

# Retour d'expérience d'intégration des principes de sécurité dans un cycle de développement logiciel

## Integrating Security into Software Development Lifecycle : an Experience Return

Laurent Butti<sup>1</sup>

<sup>1</sup>Orange

### Résumé

Dans un contexte où la livraison de nouvelles fonctionnalités des produits ou services doit se faire rapidement pour lutter commercialement, les principes de sécurité ne doivent pas être laissés sur le bord de la route. Au regard des aspects sécurité, les nouveaux modèles d'organisation (agilité, DevOps) sont à la fois une opportunité et un risque par rapport au traditionnel cycle en V. Cela nécessite une intégration des principes de sécurité de manière systématique et la plus automatisée possible dans le cycle de développement logiciel. Ceci requiert un engagement managérial fort qui pourra engendrer à terme une maturité organisationnelle, humaine et technique. Nous présenterons dans cet article un retour d'expérience de 10 ans d'intégration et d'automatisation des principes de sécurité dans un cycle de développement logiciel de services Web grand public d'un opérateur français.

In a period where new features have to be delivered fast to the customers, security principles have to be properly implemented. Regarding security, new organisation models (agility, DevOps) are both a chance and a risk in regards to traditional ways of project management. It requires security aspects to be implemented in a systematic and automatic ways in software development lifecycle. Strong management support is a prerequisite as it should lead to improve organisational, human and technical maturity. In this article, we will discuss a 10+ years experience return about integrating security principles in a software development lifecycle of a French telco public-facing web services.

### Keywords

retour d'expérience, automatisation, cycle de vie logiciel, sécurité Web, DevSecOps

## 1. À propos de l'auteur

Laurent est expert sécurité à Orange. Il a une expérience d'une dizaine d'années en environnement de recherche en développement sécurité où des domaines tels que la sécurité des réseaux sans-fil Wi-Fi ou la recherche de vulnérabilités par fuzzing ont été explorés, ce qui a valu brevets et publications dans de nombreuses conférences internationales (BlackHat USA, BlackHat Europe, FIRST, ToorCon, SSTIC...). Il est aujourd'hui impliqué depuis une dizaine d'années dans la sécurité des portails Grand Public d'Orange, aussi bien sur le cycle de développement logiciel


---

C&ESAR 2021, 16-17 novembre 2021, Rennes, France

✉ laurent.butti@orange.com (L. Butti)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

que sur la construction de leur hébergement, dans un environnement de production à très fortes contraintes opérationnelles (Orange numéro 4 dans le Top Alexa France) [1].

## 2. À propos

Dans cet article nous présenterons dans un premier temps le contexte général, puis nous détaillerons les différentes phases d'un modèle d'intégration de sécurité dans un cycle de développement logiciel : de la phase de conception au déploiement, en passant par les phases d'implémentation et de vérification de conformité.

Nous détaillerons notre retour d'expérience sur les parties organisationnelles, la sélection de l'outillage et la mise en oeuvre de son automatisation ; et nous nous efforcerons de présenter les facteurs clés de réussite mais aussi les pièges à éviter.

Nous n'aborderons pas les principes de sécurisation des infrastructures d'hébergement, ni des processus tiers manuels élevant naturellement le niveau de sécurité des développements, tels que les analyses de risques, les analyses « données personnelles », les audits de sécurité manuels (code source ou intrusifs) ou encore la mise en Bug Bounty (public ou privé) que nous avons d'ailleurs effectuée sur la quasi-totalité de nos services exposés en public [2].

Cet article présente les vues et convictions personnelles de l'auteur. Par ailleurs, il est de bon ton de rappeler que les choix réalisés sont toujours liés à un contexte et ne sauraient être systématiquement répliqués aveuglément.

## 3. Contexte

Notre organisation développe et héberge la grande majorité des services Web grand public Orange, dont les principaux sont la page d'accueil Orange, l'identité Grand Public, les boutiques en ligne, la relation client, l'assistance... Dans un contexte de digitalisation massive des parcours clients, ces services sont critiques pour développer des parts de marché en gagnant la confiance des clients du Groupe Orange. Il représente cependant une attraction toute particulière pour les actes malveillants de par l'exposition publique, les fraudes potentielles et les données personnelles qui y sont accessibles.

