes des menaces et niveaux de risques pertinents?			
		CTA3-1	Est-ce que les équipes de projet considèrent spécifiquement les risques des logiciels externes?			
	Requis de sécurité	CSR1-1	Est-ce que la plupart des équipes de projet énoncent certaines exigences de sécurité au cours du développement?			
		CSR1-1.1	Est-ce que les équipes de projet définissent leurs exigences à partir des meilleures pratiques de l'industrie?			
		CSR2-1	Est-ce que les parties prenantes ont défini des matrices de contrôle d'accès pour les projets concernés?			
		CSR2-2	Est-ce que les équipes de projet précisent les exigences en fonction des constats formulés lors des autres projets)?			
		CSR3-1	Est-ce que l'organisation examine les ententes fournisseurs en égard à ses besoins de sécurité?			
		CSR3-2	Est-ce que les exigences de sécurité spécifiées par les équipes de projet sont auditées?			
	Architecture de sécurité	CSA1-1	Est-ce que l'on fournit aux équipes de projet une liste de composants tiers recommandés?			
		CSA1-2	Pour la plupart des projets, y a-t-il au moins un intervenant qui est conscientisé à la sécurité et qui applique les exigences de sécurité?			
		CSA2-1	Est-ce que l'on fournit et communique aux équipes de projet des services communs de sécurité et des guides de conception?			
CSA2-2		Est-ce que l'on fournit aux équipes de projet des patrons de conception (design patterns) normatifs en fonction des exigences de sécurité?				
CSA3-1		Est-ce que les équipes de projet développent à partir de plateformes et de composants communs éprouvés?				
	CSA3-1.1	Est-ce que les équipes de projet font l'objet de vérifications afin de s'assurer qu'elles utilisent les composants recommandés?				
Délegation opérationnelle	DEH3-1	Est-ce que les parties prenantes connaissent les options possibles d'outils supplémentaires pouvant permettre de protéger l'application en production?				
	DEH3-2	Est-ce que des vérifications régulières assurent que l'état de santé de l'environnement de production de la plupart des projets respecte des critères de sécurité de base?				
	DOE1-1	Est-ce que vous livrez des notes de sécurité avec la plupart des livraisons logicielles?				
	DOE1-2	Est-ce que les alertes de sécurité et les conditions d'erreurs sont documentées pour la plupart des projets?				
	DOE2-1	Est-ce que la plupart des projets utilisent un processus de gestion du changement bien compris de tous?		0		
	DOE2-2	Est-ce que les équipes de projet fournissent un guide de sécurité opérationnel avec chaque livraison du produit?				
	DOE3-1	Est-ce que la plupart des projets sont audités afin de vérifier que chaque livraison contienne les informations appropriées pour la sécurité opérationnelle?				
DOE3-2	Est-ce qu'un processus formel de signature des composants est appliqué sur une base régulière?					

