Journée sur la sécurité applicative
Université Laval
29 novembre 2017

The OWASP Foundation
www.owasp.org

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

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

23 ans d'expérience TI
en développement :

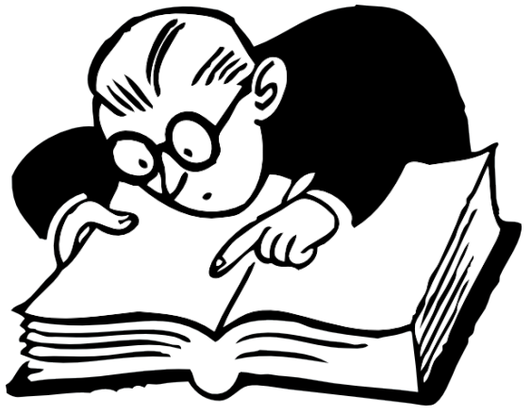
- Architecture logicielle et de sécurité

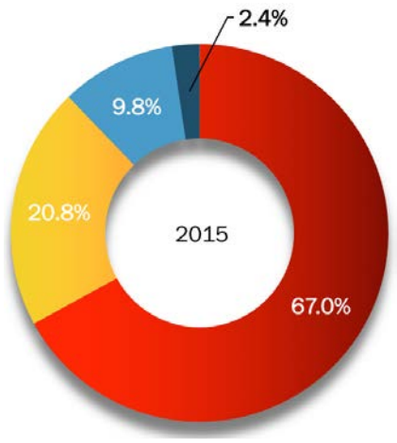
➢ Conseiller en architecture logicielle désormais dédié à la sécurité des applications

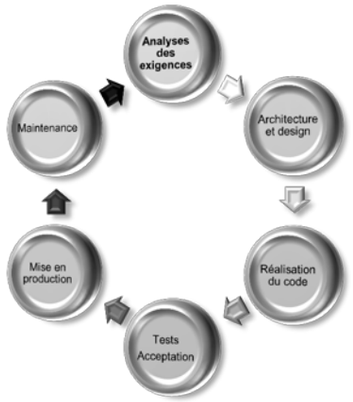
Leader du chapitre
OWASP Ville de Québec

Conseiller en sécurité des actifs informationnels à La Capitale

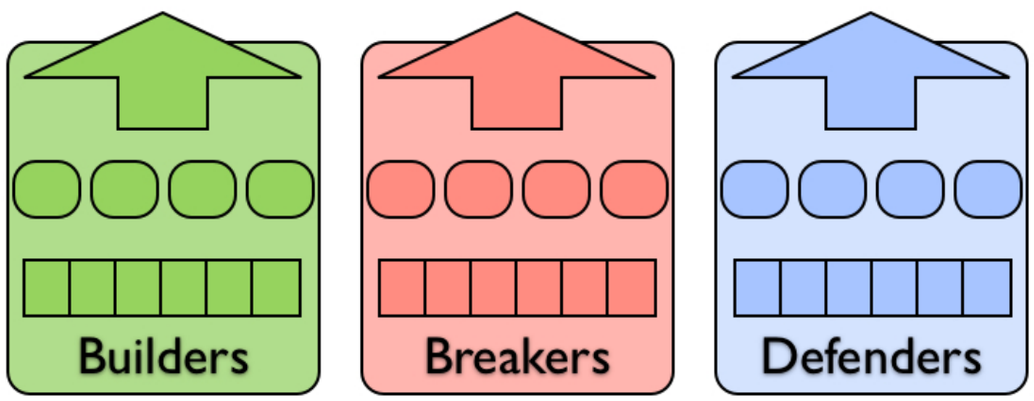
Plan de la présentation







Qu’est ce que la sécurité applicative?



Qu’est ce que la sécurité applicative?

Ce n’est pas que la couche 7 « Application » du modèle OSI…

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 **appliquées 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 couvre pas uniquement la portion logicielle. Elle couvre également tous les contrôles et mesures de sécurité impliqués dans le cycle de vie de l’application.**

Portée de la sécurité applicative

- **Regroupe** toutes les **activités** pour **assurer la sécurité** dans l'usage d'applications
- **A-Z**: De la conception à la terminaison de l'application!

…pas que les applications Web!