L'intégration des principes de sécurité dans un cycle de développement ne se fait pas naturellement, cela requiert une approche rigoureuse et systématique sur l'ensemble des phases de la construction d'un service. Mais aussi et surtout de réussir l'enjeu du développement de la culture sécurité dans ce fameux modèle produit où équipes métier, de développement et d'intégration ne font plus qu'une fonctionnellement.

Mais, à la genèse, il faut un fort engagement managérial et une vision sécurité qui l'accompagne, afin de faire progresser la maturité sécurité de l'organisation et des collaborateurs/collectifs qui la font vivre.

## 4. Introduction

Intégrer les aspects sécurité dans un cycle de développement logiciel, certes, mais dans quel(s) but(s) ?

La réponse est évidente pour des personnes férues de technique et sensibilisées aux aspects sécurité... Élever le niveau de sécurité des développements entraînera forcément moins de failles de sécurité et donc aura des répercussions très bénéfiques sur le produit final et la confiance des clients : réduction des correctifs en production, réduction des risques de compromission de données privées et/ou personnelles, réduction des risques de perte d'image de marque...

Les gains financiers pour l'organisation sont malheureusement difficiles à quantifier puisqu'ils reposent sur une estimation de la réduction conjointe de l'occurrence et des impacts de l'exploitation de failles de sécurité (l'habituel calcul de criticité : risque = probabilité d'occurrence x gravité). Par ailleurs, mesurer l'impact d'une compromission de données personnelles d'utilisateurs d'un service est souvent complexe et approximative (techniques de financierisation des risques) puisque cela peut autant se révéler dévastateur comme cela a été le cas pour DigiNotar qui a déposé le bilan (suite à l'émission de « vrai faux » certificats par son autorité de certification) [3] ou acceptable financièrement par des sociétés pesant quelques dizaines de milliards de capitalisation comme cela a été le cas pour Sony en 2012 (suite au déni de service sur le « PlayStation Network » puis à la compromission de données personnelles grâce à l'exploitation d'une injection SQL sur leur site Web) [4].

Habituellement, la survie d'une entreprise passe par un maintien de compétitivité par rapport aux concurrents, mais pas nécessairement dans tous les cas de figure... En effet, les contextes sont très variés : que dire du niveau de sécurité à atteindre dans le milieu médical où la santé des patients est potentiellement en jeu ? Dans l'industrie automobile avec l'arrivée massive de technologies interconnectées ? Dans le pilotage de systèmes industriels ? Dans le milieu défense où par nature les menaces étatiques sont fortes ? Selon les contextes, les failles de sécurité exploitées sont potentiellement dévastatrices, et parfois l'État y met son grain de sel avec des prescriptions telles que celles pour les Opérateurs d'Importance Vitale (OIV) [5].

Dans un contexte d'entreprise « classique », les motivations liées à la sécurité peuvent être multiples :

- la disponibilité du service fourni pour garantir le bon fonctionnement des activités ;
- amélioration de la qualité (les aspects sécurité devant s'inscrire naturellement dans les processus de qualité de développement) ;
- se conformer à une réglementation étatique ou de conglomérat d'états (e.g. la « General Data Protection Regulation ») [6] ;
- se conformer à des exigences contractuelles dans une relation client / fournisseur ;
- se conformer aux exigences d'un standard de sécurité (e.g. la norme de sécurité de l'industrie des paiements, « PCI-DSS : Payment Card Industry Data Security Standard ») [7] ;
- développer la confiance des clients et l'image de la marque...

Autant de raisons toutes aussi valables les unes que les autres qui élèvent le niveau de sécurité des produits et services de l'entreprise concernée.

Et dans le cadre de cet article, nous nous focaliserons dans un contexte de cycle de développement logiciel.

## 5. Modèles d'intégration des aspects sécurité dans un cycle de développement logiciel

Les modèles les plus connus sont le « Security Development Life-Cycle » de Microsoft [8], le « Software Assurance Maturity Model » [9] de l'OWASP et le « Building Security In Maturity Model » (BSIMM) [10].

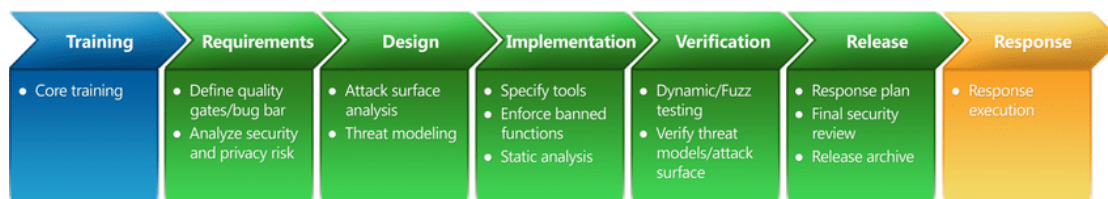


FIGURE 1: Microsoft Software Development Life-Cycle

Le modèle de Microsoft est un des premiers à avoir formalisé les étapes sécurité dans la conception d'un produit, il manque toutefois la vue globale d'une gouvernance d'organisation qui serait applicable à un ensemble de lignes de produits.