SAMM – Amélioration itérative



Strategy & Metrics

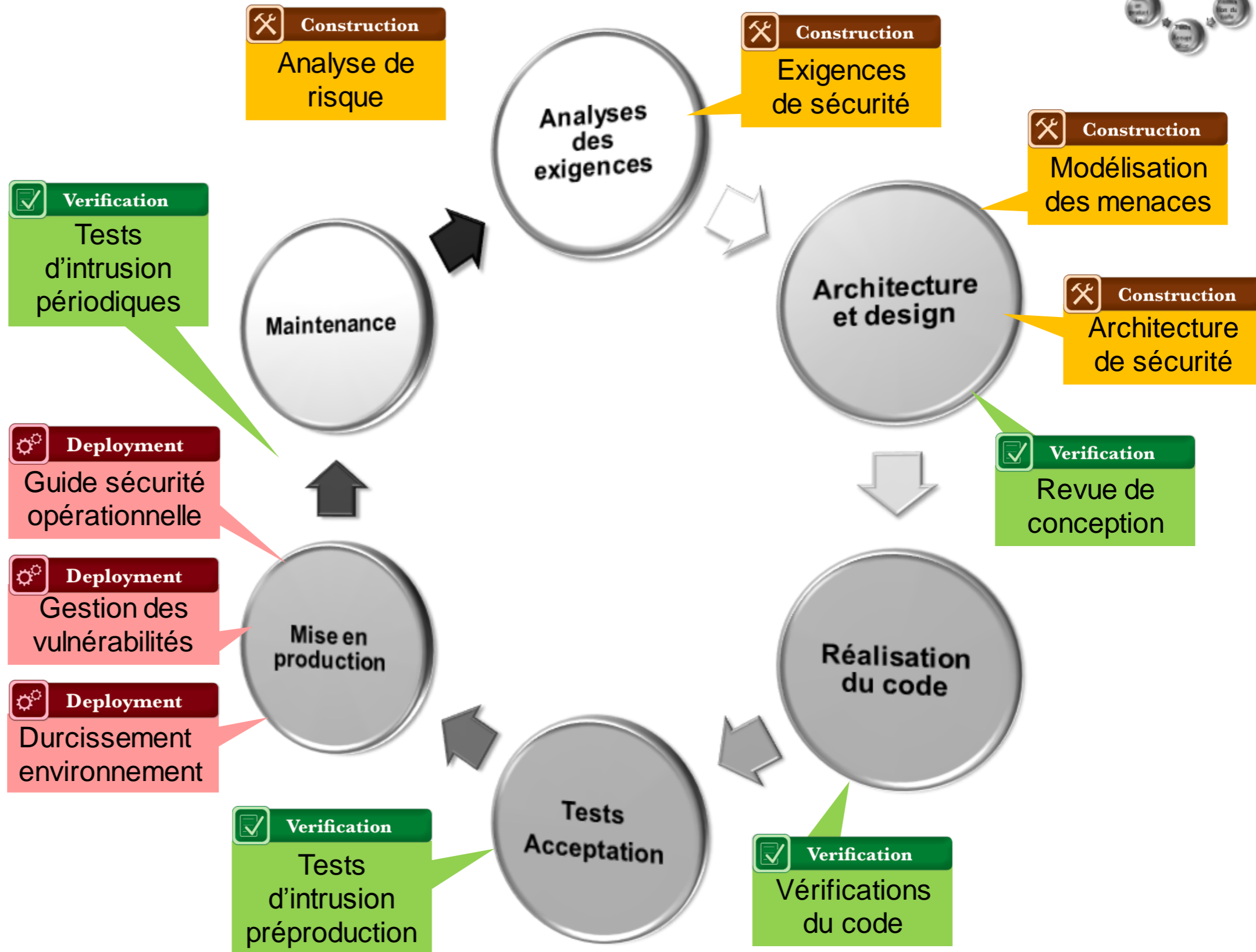
- Amélioration itérative = phase
- On définit la **stratégie** et les **métriques**
 - Enchaînement logique en fonction des interrelations et dépendances
 - Objectifs mesurables découpés en phases
 - Jusqu'à la maturité désirée



Intégration d'activités de sécurité



- Governance**
Catégorisation des données et applications
- Governance**
Guide politiques et conformité
- Governance**
Formation
- Governance**
Programme assurance sécurité
- Governance**
Rôles et responsabilités



Autres outils OWASP



OWASP Pro Active Controls

 Construction
Exigences de sécurité



 Construction
Architecture de sécurité

Techniques de base à implanter pour prévenir les vulnérabilités du **OWASP Top 10**

À **utiliser complètement** dans **toutes les applications**

Écrit pour les développeurs par des développeurs



➤ **Utile pour tous les gens impliqués dans développement logiciel**

C1 Définir les requis de sécurité	C2 Promouvoir les librairies et plateformes sécurisées	C3 Sécuriser les bases de données	C4 Encoder et échapper les données	C5 Valider tous les intrants
C6 Authentifier les identités	C7 Contrôler les accès	C8 Protéger partout les données sensibles	C9 Implémenter et surveiller les journaux de sécurité	C10 Gérer toutes les erreurs et exceptions

OWASP Cheat Sheets (Aide-mémoire)



V - T - E
... ..