Application intranet / extranet

Systèmes client-serveur

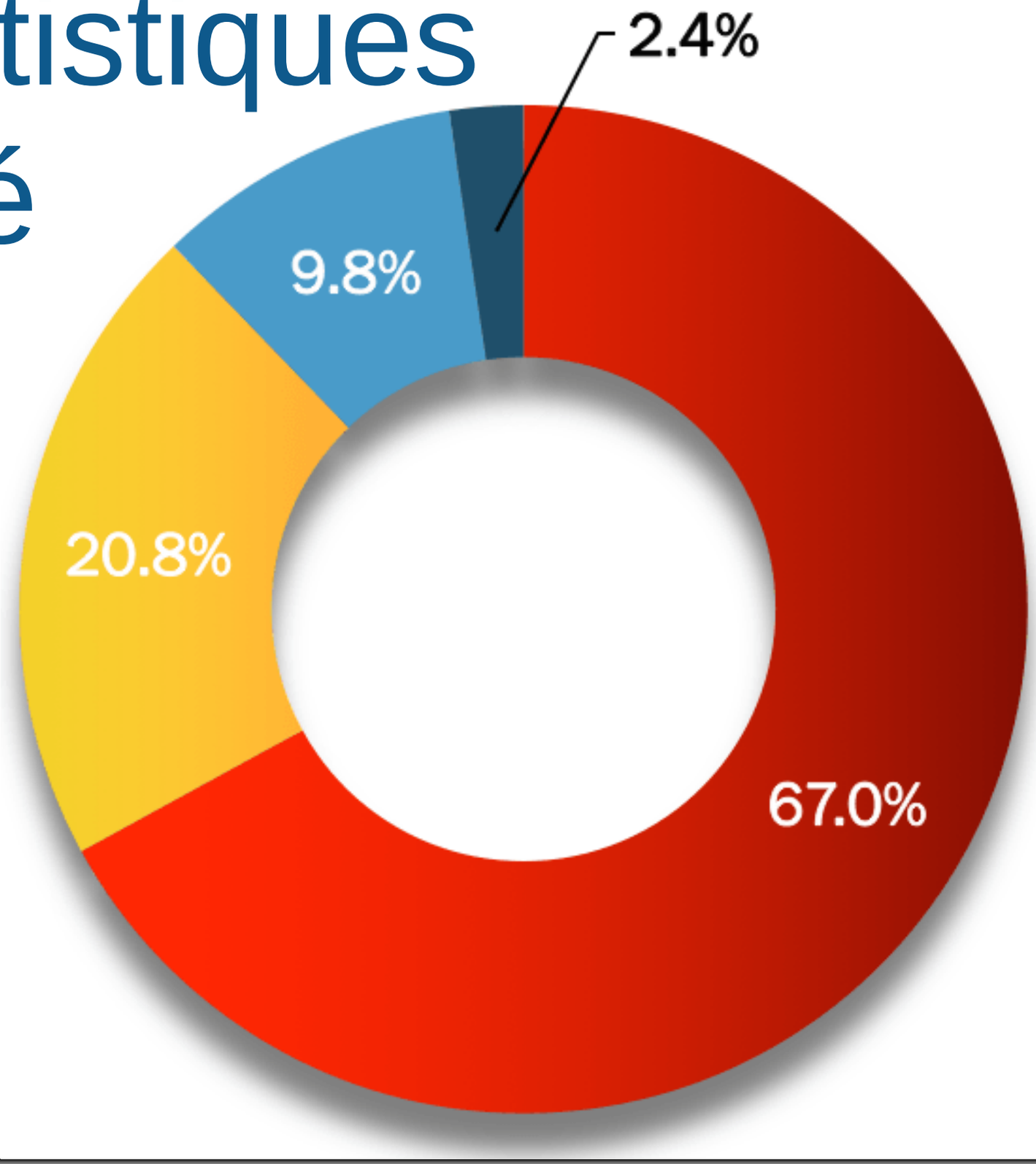
Applications mobiles / IoT

Clients riches

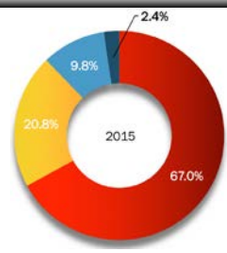
Systèmes distribués

Systèmes embarqués

Quelques statistiques sur la sécurité applicative



Quelques statistiques



84% des **attaques ciblent la couche applicative**

– *Checkmarx 2015*

75% des **vulnérabilités** se retrouvent **dans la couche applicative**

– *Checkmarx 2015 (Gartner disait la même chose en 2002…)*

70% des applications avaient **au moins une vulnérabilité** classifiée dans le **top 10 OWASP**

– *Veracode 2015*

15% des **applications Web** ont une **vulnérabilité critique ou élevée**

– *Edgescan report 2015*

Les **applications Web** sont responsables de **30% des brèches** confirmées

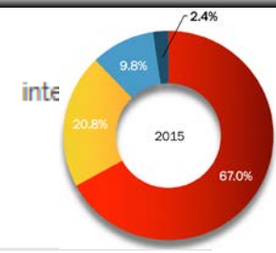
– *Verizon DBIR 2017*

96% des applications utilisent du **code open source**, et **2/3** de ces sources **comportent des vulnérabilités**

– *Black Duck OSSRA Report 2017*

World's Biggest Data Breaches

Selected losses greater than 30,000 records
(updated 10th Sep 2017)



YEAR

BUBBLE COLOUR YEAR METHOD OF LEAK BUBBLE SIZE NO OF RECORDS STOLEN DATA SENSITIVITY

2017

Brazzers
Cellebrite
DaFont
Bell
Clinton campaign
Dailymotion
85, 200, 000
Equifax
143, 000, 000
Lynda.com
owned by LinkedIn
9, 500, 000
Read a bit more
Three
Viacom
Yahoo
32, 000, 000
Weebly
43, 000, 000
Wendy's
VK
100, 544, 934
uTorrent

2016

Friend Finder Network
412, 000, 000
Minecraft
Spambot
711, 000, 000
Anthem
80, 000, 000
Mossack Fonseca
Banner Health
Carefirst
Fling
40, 000, 000
Linux Ubuntu forums
Sanrio
Turkish

2015

Les impacts…

Impacts pour l’organisation

- Perte de confidentialité
- Bris d’intégrité
- Fraudes
- Lourdes pertes financières
- Secrets d’entreprises

Selon une étude de *Ponemon Institute (2017)*

Au **Canada**:

- Coût moyen d’une brèche : **5,78 million**
- Coût moyen par identité volée: **255$**

https://www.canadianunderwriter.ca/insurance/canada-second-expensive-country-data-breaches-ibm-ponemon-institute-study-1004115487/

Impacts pour l’organisation

« 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** de marque,

et les **efforts de rétablissement**. »

Pourquoi les applications Web échappent 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

Hier:

- Applications ~~**internes**~~ **exposées** sur le Web avec **+/- les mêmes mesures**
- Maintenant accessibles à tous : usagers légitimes **et pirates informatiques**
- 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

Librairies / plugins externes – Scripts externes –
MOBILES – INTERNET des OBJETS – CLOUD – « BYOD »

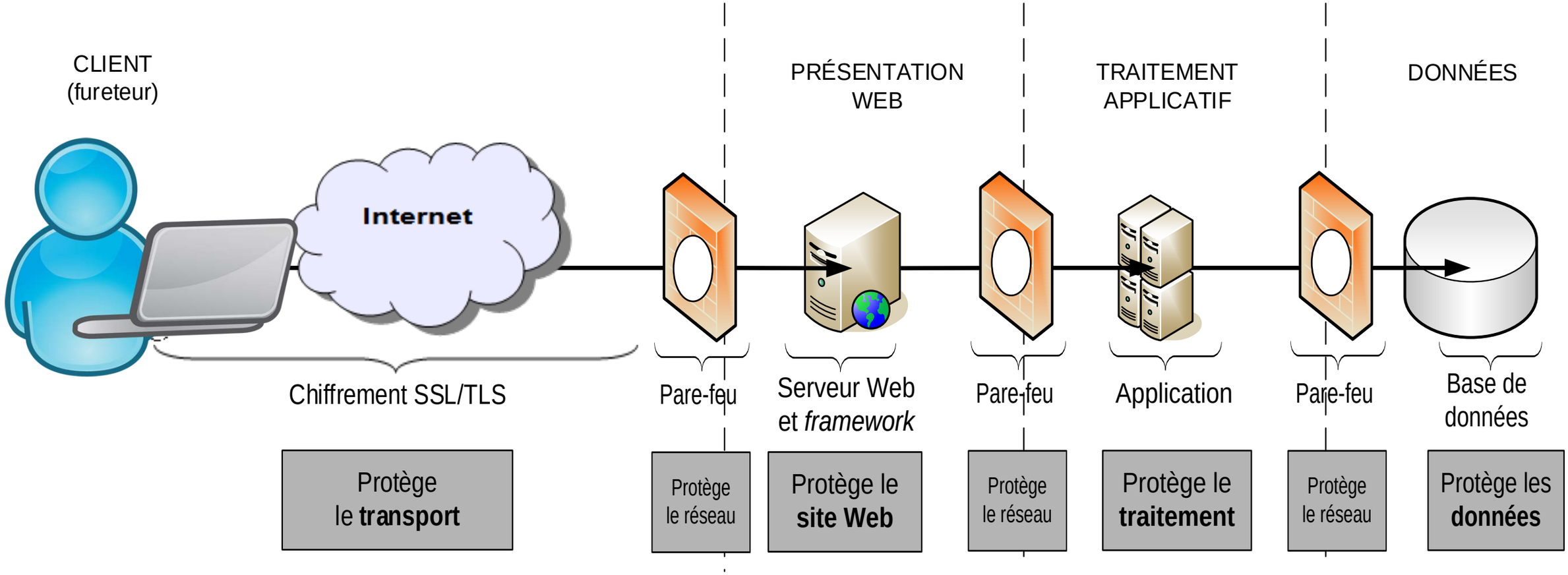
Aujourd'hui :