FIGURE 2: OWASP Software Assurance Maturity Model

Le modèle de l'OWASP a été démarré par une initiative d'un membre de la communauté de l'OWASP, et a été repris par des contributions communautaires.

Le modèle BSIMM s'inspire originellement de celui de l'OWASP en le complétant grâce à des contributions d'entreprises spécialisées dans le développement logiciel.

Ces différents modèles nous éclairent sur l'étendue du périmètre à couvrir, organisationnelles

DOMAINS			
 GOVERNANCE	 INTELLIGENCE	 SSDL TOUCHPOINTS	 DEPLOYMENT
Practices that help organize, manage, and measure a software security initiative. Staff development is also a central governance practice.	Practices that result in collections of corporate knowledge used in carrying out software security activities throughout the organization. Collections include both proactive security guidance and organizational threat modeling.	Practices associated with analysis and assurance of particular software development artifacts and processes. All software security methodologies include these practices.	Practices that interface with traditional network security and software maintenance organizations. Software configuration, maintenance, and other environment issues have direct impact on software security.
PRACTICES			
GOVERNANCE	INTELLIGENCE	SSDL TOUCHPOINTS	DEPLOYMENT
1. Strategy & Metrics (SM) 2. Compliance & Policy (CP) 3. Training (T)	4. Attack Models (AM) 5. Security Features & Design (SFD) 6. Standards & Requirements (SR)	7. Architecture Analysis (AA) 8. Code Review (CR) 9. Security Testing (ST)	10. Penetration Testing (PT) 11. Software Environment (SE) 12. Configuration Management & Vulnerability Management (CMVM)

**FIGURE 3:** BSIMM 12

et techniques.

La partie automatisation est un moyen de faire pour passer à l'échelle, c'est une conséquence vertueuse de l'amélioration continue.

Dans cet article, nous nous focaliserons sur le pilier cycle de développement logiciel.

Nous invitons le lecteur à se référer à ce dernier modèle qui est l'état de l'art sur le sujet.

Un des avantages indéniables de l'initiative BSIMM est de réaliser régulièrement une évaluation du modèle BSIMM sur un panel d'entreprises volontaires. Il en ressort alors une vue concrète de la maturité des différentes entreprises selon les sujets et selon leurs domaines d'activité.

Il est aussi très intéressant de constater des chiffres sur la force au travail dédiée aux aspects sécurité (e.g. correspondants sécurité, ou « security champions » dans les équipes produits), ce qui peut servir d'indicateur en terme de ressources à engager sur le sujet.

## 6. Retour d'expérience sur les principes organisationnels

Nous présenterons ici les changements majeurs dans le monde de l'entreprise qu'il est nécessaire de comprendre afin d'accompagner l'intégration des principes de sécurité.

Enfin nous détaillerons le modèle organisationnel mis en place afin de faciliter la responsabilisation des équipes produits tout en s'assurant que les principes de sécurité soient tout de même pris en compte.

## 6.1. L'engagement managérial et les ressources humaines

Le cercle vertueux de la sécurité ne peut s'activer que par un fort engagement managérial à haut niveau. À la manière du virage qui a été engagé par Bill Gates en 2002 avec son célèbre mail « Trustworthy Computing » : « So now, when we face a choice between adding features and resolving security issues, we need to choose security. Our products should emphasize security right out of the box, and we must constantly refine and improve that security as threats evolve. » (source : Bill Gates).

C'est en effet le nerf de la guerre puisqu'il en découle ensuite le sens et une stratégie déclinée au sein de l'entreprise.

Bien entendu, selon les priorités de l'entreprise sur les différentes lignes de produits, les moyens financiers peuvent être bien différents. En conséquence, les moyens sécurité aussi, et il faudra choisir de manière la plus pragmatique possible ce qui sera envisageable d'atteindre avec les moyens donnés.