Cheat Sheets

[Collapse]

 **Construction**
Architecture
de sécurité

Developer / Builder

 **Verification**
Revue de
conception

Assessment / Breaker

Mobile

OpSec / Defender

Draft and Beta

AJAX Security Cheat Sheet · Authentication (ES) · Choosing and Using Security Questions · Clickjacking Defense · C-Based Toolchain Hardening · Cross-Site Request Forgery (CSRF) Prevention · Cryptographic Storage · DOM based XSS Prevention · Forgot Password · HTML5 Security · Input Validation · JAAS · 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


 **Verification**
Tests
d'intrusion


Attack Surface Analysis · XSS Filter Evasion · REST Assessment · Web Application Security Testing


IOS Developer · Mobile Jailbreaking

Virtual Patching

3rd Party Javascript Management · Access Control · Android Testing · Application Security Architecture · Business Logic Security · Injection Prevention Cheat Sheet · PHP Security · Secure Coding · Secure SDLC · Threat Modeling · Grails Secure Code Review · IOS Application Security Testing · Key Management · Insecure Direct Object Reference Prevention · Content Security Policy

 **Deployment**
Durcissement
environnement

 **Verification**
Vérifications
du code

 **Construction**
Modélisation
des menaces

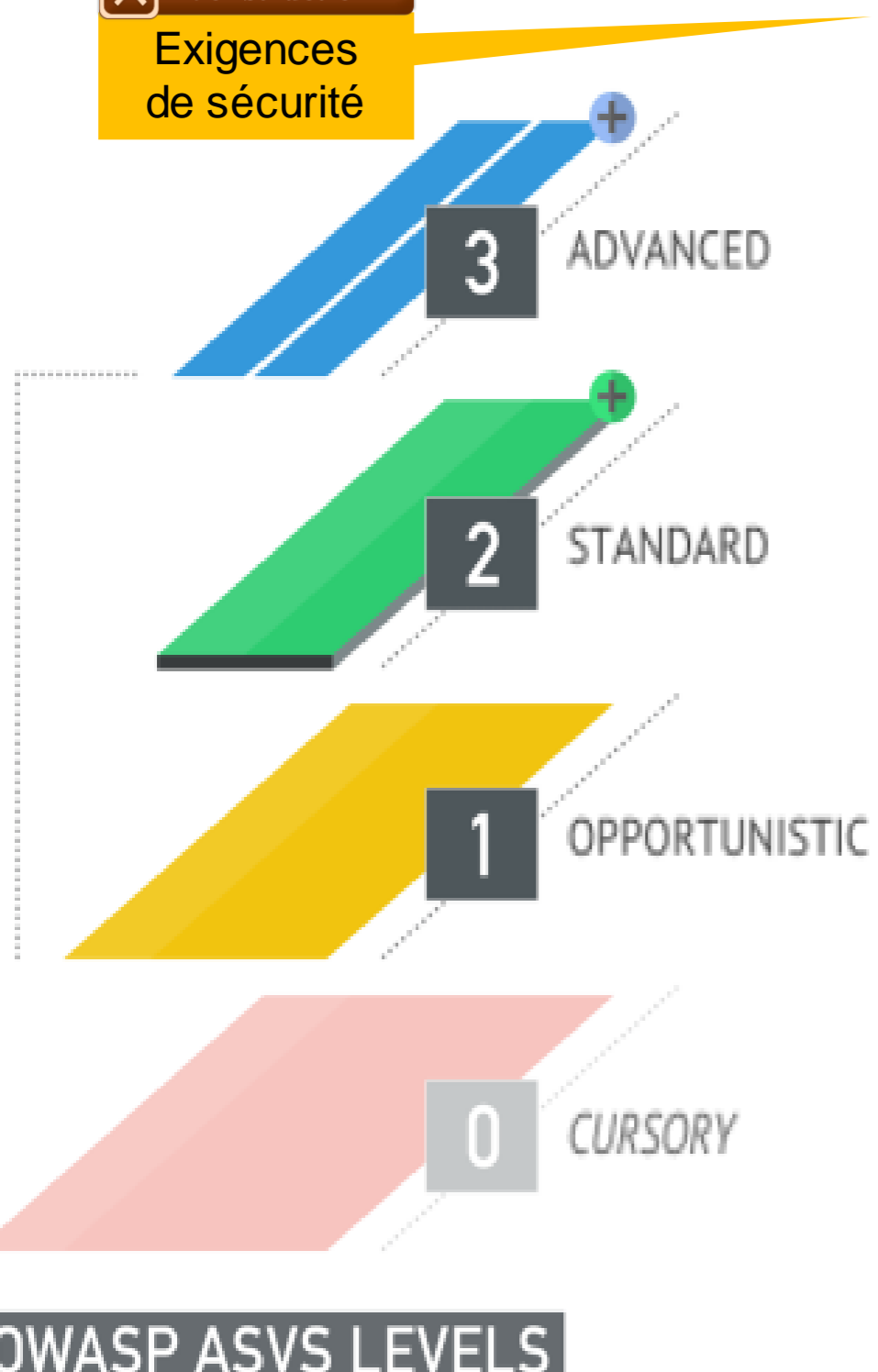
OWASP Application Security Verification Standard



