



zensec
<by> zenika

Remettre la sécurité au centre des
projets



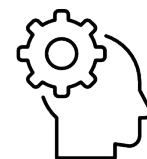
PÔLE SÉCURITÉ

NOS PROFESSIONNELS



PROFILS

Généraliste
Spécialiste / Hacker éthique
Formateur



COMPÉTENCES

Architecture
Développement
Transverses
Juridique
Contributeurs Opensource (Metasploit, OWASP, Multichain)



PÔLE SÉCURITÉ

NOS ACTIVITÉS



Audits

Audit de code
Audit d'architecture
Audit de configuration



IAM

Authentification
Gestion des accès



Formations

Initiation à la sécurité
Sécurité offensive
Formation à la demande



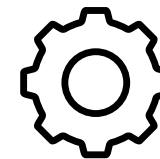
Tests

Tests d'intrusion
Social Engineering



Applicatif

Sécurité applicative
"Security & Privacy by Design"
"SecDevOps"



Veille

R&D
Opensource
Projet novateur





PÔLE SÉCURITÉ

NOS RÉFÉRENCES



Pipeline de déploiement sécurisé SecDevOps
Sécurité applicative (Angular, Java, Docker)
Audits applicatifs
Tests d'intrusions



POC et études de solutions SSO
Mise en place et déploiement d'OpenAM
Sécurité applicative (Java/JEE)
Audits de sécurité



Kerberisation plateforme BigData
Préconisations RGPD
Sécurité applicative





PÔLE SÉCURITÉ

NOS RÉFÉRENCES



Audit d'architecture
Audit intégration continue
Pentest application Android



Sécurité applicative (R, Java, MySQL)
Chiffrement des postes de travail
Chiffrement des bases de données
Automatisation/Sécurisation du SI
Coaching



Audit de code
Audit d'architecture
Tests d'intrusions
Démarche SecDevOps





PÔLE SÉCURITÉ

NOS RÉFÉRENCES

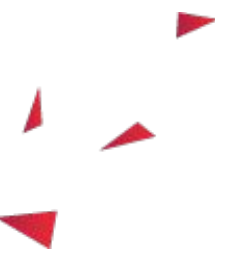


Audits
Tests d'intrusions



Sensibilisation & formation
Audits
SecDevOps
Coaching





Zenbox SecDevOps





LE PROGRAMME

LES BONS RÉFLEXES

- Analyse des dépendances
- Analyse statique du code
- Base de données
- Middlewares & Divers
- Tests d'intrusions





OWASP TOP 10

LES VULNÉRABILITÉS LES PLUS CONNUES

Priorité	Vulnérabilité
1	Injection
2	Broken Authentication
3	Sensitive Data Exposure
4	XML External Entities (XXE)
5	Broken Access Control
6	Security Misconfiguration
7	Cross-Site-Scripting (XSS)
8	Insecure Deserialization
9	Using Components with Known Vulnerabilities
10	Insufficient Logging&Monitoring





DÉPENDANCES

TRUST IS NOT ENOUGH

- Très souvent le premier vecteur de vulnérabilités (bibliothèques tierces)
- OWASP est à la rescousse :
 - Dependency Check (plusieurs plugins disponibles : Gradle, Maven, ligne de commandes...)
 - Dependency Track : suivi au niveau du SI
- Pour le front d'autres outils peuvent être utilisés :
 - nsp
 - RetireJS
- ...Et même Github s'y met !



```
One or more dependencies were identified with known vulnerabilities in test-dc:
commons-collections-3.2.1.jar <commons-collections:commons-collections:3.2.1, cp
e:/a:apache:commons_collections:3.2.1> : CVE-2015-6420
spring-boot-starter-mail-1.5.8.RELEASE.jar <cpe:/a:mail_project:mail:1.5.8, org.
springframework.boot:spring-boot-starter-mail:1.5.8.RELEASE> : CVE-2015-9097

See the dependency-check report for more details.
```

commons-collections-3.2.1.jar

Description: Types that extend and augment the Java Collections Framework.

License:

<http://www.apache.org/licenses/LICENSE-2.0.txt>

File Path: C:\Users\lenoir\.m2\repository\commons-collections\commons-collections\3.2.1\commons-collections-3.2.1.jar

MD5: 13bc641afd7fd95e09b260f69c1e4c91

SHA1: 761ea405b9b37ced573d2df0d1e3a4e0f9edc668

Referenced In Project/Scope: test-dc:compile

Evidence

Identifiers

- **maven:** [commons-collections:commons-collections:3.2.1](#) ✓ *Confidence:*HIGHEST
- **cpe:** [cpe:/a:apache:commons_collections:3.2.1](#) *Confidence:*HIGHEST

Published Vulnerabilities

[CVE-2015-6420](#)

Severity: High

CVSS Score: 7.5 (AV:N/AC:L/Au:N/C:P/I:P/A:P)