Ne nous leurrions pas, l'objectif est de faire mieux avec moins. Et dans ce cadre-là, l'automatisation est systématiquement une carte à jouer puisqu'elle est censée réduire les efforts à terme sur des sujets atteignables (ne nécessitant pas d'analyse humaine pointue) à condition toutefois de mettre les moyens pour construire ces processus/outils d'automatisation, ce qui est clairement un défi.

## 6.2. Le changement de culture : agilité et DevOps

Le traditionnel cycle en V est de moins en moins prévalent dans les organisations modernes et des notions d'agilité sont apparues au fil du temps.

L'agilité est « est une approche itérative de la gestion de projets et du développement logiciel qui permet aux équipes de créer de la valeur plus rapidement et plus facilement. Plutôt que de tout miser sur un lancement majeur, une équipe Agile livre son travail par petits incréments exploitables. Les exigences, les plans et les résultats sont évalués en continu. Les équipes disposent donc d'un mécanisme naturel pour répondre rapidement au changement » [11].

Tout ceci bouscule les modes de fonctionnement « traditionnels » des entités sécurité qui étaient habituées à intervenir de manière quasi-systématique en amont et aval des projets en cycle V, une adaptation a donc été nécessaire.

De la même manière, les cycles de développement/déploiement se sont raccourcis, et les mises en lignes de nouvelles fonctionnalités et de maintenance corrective sont maintenant journalières, comment s'assurer que cela n'introduise pas de problématiques de sécurité ? C'est un défi majeur pour les équipes produits et les équipes sécurité qui les accompagnent.

Là encore, l'automatisation apporte une réduction des risques, mais ne peut pas être aussi performant qu'un audit manuel avant chaque mise en ligne, il y a donc un équilibre à avoir entre efforts de vérifications manuels, efforts de vérifications automatiques et les risques résiduels.

En contrepartie, cette agilité est aussi extrêmement bénéfique sur le traitement des vulnérabilités, puisque les cycles se sont extrêmement raccourcis pour traiter les vulnérabilités. Preuve il en est, la quasi totalité de notre parc applicatif Grand Public exposé sur Internet est en Bug Bounty [2], ce qui témoigne d'une maturité certaine des équipes produits dans le traitement des vulnérabilités qui sont sous des délais cadrés en fonction du niveau de criticité.

Nous conseillons au lecteur de se référer au guide de l'ANSSI [12] et au pages de Atlassian [13] qui apportent un éclairage supplémentaire à l'état de l'art sur le sujet [14] [15].

### **6.3. Le changement de culture : la mise en responsabilisation des responsables produits et équipes produits**

Dans un monde Agile, les « Product/Service Owners » sont pleinement responsables de la feuille de route de leur produit ou service. En conséquence, ils sont aussi garants des aspects sécurité sur leur produit ou service.

La mise en responsabilité est bien entendu vertueuse et une belle cible à atteindre, ce qui résonne à ce qu'affirme Werner Vogels (Amazon CTO, 2018) « security is everyone's job now, not just the security team's. » repris de concert par Dino Dai Zovi dans sa présentation d'introduction à BlackHat USA 2019 « Every Security Team is a Software Team Now » [16], mais elle ne peut réussir que si des prérequis sont en place.

Il ne s'agit pas d'un modèle de mise en responsabilité sans en donner les leviers et moyens associés. Une organisation des forces sécurité au service de la conception des produits doit être mise en place en parallèle d'une organisation / responsabilité sécurité au sein de l'équipe produit en elle-même.

En effet, ce modèle de mise en responsabilité, pour qu'il soit fonctionnel en pratique, doit prendre en compte plusieurs aspects :

- les compétences sécurité doivent être réparties au sein de l'équipe produit (e.g. développeur, système, réseau...);
- un rôle sécurité doit être identifié au sein de l'équipe produit, ce rôle pourra être l'interface entre l'équipe produit et les acteurs sécurité (correspondants sécurité ou équipes sécurité);
- l'équipe produit ne peut construire de zéro tous les processus/services sécurité transverses, elle doit se reposer sur l'existant au niveau de l'organisation;
- l'équipe produit ne gère pas forcément les recrutements de la force au travail, un alignement avec le management est nécessaire pour disposer des ressources avec des compétences sécurité...

### **6.4. Organisation à plusieurs étages**

La mise en responsabilisation des équipes produits est facilitée par une organisation à plusieurs étages.