Construction
Exigences de sécurité

Requirements

Verification
Revue de conception



#	Description	1	2	3	Since
2.1	Verify all pages and resources by default require authentication except those specifically intended to be public (Principle of complete mediation).	✓	✓	✓	1.0
2.2	Verify that all password fields do not echo the user's password when it is entered.	✓	✓	✓	1.0
2.4	Verify all authentication controls are enforced on the server side.	✓	✓	✓	1.0
2.6	Verify all authentication controls fail securely to ensure attackers cannot log in.	✓	✓	✓	1.0
2.7	Verify password entry fields allow, or encourage, the use of passphrases, and do not prevent long passphrases/highly complex passwords being entered.	✓	✓	✓	3.0
2.8	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.	✓	✓	✓	2.0
2.9	Verify that the changing password functionality includes the old password, the new password, and a password confirmation.	✓	✓	✓	1.0
2.12	Verify that all suspicious authentication decisions are logged. This should include requests with relevant metadata needed for security investigations.		✓	✓	2.0
2.13	Verify that account passwords make use of a sufficient strength encryption routine and that it withstands brute force attack against the encryption routine.		✓	✓	3.0
2.16	Verify that credentials are transported using a suitable encrypted link and that all pages/functions that require a user to enter credentials are done so using an encrypted link.	✓	✓	✓	3.0

OWASP ASVS LEVELS

OWASP Security Knowledge Framework



Base de connaissance de sécurité applicative



Security Knowledge Framework

• Descriptions et guides d'implémentation sur

- **Principes** de développement sécuritaire
- **Fonctionnalités** de sécurité
- **Vulnérabilités** et problèmes communs



Governance

Formation

• **Exemples de code** pour plusieurs fonctions de sécurité!

• Génération de listes vérifications (ASVS/MASVS)

- Selon votre niveau ASVS à atteindre
- En fonction des fonctionnalités que vous devez développer



Construction

Exigences
de sécurité

</> Code Language

PHP

C#/.net

JAVA

Py-Flask

Py-Django

Ruby on Rails

Go

OWASP Dependency-Check



Utilitaire permettant d'**analyser les applications** et les **librairies externes** et de vérifier si elles contiennent des **vulnérabilités connues** dans la base de données du NIST

Java


.NET

Ruby

Node.js

Python

C/C++

 **Deployment**
Gestion des vulnérabilités

DependencyCheck Result



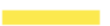

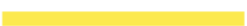
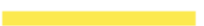


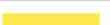
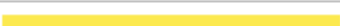
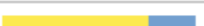
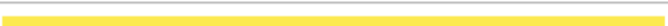


Warnings Trend

All Warnings	New Warnings	Fixed Warnings
153	138	0

Summary

Total	High Priority	Normal Priority	Low Priority
153	24	111	18

Details

Files	Categories	Types	Warnings	Details	New	High	Normal	Low
	Category		Total	Distribution				
	CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer		5					
	CWE-134 Uncontrolled Format String		1					
	CWE-189 Numeric Errors		2					
	CWE-20 Improper Input Validation		7					
	CWE-200 Information Exposure		5					
	CWE-22 Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')		4					
	CWE-264 Permissions, Privileges, and Access Controls		4					
	CWE-287 Improper Authentication		2					
	CWE-310 Cryptographic Issues		2					
	CWE-399 Resource Management Errors		7					
	CWE-59 Improper Link Resolution Before File Access ('Link Following')		4					
	CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')		14					
	CWE-89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')		2					
	CWE-94 Improper Control of Generation of Code ('Code Injection')		10					
	Total		153					