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

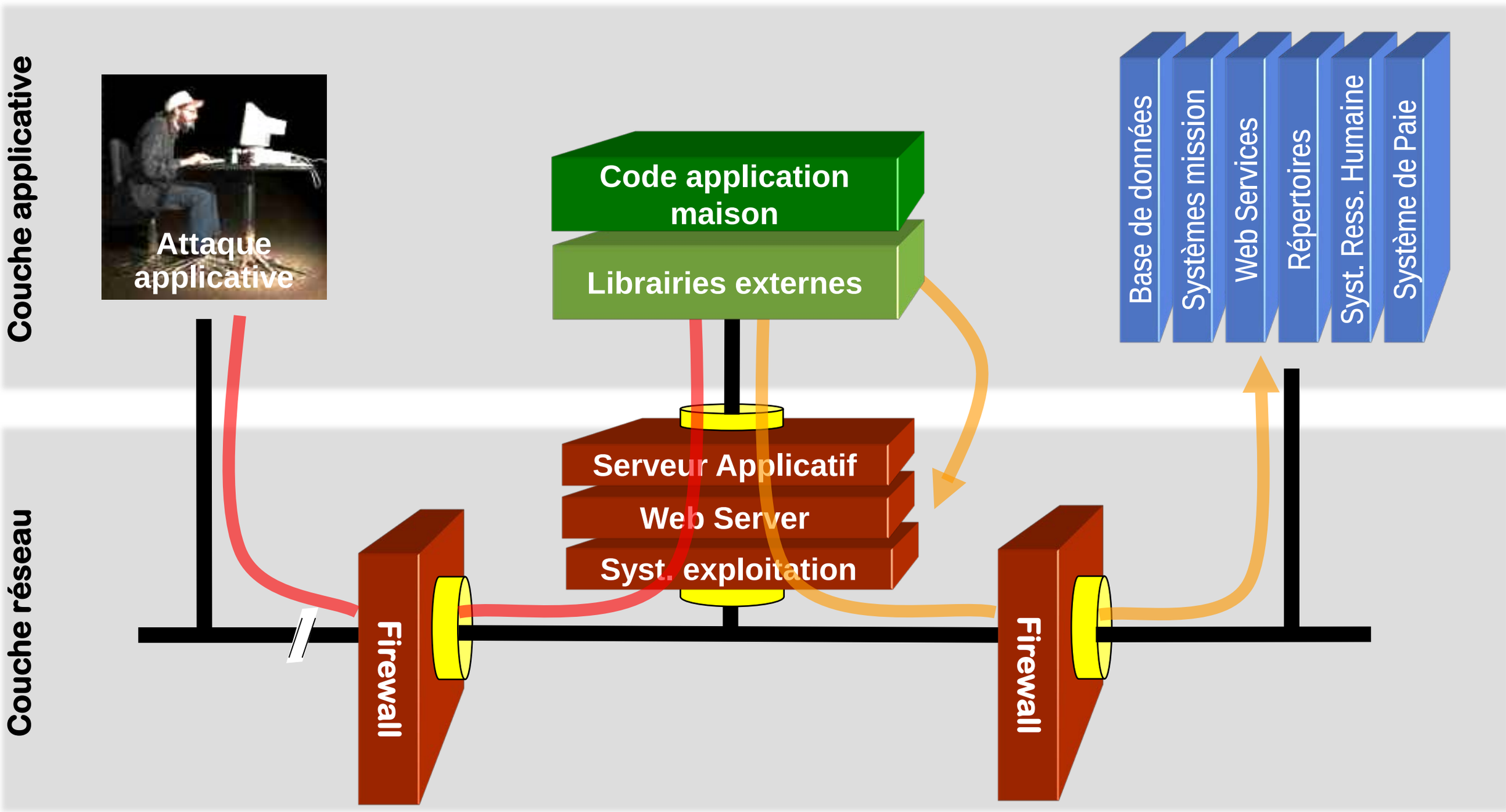
➢ **Où est le périmètre?**

La sécurité applicative est le nouveau périmètre

Architecture n-tiers typique



Le problème avec les applications Web…



Les 10 risques les plus critiques des applications Web

OWASP Top 10 2017
Fiches explicatives

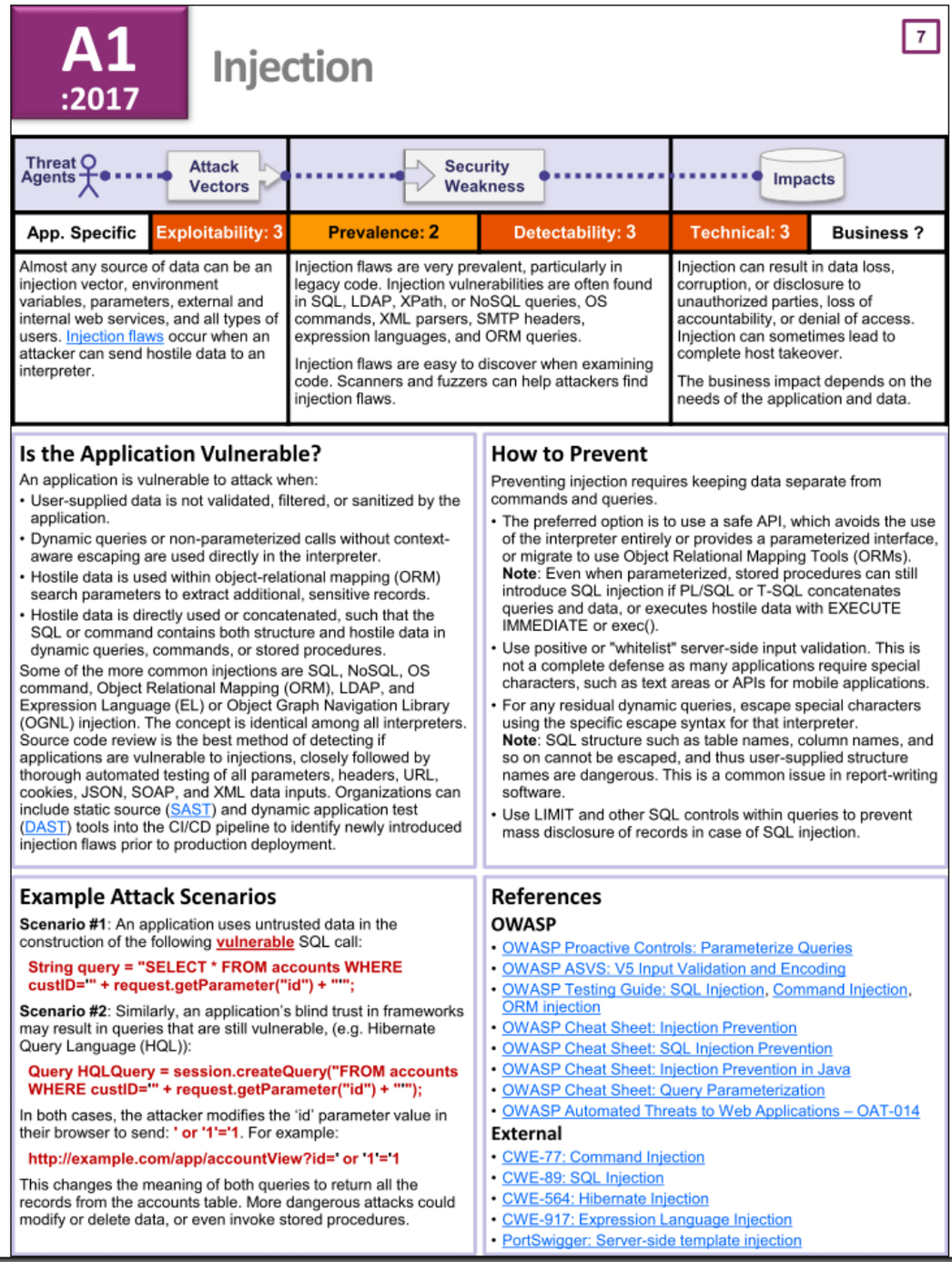
Menaces et vecteurs d'attaque
Vulnérabilités
Impacts

Niveaux de risques:

Threat Agents	Exploitability	Weakness Prevalence	Weakness Detectability	Technical Impacts	Business Impacts
Appli-cation 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

Threat Agents → Attack Vectors → Security Weakness → Impacts

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

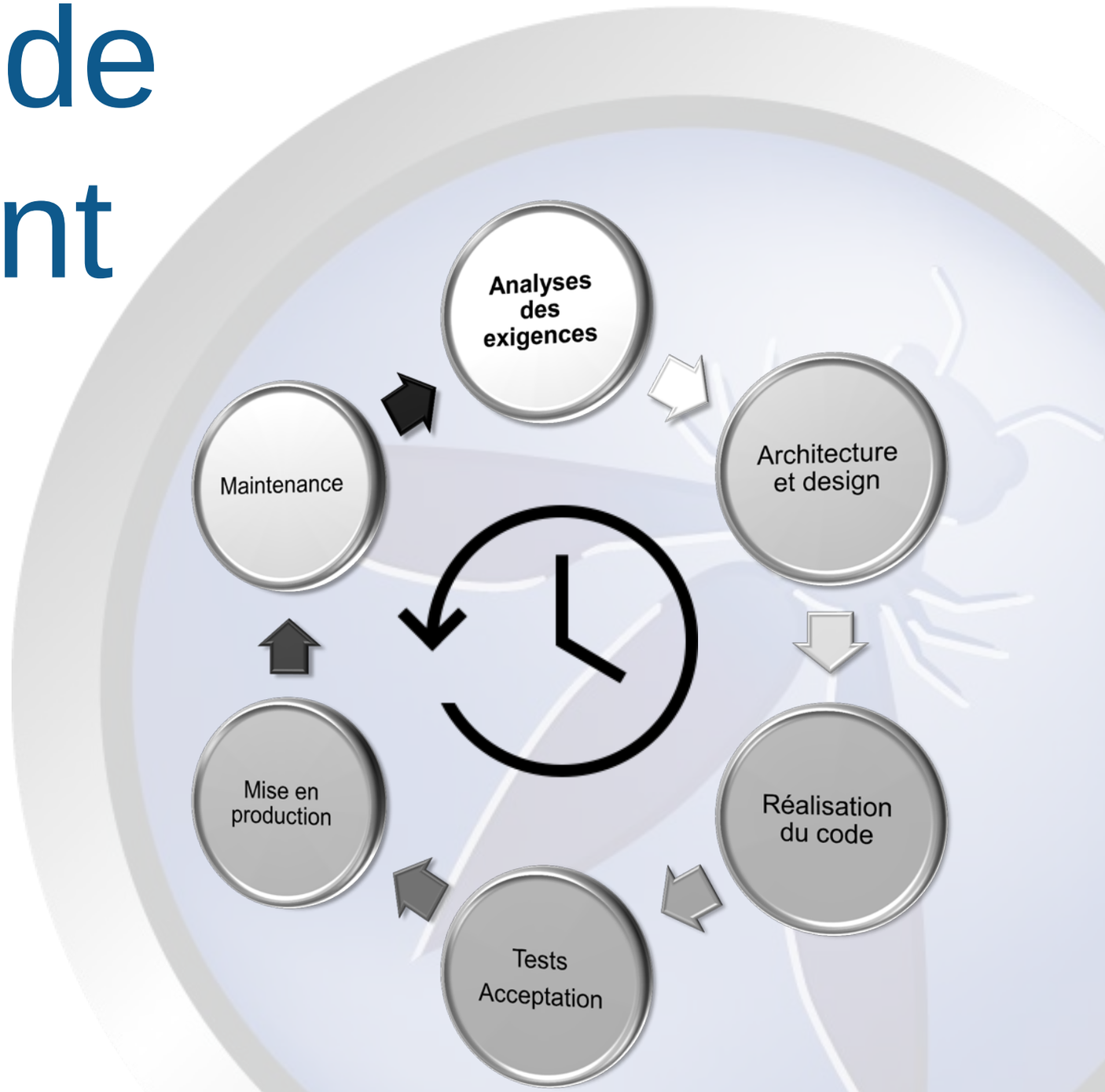
OWASP Top 10
Risques des applications Web

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.]

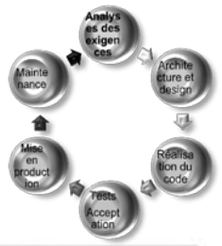
OWASP Top 10 2017
Risques des applications Web

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**

Intégrer la sécurité dans le cycle de développement

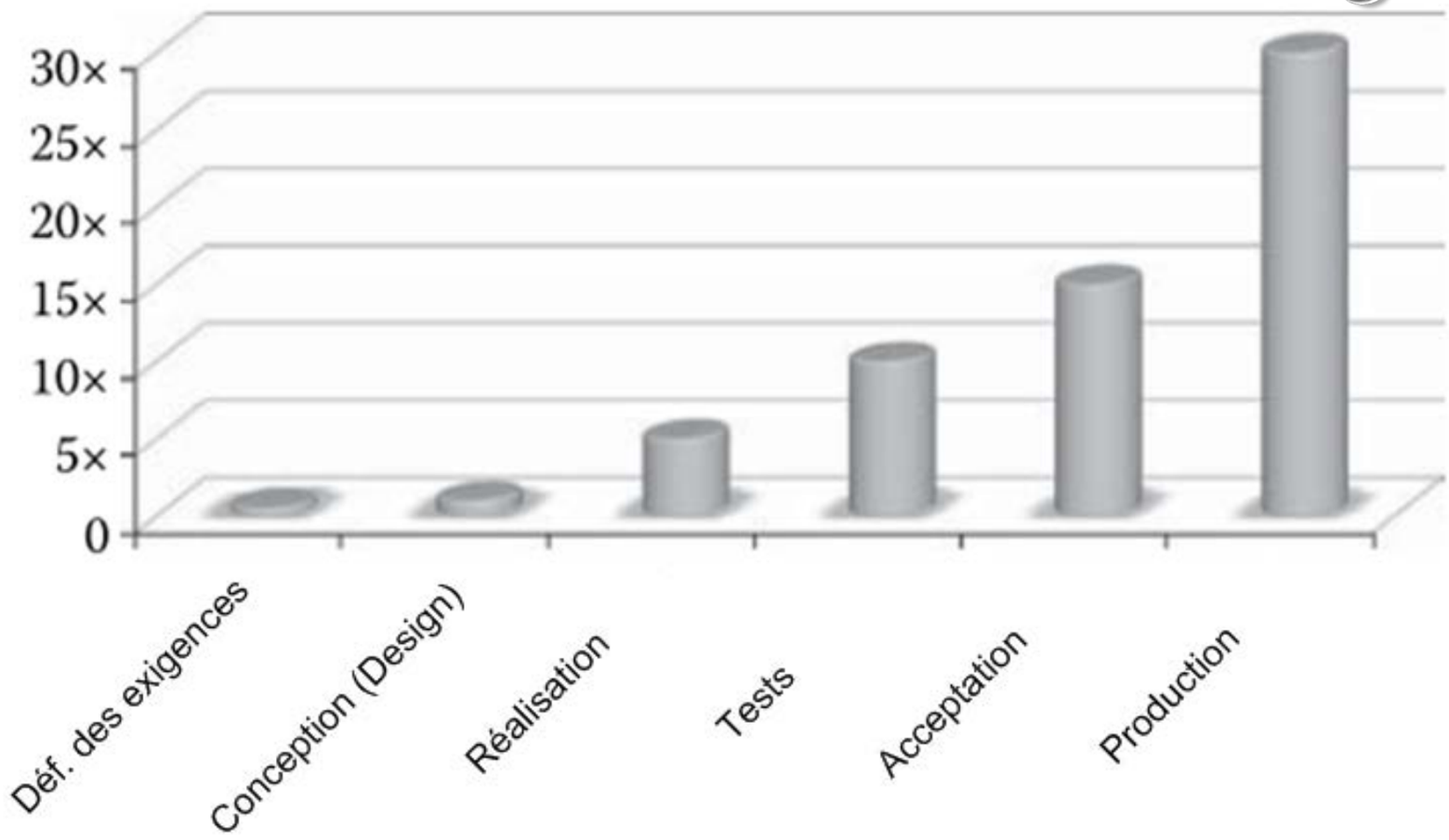


Sécurité dans le cycle de développement



Observation:

Les **coûts** reliés aux **corrections** des risques de sécurité **augmentent** de façon **exponentielle** quand les corrections sont découvertes tardivement dans le cycle de développement...