Nous avons initié ces changements en 2010, où nous avons proposé la construction d'un réseau de correspondants sécurité au plus proche des équipes produits. Ceci est un facilitateur et un moyen pour démultiplier les efforts sur la sécurité « terrain ». Cette initiative étant sponsorisée par notre management, elle a pu être mise en oeuvre dans les différentes entités de notre organisation. Il ne faut pas oublier qu'il s'agit d'un acte fort, car il faut faire un choix, et qui dit choix dit aussi renoncement sur d'autres priorités, et ce en faveur de la sécurité.

Nous avons donc une équipe sécurité centrale à l'organisation qui se repose sur un réseau de correspondants sécurité pour distiller les principes de sécurité et aussi recueillir l'ensemble des besoins / douleurs exprimées par les équipes produits. Cette approche a fait franchir un cap dans la prise de conscience et prise en compte des principes de sécurité. L'étape suivante, que nous

sommes en train de catalyser, est que les équipes produits bénéficient de force au travail déjà formées et appétentes en sécurité de leur domaine d'application (développement, intégration, déploiement...). Nous aurons alors une organisation à plusieurs étages, chacun alimentant l'autre, afin de fluidifier les échanges entre tous les acteurs sécurité. L'aspect humain est primordial, un des défis est que les personnes se connaissent bien et échangent de manière constructive avec de l'entraide jusqu'à aller à de la co-construction de services sécurité conjointement entre équipe sécurité, correspondants sécurité et équipes produits clientes de ces services sécurité.

## 7. Retour d'expérience sur l'outillage et son automatisation

Si un des objectifs est de responsabiliser les équipes produits, alors le choix des processus et des outils associés doit être guidé par ce paradigme.

Le changement de posture avec l'émergence du modèle client/fournisseur, afin que les équipes produits soient à terme demandeuses d'outils sécurité automatisés, est un changement majeur de culture : la sécurité intégrée est à la main des équipes produits avec le support d'équipes sécurité transverses, qui fournissent des services sécurité avec le support associé.

Pour y réussir, nous pouvons noter plusieurs défis (non exhaustif) à relever pour la construction de services :

- apport de valeur connue et reconnue ;
- facilité et flexibilité d'intégration dans le cycle de développement et les habitudes (potentiellement très diverses) des équipes produits ;
- personnalisation et configuration « à la main » des équipes produits ;
- intégration dans l'écosystème du système d'information de l'entreprise (gisement de données, corrélation, indicateurs...);
- coûts d'acquisition (Open Source ou licences commerciales) et d'intégration (dans le SI de l'entreprise ou en SaaS)...

En résumé, le défi majeur est qu'ils soient adoptés puis plébiscités par leurs utilisateurs : c'est LE facteur clé de réussite.

Dans ce chapitre, nous nous focaliserons sur les outils à intégrer dans un cycle de développement logiciel et leur automatisation.

### 7.1. Que choisir ? Quelles priorités selon le contexte ?

En tant qu'Orange, nous n'avons pas de spécificités vis-à-vis des principes de sécurité à implémenter. Cependant, en tant qu'offre de services Orange Grand Public exposés directement sur Internet, nous sommes particulièrement exposés et attractif pour tout un ensemble de classes d'attaquants.

Nous nous focaliserons sur des catégories de vulnérabilités atteignables par de l'outillage, principalement les vulnérabilités « d'implémentation ». En effet, les vulnérabilités « logiques » ne sont pas découvrables par de l'outillage standard, et en particulier lorsqu'il s'agit d'abus de parcours clients, qui par définition, exploitent des parcours clients autorisés sans vulnérabilité applicative, mais pouvant présenter une faiblesse ou une sensibilité à la fraude. Toutes ces classes de vulnérabilités sont naturellement hors de portée de l'outillage auquel nous allons



faire référence, et donc hors de portée d'une automatisation « standard ». Par contre, il est envisageable, dans ces cas de vulnérabilités « logiques » de détecter des biais dans le comportement de l'application ou des clients qui s'y connectent avec des mécanismes d'observabilité couplés à de la supervision sécurité par détection d'anomalie pour en découvrir des comportements suspects, cet aspect-là n'est pas dans le périmètre de cet article.

En conséquence, l'approche retenue est une amélioration de la qualité des produits vis-à-vis de la sécurité, ce qui rend l'initiative indépendante des classes d'attaquants.