OWASP Dependency-Track

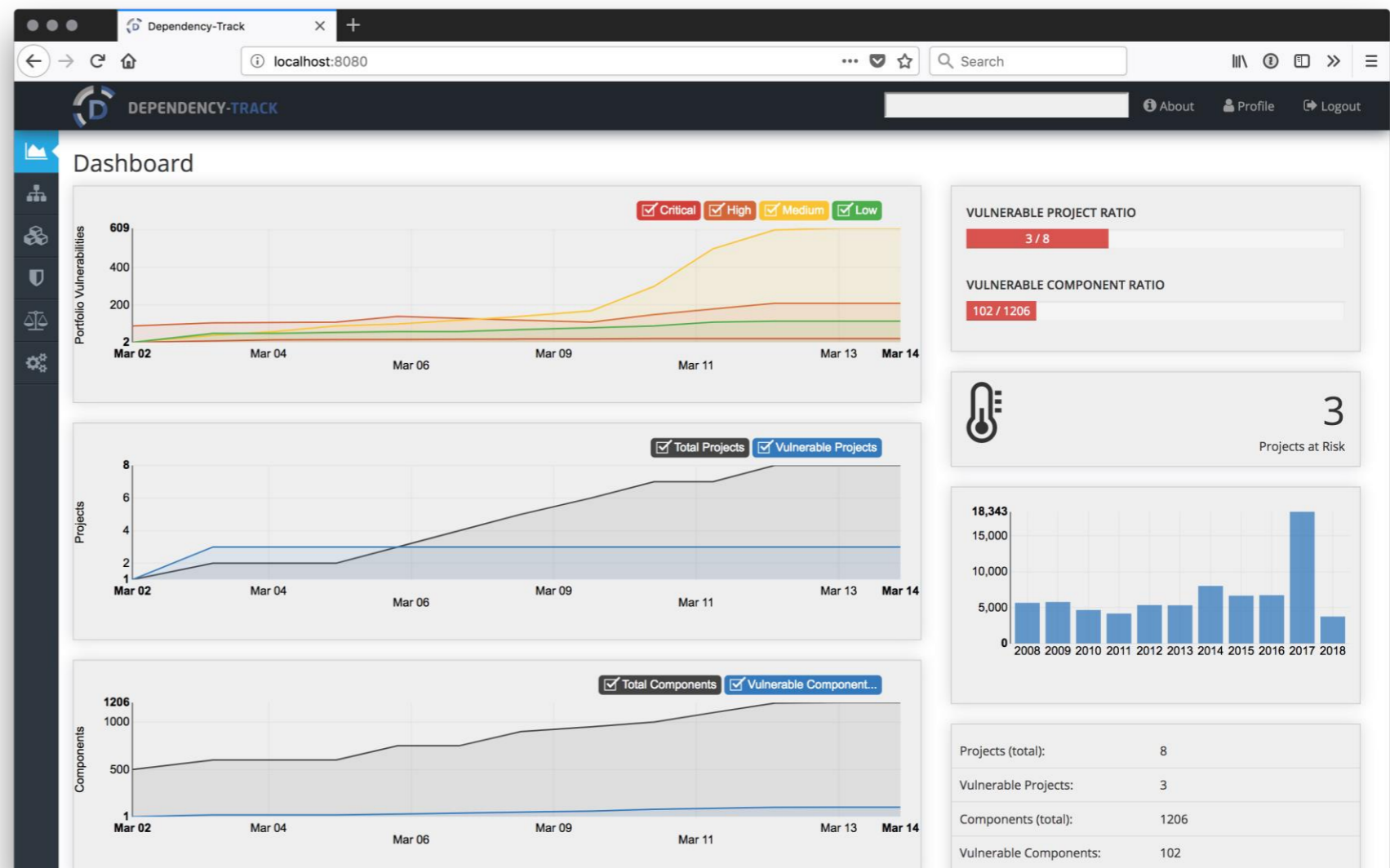
Deployment
Gestion des
vulnérabilités



Utilitaire permettant de **garder trace des composants tiers utilisés** de toutes les applications qu'une organisation utilise (vision globale)