Serialized-object interfaces in certain Cisco Collaboration and Social Media; Endpoint Clients and Client Software; Network Application, Service, and Acceleration; Network and Content Security Devices; Network Management and Provisioning; Routing and Switching - Enterprise and Service Provider; Unified Computing; Voice and Unified Communications Devices; Video, Streaming, TelePresence, and Transcoding Devices; Wireless; and Cisco Hosted Services products allow remote attackers to execute arbitrary commands via a crafted serialized Java object, related to the Apache Commons Collections

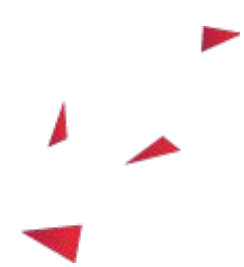












































DÉPENDANCES

TRUST IS NOT ENOUGH

- Et les licences alors?
 - Gros enjeu financier : un oubli ou une erreur peut coûter cher (revente de logiciel, modification de code source)
 - Pas vraiment d'outil gratuit permettant de gérer complètement cette problématique
 - Black Duck Software propose une plateforme permettant de gérer globalement et suivre ses dépendances Opensource (vulnérabilités, licences...)





Component ^		Source	Match Type	Usage	License	Security Risk			Operational Risk	
	 iText, a Free Java-PDF library 5.3.2	 1 Match	Exact Directory	Dynamically Linked	 AGPL-3.0				High	
	 JavaServer Pages (TM) TagLib Implementation 1.2	 1 Match	Exact Directory	Dynamically Linked	 Sun GPL With Classpath Exception v2.0 and 1 more...				High	
	 jep 2.24	 1 Match	Exact Directory	Dynamically Linked	 License Not Found					
	 Jersey 1.13	 1 Match	Exact Directory	Dynamically Linked	 Sun GPL With Classpath Exception v2.0 and 1 more...				High	
	 jersey-client 1.13	 1 Match	Exact Directory	Dynamically Linked	 Sun GPL With Classpath Exception v2.0 and 1 more...				High	
	 Transaction 1.1 API 1.0.0.Final	 1 Match	Exact Directory	Dynamically Linked	 CDDL-1.0 and 1 more...				High	





LE CODE SOURCE

ET OUI AUSSI....

- En utilisant correctement les frameworks, moins de vulnérabilités sont possibles
- Néanmoins, des analyseurs de code source existent :
 - Sonarqube : TOP 10 OWASP, CVE
- Pour le front, possibilité d'utiliser des outils d'analyse syntaxique :
 - Respect des standards de développement
 - Applicable à la plupart des langages
- N'oubliez pas de protéger vos APIs :
 - Authentification
 - /!\ CORS, CSRF...





BASE DE DONNÉES

EVITEZ LES FUITES

- Activer l'authentification...
- Chiffrement physique :
 - Évite un vol de données en les copiant physiquement
- Chiffrement logique :
 - Évite de dévoiler des informations sensibles
 - Parfois obligatoire selon la législation





LES MIDDLEWARES

SECURITY IS EVERYWHERE

- Maîtriser les flux :
 - Configuration des firewalls, Fail2Ban
- Protéger votre infrastructure :
 - IDS/IPS... (entre autres)
- Sur votre serveur frontal (HTTPD, Nginx...) :
 - Mettre en place HTTPS
 - Spécifier les bons headers





LES HEADERS

PROTÉGER ET MITIGER

Nom	Rôle
HSTS	Forcer le passage en HTTPS
Content-Security-Policy	Maîtriser le chargement de sources (et mitiger les XSS) en limitant à des domaines / éviter également les scripts inline
X-Content-Security-Policy	Complément à CSP sur des vieux navigateurs
X-XSS-Protection	Complément à CSP sur des vieux navigateurs
X-Frame-Options	Eviter le chargement par iframe
X-Content-Type-Options	Contrôler les types MIME
HPKP	Stocker les clés de confiance pour détecter les éventuelles compromissions de certificats





MAIS AUSSI...

NE LES OUBLIEZ PAS

- Gestion du triplet “AAA” :
 - Authentication : qui es-tu?
 - Authorization : que peux-tu faire?
 - Accounting : qu’as-tu fait? (traçabilité)
- Gestion des secrets :
 - Hashicorp Vault
- Gestion des logs :
 - Utiliser les modules existants sur les middlewares
 - Côté back sur API spécifique, l’AOP peut être utilisée
 - Attention à ce que vous mettez dedans (législation)





TESTS D'INTRUSIONS

ET OUI ON PEUT EN AUTOMATISER UNE PARTIE...