Relativement aux vulnérabilités découvrables par automatisation, l'écosystème des outils sécurité dans un contexte de développement logiciel pourrait se diviser en plusieurs catégories (non exhaustif) :

- la découverte de failles de sécurité dans les développements en eux-mêmes (par de l'analyse en boîte blanche, boîte grise ou boîte noire);
- la vérification de vulnérabilités dans les composants logiciels utilisés par les produits (gestion des correctifs et des dépendances applicatives);
- la vérification de conformité des bonnes pratiques de configuration des composants utilisés par les produits (durcissement de configuration);
- la vérification de l'absence de fuite de secrets dans les dépôts de code source (GitLab, GitHub...)...

Et nous pouvons rajouter dans les architectures modernes « cloud », à la responsabilité des équipes produits :

- la vérification de vulnérabilités dans les images Docker, buildpacks CloudFoundry et consors...

Il est important de bien être au clair sur le domaine de responsabilités de chacun :

- le périmètre applicatif aux équipes produits applicatifs (e.g. les images Docker pour les composants applicatifs...);
- le périmètre infrastructure aux équipes produits infrastructure (e.g. les images Docker pour les composants d'infrastructure...)...

L'arrivée du cloud a le grand avantage de faciliter cette segmentation des responsabilités, les principes de sécurité devant être portés à la fois sur les composants d'infrastructure et sur les composants applicatifs, mais surtout en essayant au possible de découpler les deux afin de ne pas avoir de dépendances ou de zones d'ombres en terme de responsabilité. Ceci n'empêche d'ailleurs pas, de la même manière que les « Cloud Service Providers », que les équipes d'infrastructures fournissent des services sécurité utilisables par les équipes produits applicatives (e.g. Marketplace sécurité).

Nous n'aborderons pas d'autres principes de protection nécessaires dans un contexte Web tels que :

- la protection applicative une fois en production (filtrage applicatif Web);
- la supervision sécurité pour détecter/contrecarrer les attaques applicatives et les abus de parcours clients;
- la protection des parcours clients contre les attaques automatisées (partiellement ou non) via des systèmes d'authentification à multiple facteurs, et/ou de captcha et/ou d'antibot...

Nous nous focaliserons sur le contexte de l'automatisation d'outillage de sécurité en développement logiciel.

Concernant le contexte des environnements mobiles, nous recommandons le lecteur à se référer à l'article [17].

## 7.2. Les choix réalisés

Les choix à l'instant « t » sont liés à la maturité sécurité de l'organisation et des collectifs à l'instant concerné. Les choix peuvent changer dans le temps afin de franchir des paliers, une fois les paliers précédent stabilisés.

Dans cette partie, nous décrivons l'ensemble des avantages et inconvénients de ces différentes approches.

### 7.2.1. L'écosystème

L'écosystème des outils de sécurité dans un contexte d'intégration continue pourrait se diviser en plusieurs catégories :

- l'analyse de code source en boîte blanche, aussi appelée « Static Analysis Security Testing » (SAST);
- l'analyse dynamique en boîte noire, aussi appelée « Dynamic Analysis Security Testing » (DAST);
- l'analyse dynamique en boîte blanche, aussi appelée « Interactive Analysis Security Testing » (IAST)...

Lors de ces premières réflexions, en 2010, seules les technologies SAST et DAST avaient une maturité suffisante, que ce soit dans des produits commerciaux ou Open Source.

### 7.2.2. Le choix SAST

Dans notre contexte, nous avons choisi en 2010 d'évaluer les technologies SAST et DAST, et nous avons considéré que la technologie SAST était la plus à même de faire progresser l'organisation, et ce pour plusieurs raisons :

- très forte adhérence avec les équipes de développement, ce qui nécessite alors une implication forte de ces équipes en tant que parties prenantes des principes de sécurité (ce qui est alors en cohérence avec la promotion de la mise en responsabilité de la sécurité de leurs produits);
- l'approche SAST est en théorie plus en profondeur que DAST car dispose de la vue en boîte blanche;
- les langages PHP et Java étant majoritaires dans notre contexte, une solution du marché était particulièrement efficace sur ces langages;
- l'intégration de ce type d'outil est toute naturelle avec des principes d'automatisation d'analyse de code source en intégration continue;
- l'approche DAST se heurte à des problématiques de mise en oeuvre opérationnelles dès lors que nous avons des composants supervisés dans de nombreuses entités : en effet, les tests en boîte noire ne doivent pas perturber les sondes de supervision classique, de supervision sécurité ni bien sûr le fonctionnement nominal du service audité.