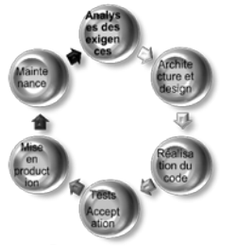
Évidence:

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

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

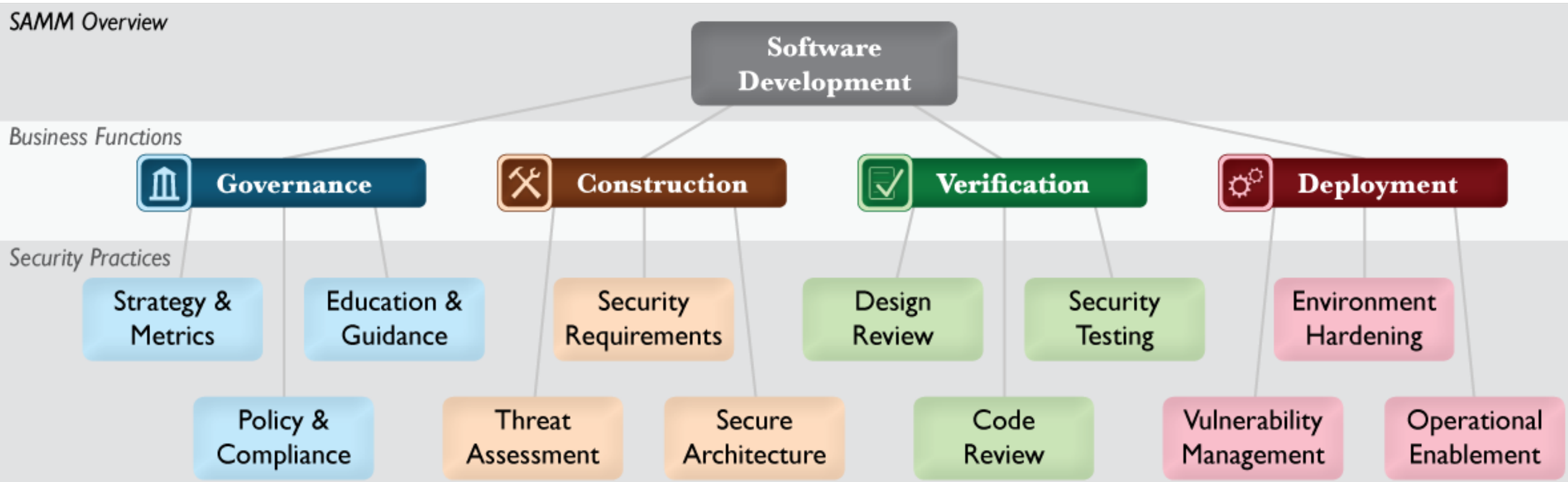
Phases d’un cycle simplifié…

SAMM (Software Assurance Maturity Model)

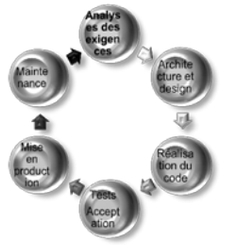


Outil pour vous aider **à définir l'approche idéale** de prise en charge de la sécurité applicative **en fonction du contexte particulier** et des **risques** de votre organisation

- Permet **l'intégration d'activités de sécurité** dans votre cycle de développement… peu importe la méthodologie que vous utilisez



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

Construction
Analyse de risque

Verification
Tests d'intrusion périodiques

Deployment
Guide sécurité opérationnelle

Deployment
Gestion des vulnérabilités

Deployment
Durcissement environnement

Verification
Tests d'intrusion préproduction

Construction
Exigences de sécurité

Construction
Modélisation des menaces

Construction
Architecture de sécurité

Verification
Revue de conception

Verification
Vérifications du code

Autres outils OWASP

OWASP Application Security Guide For CISOs

Guide réalisé pour les CISO (Chief Information Security Officer)

1. Critères **d'orientation** pour les **investissements** en sécurité applicative
   - *Conformité, gouvernance, audits, quantification des risques, bénéfices des mesures…*
2. Critères de **gestion des risques** de sécurité applicative
   - *Priorisations des vulnérabilités, menaces et contremesures*
3. **Programme** de sécurité applicative
4. **Métriques de gestion** de risques et d'investissement en sécurité applicative

OWASP Cheat Sheets (Aide-mémoire)

V - T - E

Cheat Sheets

[Collapse]

Construction
Architecture de sécurité

Developer / Builder

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 Prote

Verification
Revue de conception

Web Service Security · XSS (Cross Site Scripting) Prevention · XML External Entity (XXE) Prevention Cheat Sheet