- OWASP (encore...) à la rescousse : Zed Attack Proxy (ZAP)
- Plusieurs modes pour ZAP :
 - Passif : aucune action offensive (il peut donc être utilisé sur un site de production)
 - Actif : l'outil est plus offensif et tente des attaques sur l'application
- Possibilité de lancer l'outil via un conteneur Docker et de lui passer facilement les configurations souhaitées
- Possibilité d'utiliser des outils déjà mis en place





EXEMPLE DE PIPELINE DE DÉPLOIEMENT

AVEC UN PEU DE SÉCURITÉ C'EST TOUJOURS MIEUX

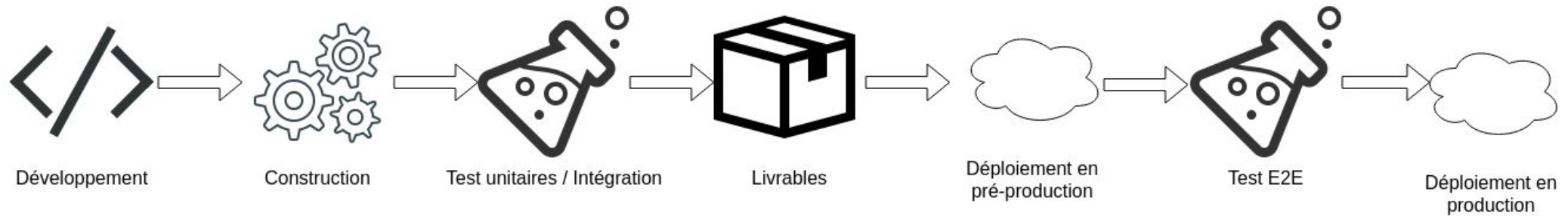
- Un pipeline classique permettant d'enchaîner les étapes suivantes :
 - Tests unitaires
 - Tests d'intégration
 - Construction des livrables
 - Déploiement sur environnement de test (fonctionnel)
 - Tests E2E
 - Archivage sur Artifactory/Nexus



EXEMPLE DE PIPELINE DE DÉPLOIEMENT

AVEC UN PEU DE SÉCURITÉ C'EST TOUJOURS MIEUX

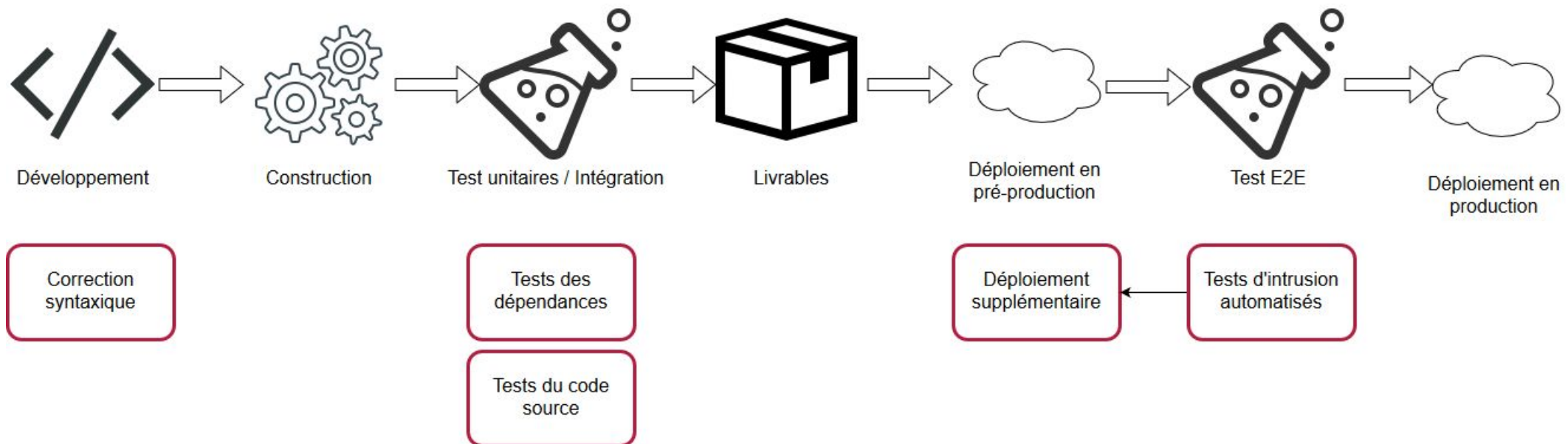
- Un pipeline classique permettant d'enchaîner les étapes suivantes :



EXEMPLE DE PIPELINE DE DÉPLOIEMENT

AVEC UN PEU DE SÉCURITÉ C'EST TOUJOURS MIEUX

- Un pipeline incluant des étapes de sécurisation :





PÔLE SÉCURITÉ

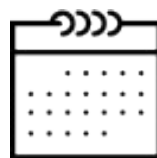
NOS MÉDIAS



PRESSE

MISC : article SIEM avec Elastic Stack

Programmez! : articles (Metasploit, Darknets)



EVÉNEMENTS

La Nuit du Hack (24/06/2017)

Identity Live (21/11/2017)

FIC (2018)

SSTIC (2018)



ARTICLES DE BLOG

REX LNDH

Saga sur Metasploit

Darknets

IAM

Blockchain





zenika

<animés par la passion>

MERCI

www.zenika.com

blog.zenika.com



pole-securite@zenika.com

01 45 26 19 15