Nous avons évalué trois produits commerciaux et un Open Source, les résultats partiels étant disponibles sur [18], et en 2011, nous avons déployé deux solutions commerciales pour couvrir avec efficacité pour l'une les langages PHP et Java, et pour l'autre les langages C/C++.

Il est à noter que ces choix doivent être remis en cause régulièrement car le défi actuel est d'arriver à maintenir une efficacité de détection de vulnérabilités avec un contexte où les langages de développement évoluent beaucoup, et tout particulièrement dans le mode du Web. Par exemple, dans notre contexte, PHP était prédominant pour réaliser des interface homme-machine, et maintenant il est de plus en plus courant d'avoir des technologies à base JavaScript à la fois côté serveur (Node.js) et côté client (Angular), le tout en mode « micro services ».

Dans le monde du développement sur environnements mobiles, c'est tout aussi le cas, avec les classiques (Java/Android, iOS/Obj-C, iOS/Swift) mais aussi les bibliothèques telles que REACT.js.

Nous comprenons alors toutes les difficultés techniques que rencontrent ces SAST, il faut savoir comprendre le langage afin de le modéliser, et les solutions actuelles SAST éprouvent beaucoup de difficultés à être efficaces sur ces langages qui sont par nature difficiles à modéliser (extrêmement dynamiques et très forte évolutions).

### 7.3. L'automatisation

Dans cette partie, nous décrivons les principes d'automatisation retenus, et en particulier dans quelle mesure l'humain doit intervenir pour qualifier les remontées des outils de sécurité.

En effet, la cible étant la réduction de volumes de vulnérabilités présentes en production, il faut attacher une vigilance toute particulière à ce que l'équipe monte en maturité dans l'analyse des vulnérabilités remontées par les outils. Ceci ne peut pas se faire immédiatement, et des phases progressives sont nécessaires pour que les équipes appréhendent l'outillage, comprennent les remontées, et réalisent les correctifs au fur et à mesure.

Et lorsque la maturité est suffisante, il est possible pour l'équipe, d'implémenter des verrous au niveau de leur intégration continue afin de mettre en échec tout « soumission de code » présentant des vulnérabilités d'une criticité définie. Implémenter ces principes par paliers est nécessairement à la main des équipes pour éviter toute frustration de marche trop haute à passer à l'initialisation de la démarche.

En adoptant cette approche de « petit pas », l'ambition est de rendre les équipes ambassadrices de ces principes de sécurité automatisés à bas coûts et peu de douleurs d'intégration.

### 7.4. L'orchestration de l'automatisation

Nos premiers déploiements SAST datant de 2011, nous avons dès le début identifié le pré-requis d'intégration/déploiement continu déjà en place dans les habitudes des équipes. En effet, si les principes d'automatisation ne sont pas déjà en place pour des invariants classiques (tests unitaires, tests fonctionnels, tests de non régression...), il devient alors difficile de passer la marche de l'automatisation sécurité.

Nous avons commencé avec les équipes ayant déjà une maturité suffisante dans leurs processus et l'utilisation d'outillage CI/CD tels que Jenkins à l'époque.

Ensuite, notre organisation a basculé sur la solution GitLab, une migration des processus d'intégration des services sécurité s'en est naturellement suivie. Aujourd'hui, la solution CI/CD

est GitLab pour l'ensemble des composants développés dans notre organisation.

Comme de tout, il est critique de mettre les efforts pour accompagner les équipes à intégrer l'outillage d'automatisation sécurité. La mise en place d'une organisation à cet effet prend tout son sens, il ne s'agit pas de d'imposer un standard sans donner les moyens d'y arriver ni d'accompagner de manière rapprochée pour donner du sens et faire adhérer les équipes à l'initiative.

Nous pouvons créer des modèles (GitLab Templates) qui seront alors utilisés par les équipes produits directement dans leurs projets GitLab.

Le principal intérêt d'utiliser des modèles est de rajouter une couche d'abstraction du service sécurité utilisé en dessous, ce qui entraîne moins d'actions de la part de l'équipe produit, une uniformisation de l'intégration, et la capacité à modifier facilement le modèle en répercutant ces modifications de manière uniforme sur tous les projets GitLab.

Le point essentiel à retenir est qu'il est nécessaire d'avoir un socle commun qui facilite l'intégration des services sécurité dans les chaînes CI/CD des équipes produits, et en ce sens, les acteurs sécurité doivent se positionner en « offre de services ».