Surveille toutes les applications de son portefeuille afin d'**identifier proactivement** les **vulnérabilités des composants**

- **Conçu pour être utilisé** dans un environnement **DevOps automatisé**
- Offre une **vue des projets à risque**
- Vulnérabilités / projet
- Quelle vulnérabilité affecte quel(s) projet(s)




OWASP Code Review Guide




Guide pour implanter une activité de revue de code:

- Explique la méthodologie et processus
- Couvre le OWASP Top 10 (2013) en détail avec plusieurs exemples
- Comprend une liste de vérification
- Exemples de modélisation des menaces

 **Governance**
Rôles et responsabilités

 **Verification**
Vérifications du code

 **Construction**
Modélisation des menaces

CODE
REVIEW
GUIDE

2.0



OWASP ZAP



The screenshot shows the OWASP ZAP interface. A green callout box with a checkmark icon and the text "Verification" and "Vérifications du code" points to the "Verification" button in the top toolbar. Another green callout box with a checkmark icon and the text "Verification" and "Tests d'intrusion" points to the "Scan" button in the bottom toolbar. A large blue lightning bolt icon is overlaid on the right side of the interface.

Verification
Vérifications du code

Verification
Tests d'intrusion

Applications Places Sun Dec 8, root
Untitled Session - OWASP ZAP

File Edit View Analyse Report Tools Online Help

Standard mode

Sites

Quick Start Request Response Break

Sites

- http://192.168.147.133
 - GET:mutillidae
 - GET:robots.txt
 - mutillidae
 - GET:index.php(page)
 - GET:set-up-database.php
 - GET:index.php(do,page)
 - GET:index.php
 - GET:index.php(page,username)
 - GET:fra
 - GET:fav
 - document
 - images
 - POST:in
 - GET:inc
 - GET:inc
 - POST:in
 - POST:in

History Search

Site: 192.168.147.133

Processed	Method	Flags
●	GET	SEED
●	GET	SEED
●	GET	
●	GET	
●	GET	
●	GET	
●	GET	
●	GET	
●	GET	
●	GET	
●	GET	

Alerts 0 0 2 1

Current Scans 0 | URIs Found: 195

Current Scans 0 0 0 0 0 0 0 0

OWASP Security Shepherd & WebGoat



2 plateformes d'apprentissage de sécurité applicative

- **OWASP Security Shepherd new v3.1!**

- Applications **Web** et **mobiles**
- Leçons et « challenges »
- 70 niveaux (de novice à expert)
- Parfait pour les classes et les tournois CTF



- **OWASP WebGoat new v8!**

- Application Java délibérément non sécurisée
- Focus sur l'apprentissage « *Find and Fix* »
- Leçons et indices
- « Challenges » réalistes