Verification
Tests d'intrusion

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

Deployment
Durcissement environnement

Mobile IOS Developer · Mobile Jailbreaking

Verification
Vérifications du code

OpSec / Defender Virtual Patching

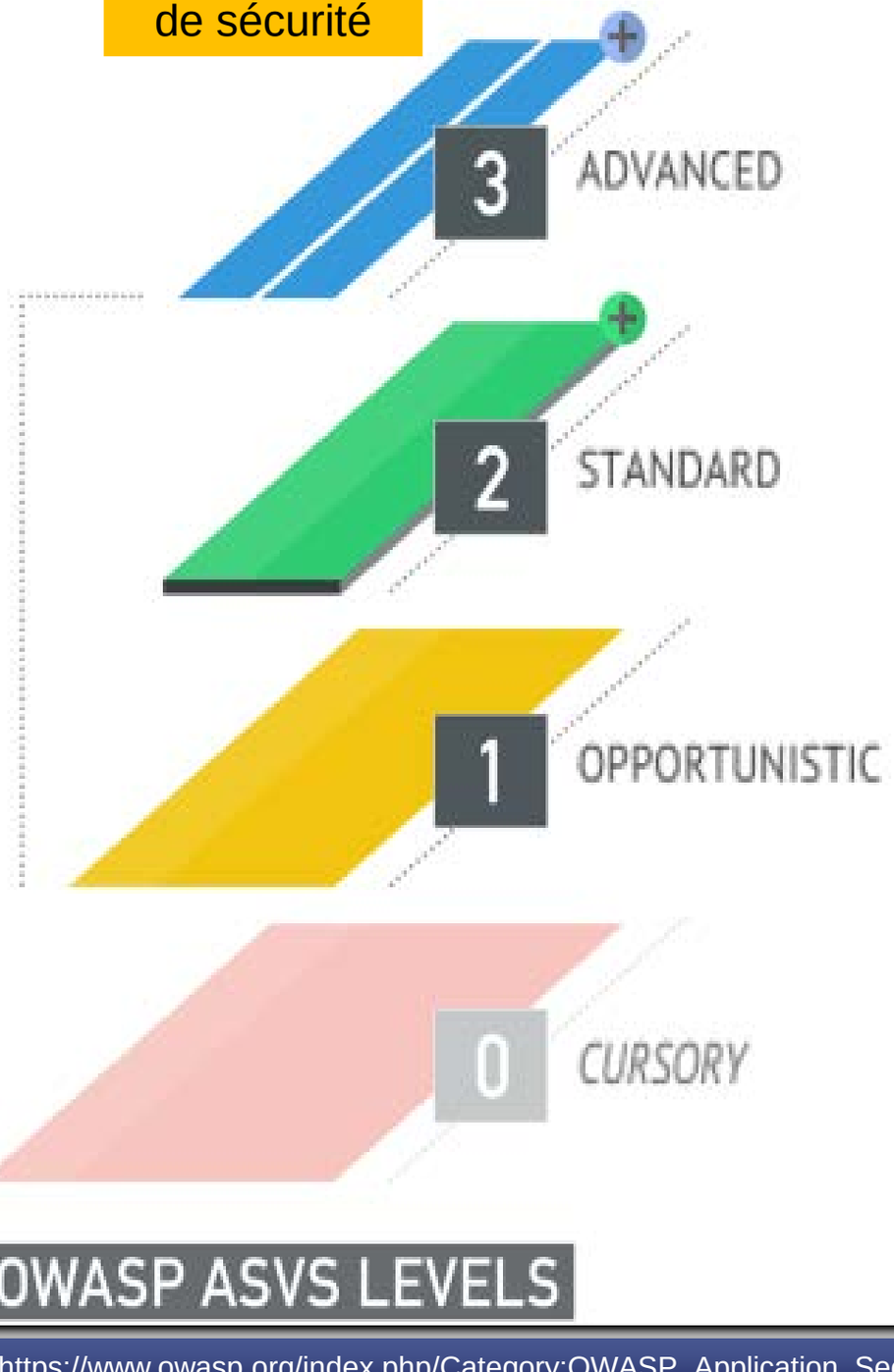
Draft and Beta 3rd Party Javascript Management · Access Control · Android Testing · Application Security Architecture · Business Logic Security · Injection Prevention Cheat Sheet · PHP Security · Secure Coding · Secure SDLC · Threat Modeling · Grails Secure Code Review · IOS Application Security Testing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

Construction
Modélisation des menaces

OWASP Application Security Verification Standard

Construction
Exigences de sécurité

Verification
Revue de conception



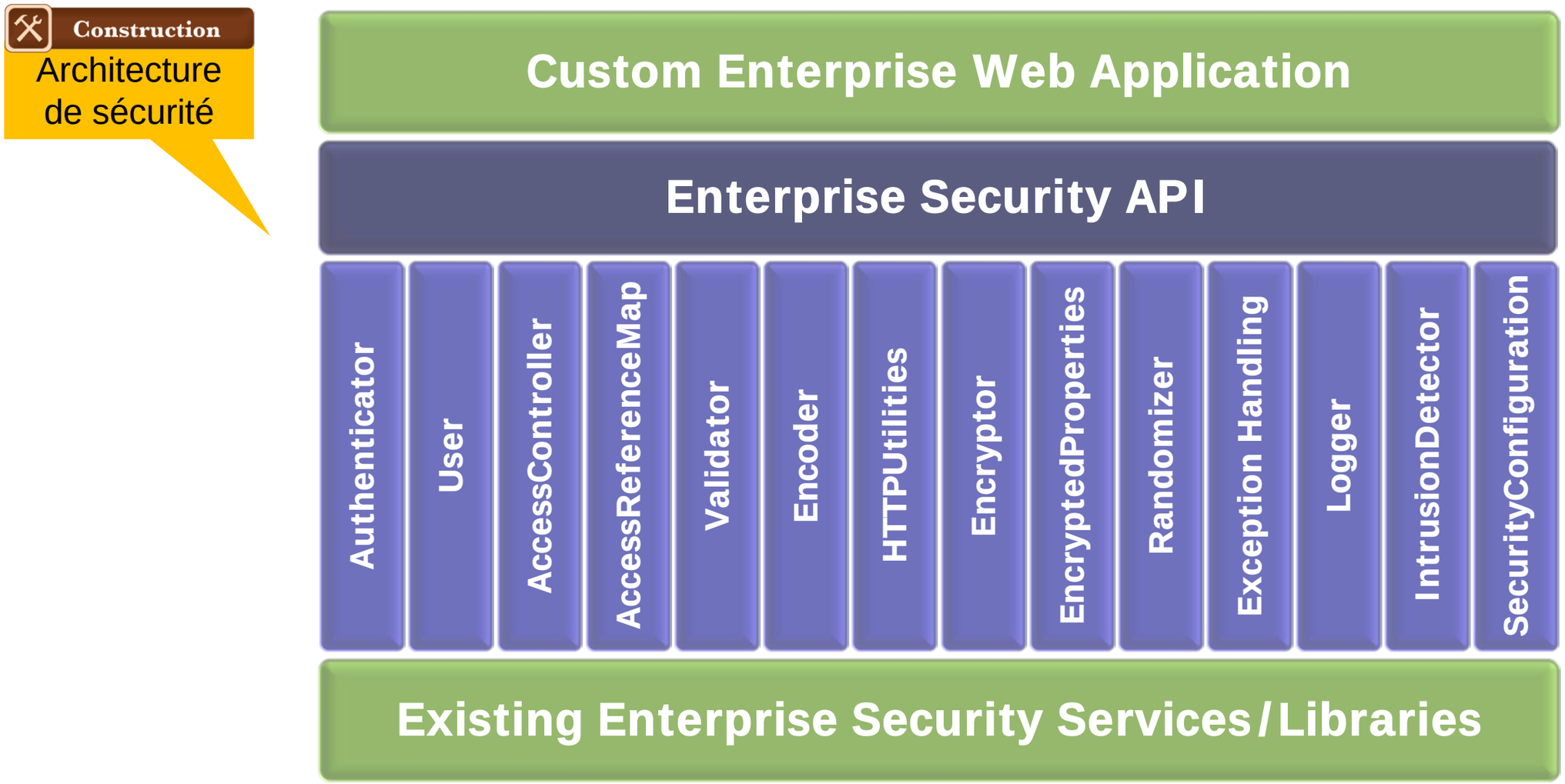
Requirements

#	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 ESAPI (Enterprise Security API)

« ***Framework*** » **JAVA*** offrant **plusieurs fonctions de sécurité usuelles**