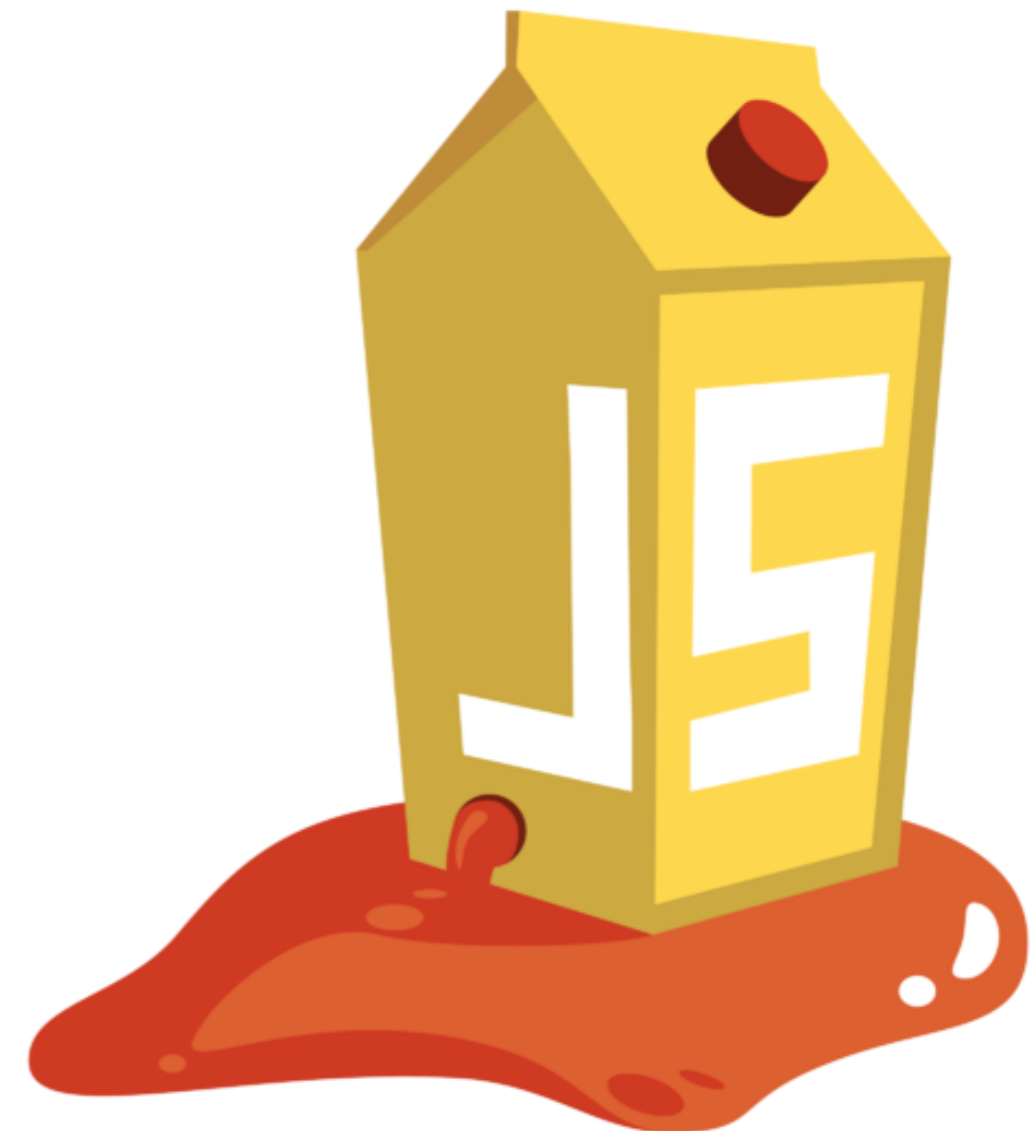
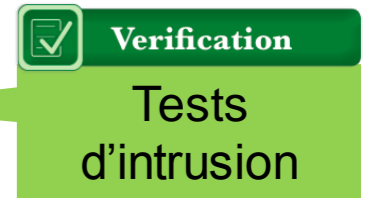


OWASP Juice Shop



- Application faite en JavaScript
 - Node.js, Express, Angular
- 60+ challenges (de simple à diabolique)
- Possibilité de faire des CTF (custom)
- Progrès sur un tableau de pointage
- La BD se répare à chaque redémarrage

- Moins orienté « leçons » que *WebGoat* et *Security Shepherd*
- Communauté très active



OWASP ModSecurity Core Rule Set



Règles de **détection d'attaque génériques** pour pare-feu applicatif (WAF) ModSecurity (open-source) ou compatibles

- Protection **OWASP Top 10 et +**
- Réputation IP malicieux
- DoS pour les applications
- Détection des scanners
- Détection d'injection de commande OS
- 4 niveaux de paranoia
 - Réduction des faux positifs
 - Attention: Faux positifs augmentent avec le niveau de paranoia



 **Deployment**
Durcissement environnement


OWASP
ModSecurity
Core Rule Set
THE 1ST LINE OF DEFENSE

➤ DevOps: intégrez-le dans les tests de votre pipeline!

OWASP ModSecurity Core Rule Set



INDICATIONS : Pour le soulagement rapide des **symptômes** de vos applications Web affectées par des **vulnérabilités applicatives**. Offre une **couche de protection supplémentaire** pour défendre les applications Web.

 **Deployment**
Durcissement environnement

POSOLOGIE : Incorporer le « Core Rule Set » à vos pare-feux. Réadministrer sans tarder à chaque mise à jour.

INGRÉDIENTS ACTIFS: Ensemble de **règles de détection** d'attaques pour le pare-feu applicatif « open source » et multiplateforme « ModSecurity ».

 **MISE EN GARDE** : Peut procurer un **faux sentiment de sécurité**. Ne constitue **pas un remède** à la **correction** de vos applications.



OWASP
ModSecurity
Core Rule Set
THE 1ST LINE OF DEFENSE

OWASP Ville de Québec

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



Présentations gratuites et orientées pour les gens de développement :

2018

- Survol de la sécurité de Microsoft Azure
- Pourquoi le SPA (Single Page Application) est faillible?
- Introduction OAuth 2.0 et OpenId Connect 1.0
- La sécurité dans un environnement docker / kubernetes / cloud avec microservices
- Explication des nouveaux risques "OWASP Top 10 2017"
- La sécurité web pour les développeurs .NET

2017

- Comment intégrer la sécurité dans un cycle de développement logiciel rapide
- Comprendre l'usage de la crypto pour les développeurs
- Comment bien gérer les incidents de sécurité
- Outils de modélisation des menaces
- Les logiciels malveillants et les objets connectés (IoT)
- Rétro-ingénierie de protocoles crypto

Partenariats pour faire connaître la mission d'OWASP:

- Semaine Numérique de Québec et SeQCure 2019
- **>> Journée ISACA-Québec 2018 : Sécurité des systèmes d'information dès leur conception <<**
- Hackfest (2012-2018)
 - Événements « Capture The Flag » OWASP au HackFest (2014-2018)
 - Conférence sur le Top 10 des défenses Web (2012)
- CQSI 2018
- Journée sur la sécurité applicative à l'Université Laval 2017
- ...

Conférence OWASP du 5 décembre!



CTF What? Présentation du monde CTF! « Capture The Flag »

Franck Desert

Analyste en sécurité applicative
CGI Inc

Description

Attaque/Défense, Jeopardy, Puzzle Technique, Retro ou New Tech, nous allons vous raconter l'histoire, l'existant, ce qui s'en vient dans le monde du CTF et présenter également le devant et le derrière du miroir. Mais ne vous y trompez pas! Le CTF devient la nouvelle façon de se former, de rester sur le "Edge", de recruter, etc. Donc, que vous soyez développeur, pentester, defender, chargé de projet, manager ou tout simplement curieux, cette présentation est pour vous! En effet, l'apprentissage ludique qu'est le CTF doit prendre une place importante dans votre quotidien, que vous le fassiez à titre personnel ou afin de convaincre votre entreprise de l'intégrer, il est temps de s'y mettre!

Si vous le désirez, apportez vos portables! Il y aura quelques petits CTF au cours de la présentation!

Lieu

Local A-225 du Cégep de Sainte-Foy

Horaire

18:15 - 18:30 Accueil

18:30 - 20:30 Conférence (spécial 2h!)

- 1. Budgéter et intégrer la sécurité tôt dans votre cycle de développement**
- 2. Avec tout le matériel OWASP disponible gratuitement, vous n'avez aucune raison de ne pas savoir que faire!**



Pour devenir membre ou contribuer



Devenir membre pour un don annuel de:

- Individuel \$50 USD
- Corporatif \$5000 USD

Permet à OWASP d'offrir tout le matériel gratuit et de continuer de supporter les initiatives:

- Projets et outils
- Conférences
- Podcasts, bourses et coordination des activités mondiales...

<https://www.owasp.org/index.php/Membership>

Merci!

patrick.leclerc@owasp.org

OWASP Québec

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