**Malheureusement les autres langages manquent un peu d'amour ☹*



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

OWASP Dependency-Check

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**	

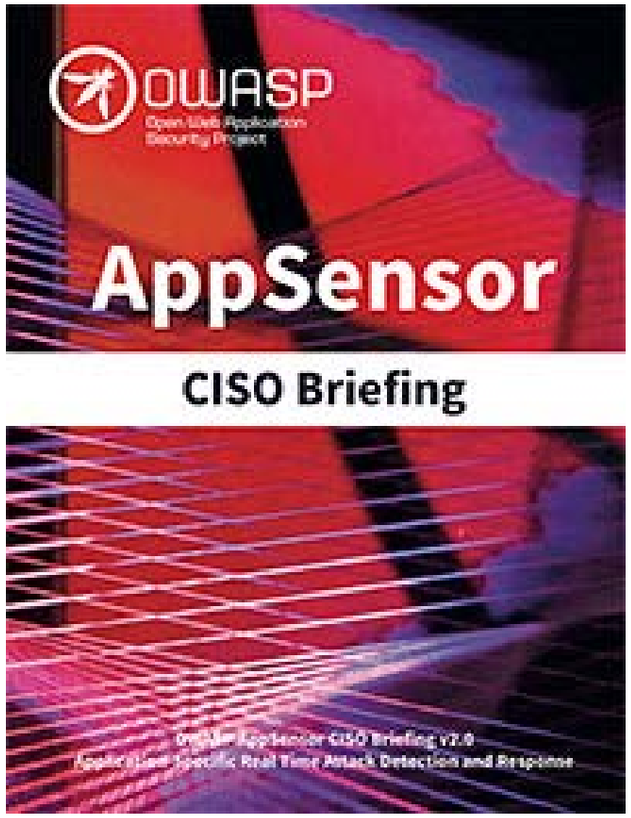
OWASP AppSensor

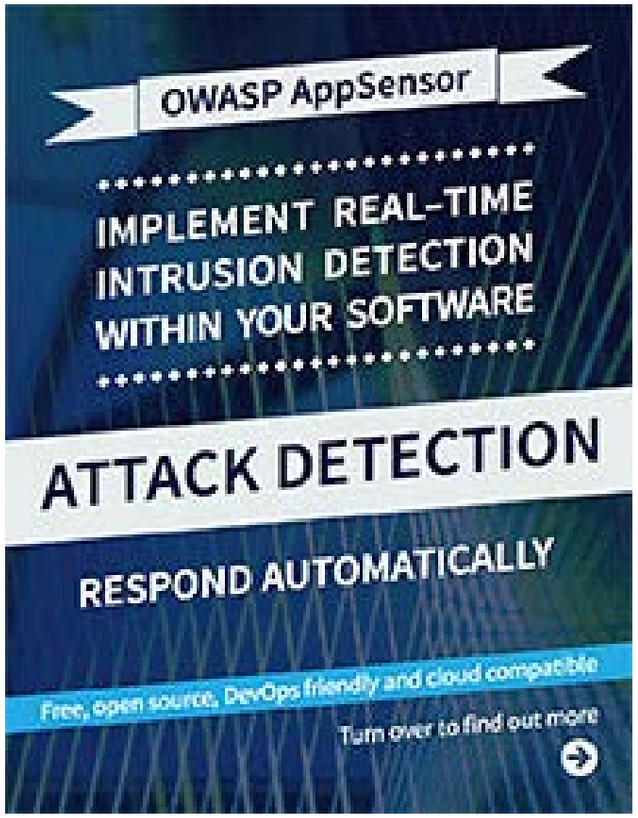
Construction
Architecture de sécurité

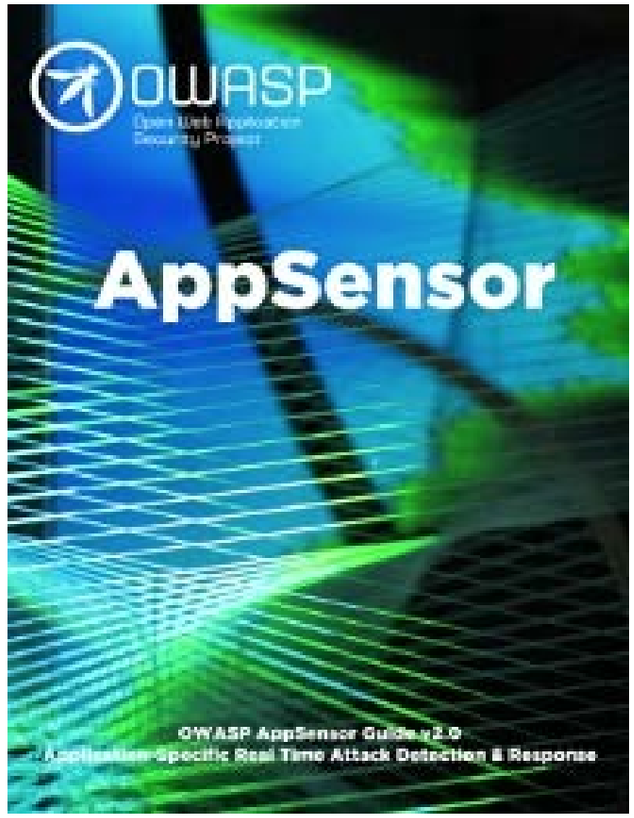
Technique avancée:

- **Méthodologie** et **cadre conceptuel** pour intégrer des **mécanismes de détection d'intrusion** et de **réponse automatique** dans les applications
- **Détection** en fonction de la logique de l'application
- **Réaction**: de l'alertage à la déconnexion (plus d'une douzaine d'actions décrites)

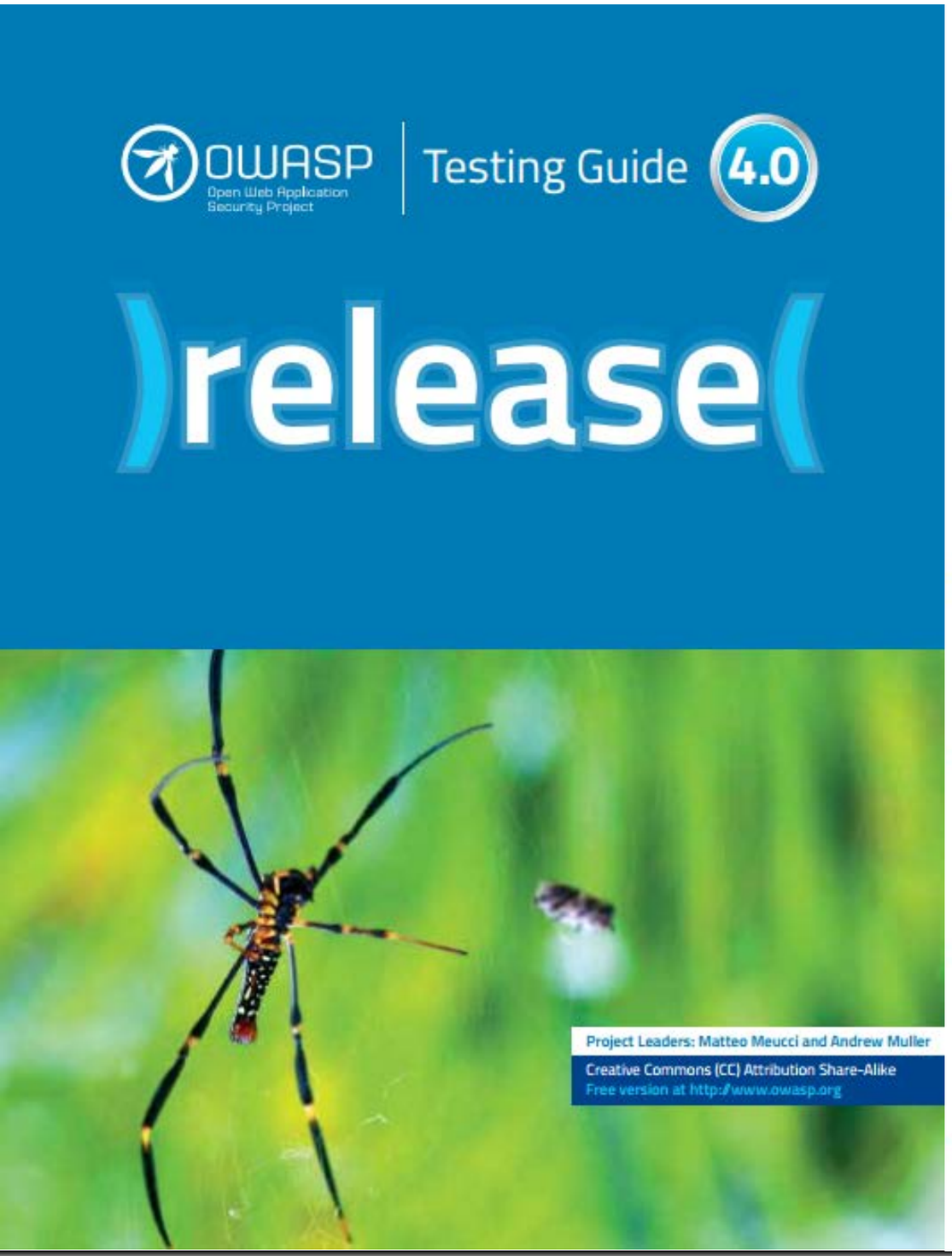
Deployment
Durcissement environnement

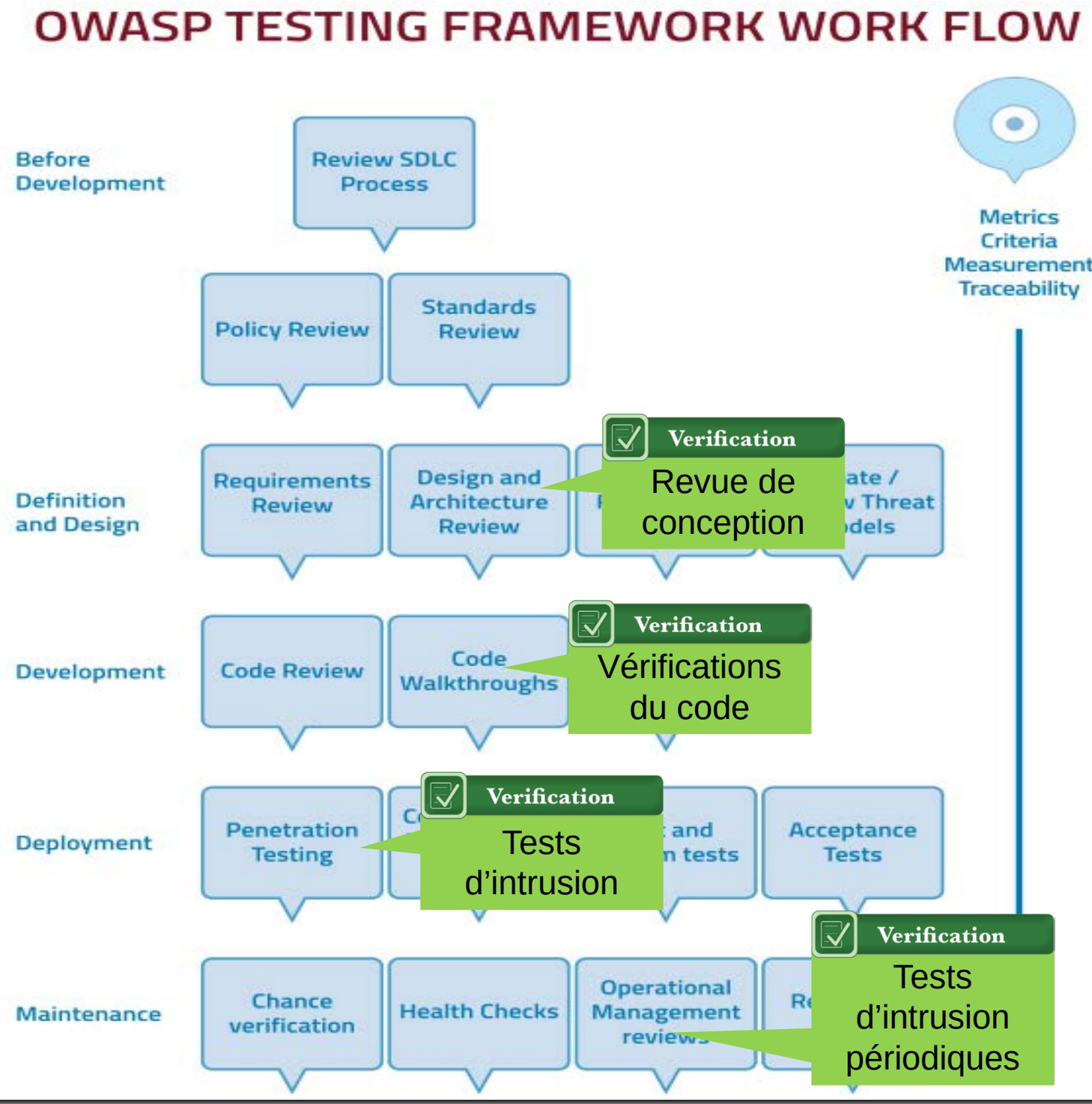






OWASP Testing Guide 4.0





OWASP Code Review Guide

Guide pour implanter une activité de revue de code:

- Explique la méthodologie et processus
- Couvre le 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

BEST 2015 SECURITY TOOLS

toolswatch
HACKERS ARSENAL

Verification
Vérifications du code

Verification
Tests d'intrusion

OWTF (OWASP Web Testing Framework)

Environnement pour **faciliter** et rendre plus **efficaces** les **tests d'intrusion applicatifs**

- **Orchestre** le lancement de **plusieurs outils** sur **plusieurs sites**
- **Collecte et organise** les résultats sous forme d'un **rapport interactif**

OWASP Training projects

Governance
Formation

2 plateformes **d'apprentissage** de sécurité applicative

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

Verification
Tests d'intrusion

- **OWASP WebGoat**
  - Applications délibérément non sécurisées
  - Plus de 30 leçons
  - Leçons et indices
  - « Challenges » réalistes

OWASP ModSecurity Core Rule Set

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

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

Deployment

Durcissement environnement

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.

modsecurity

Open Source Web Application Firewall

OWASP Ville de Québec

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

Présentations gratuites et orientées principalement pour les **gens de développement** :

- 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
- SSL et les problèmes de validation de certificats dans les applications mobiles
- Ce que vous devriez savoir sur le « Cloud computing »
- Développement d'une application sécurisée : étanchéité par le pipeline Web et le cadre d'application
- …

Partenariats pour faire connaitre la mission d'OWASP:

- **>>> Journée sur la sécurité applicative à l'Université Laval! <<<**
- Hackfest (2012-2017)
  - Événements « Capture The Flag » OWASP au HackFest (2014-2017)
  - Conférence sur le Top 10 des défenses Web (2012)
- Web à Québec 2016
- ASIQ 2016
- CQSI 2015
- ICCE 2014

Merci!

patrick.leclerc@owasp.org

OWASP Québec

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

Supportez les projets OWASP!

Bénéfices des membres OWASP:

- Rabais sur les conférences OWASP et autres
- Adresse de courriel: *toi*@**owasp.org**
- Droit de vote dans les élections
- Reconnaissance sur le site de l'OWASP
- 40% du montant peut être versé au chapitre

- Pour nous commanditer, suivre le lien Donate sur https://www.owasp.org/index.php/Quebec_City

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