## 7.5. Automatisation des technologies SAST

Les technologies SAST sont très susceptibles aux faux positifs, qui s'ils sont nombreux, créent un sentiment de rejet du fait de travail laborieux et inutile sur des vulnérabilités non pertinentes. Un élément clé, là-aussi, est de n'offrir un service SAST que si nous avons un volume de faux positifs acceptables aux yeux des équipes produits.

Il est possible de les réduire avec plusieurs approches :

- modifier les règles qui déclenchent trop de faux positifs pour que toutes autres analyses en bénéficient ;
- marquer les faux positifs pour qu'ils n'apparaissent plus dans les prochaines analyses ;
- remonter à l'éditeur de logiciel les faux positifs afin que le support technique fasse faire une évolution qui corrige le défaut ;
- être le plus à jour possible afin de bénéficier des évolutions logicielles du produit SAST...

Les effets de bord positifs que nous avons pu constater sont :

- réduction du code « mort », du fait qu'il est plus facilement identifié lorsque des vulnérabilités sont remontées ;
- découverte de vulnérabilités tierces, du fait de la lecture plus approfondie du code source lors de l'analyse des vulnérabilités remontées ;
- montée en compétence des développeurs du fait de dialogues plus réguliers avec les correspondants et équipes sécurité ;
- capacité à produire des indicateurs de qualité sécurité avec une ambition à atteindre (e.g. pas de vulnérabilité de niveau critique avant passage en production)...

Mais bien entendu, ce n'est pas sans des limites qu'il convient de rappeler :

- ne découvrent que des erreurs d'implémentation ;
- ne couvrent pas efficacement tous les langages ;
- très sujet aux faux positifs...

Nous avons pris parti pour couvrir l'ensemble des langages de se reposer sur deux produits commerciaux du marché. Il convient de les évaluer régulièrement pour sonder le marché à

la vue des évolutions sur les habitudes/langages de développement. En pratique, ces produits commerciaux ont des approches différentes et souvent complémentaires qui leur revêt une aisance ou au contraire des difficultés selon les langages analysés.

Le principal reproche adressé aux SAST, est qu'il est nécessaire d'avoir une analyse manuelle des remontées, et que ces remontées analysées doivent persister dans le temps pour que ce qui a été marqué comme étant un faux positif ne pollue plus les analyses ultérieures. En effet, cette partie est malheureusement non automatisable, seule l'automatisation de l'outillage l'est, ainsi que l'orchestration de la création des projets et des droits d'accès pour que l'équipe produit puisse réaliser l'intégration la plus simple possible.

Il est aussi envisageable d'automatiser la création des « issues » dans GitLab ou des « cartes » dans JIRA, mais nous avons constaté en pratique que les équipes n'ont pas été friandes de ces fonctionnalités, probablement dû au fait que cela est peu automatisable (quid de la création de dizaines de vulnérabilités à traiter?) et nécessite dans tous les cas une analyse manuelle pour distinguer la vulnérabilité du correctif (un correctif peu corriger des dizaines de vulnérabilités similaires). Et pour compléter, les équipes produits gèrent leur « backlog » avec leurs habitudes grâce au pilotage du Product Owner, ce qui nous fait atteindre les limites en terme d'automatisation.

En résumé, l'approche SAST est essentielle pour passer un cap mais elle nécessite des prérequis en terme d'organisation et d'acceptation de la part des équipes produits qui ne s'acquièrent que dans le temps. Il faut procéder par petit pas et co-construire de manière la plus étroite possible avec les équipes produits, qui sont les utilisatrices de ces solutions, même si originellement issues d'une initiative sécurité.

## 7.6. Automatisation de la gestion des correctif et des dépendances applicatives

En complément de la découverte de vulnérabilités applicatives dans le code développé par les équipes produits, il est tout aussi important d'avoir une politique de gestion des correctifs sur les composants utilisés par les produits concernés.

En effet, nous pouvons distinguer les failles de sécurité connues (i.e. référencées en tant que Common Vulnerabilities and Exposures (CVE) [19] dans :

- des composants du socle exécutant les composants applicatifs (e.g. le système d'exploitation dans le cadre d'une machine physique, d'une machine virtuelle ou d'une image Docker);
- des bibliothèques applicatives utilisées par les composants applicatifs.

Nous nous sommes basés sur les outils Clair/PaClair [20, 21] pour découvrir les vulnérabilités dans les images Docker produites par les équipes produits. En standard, l'outillage apporte cette analyse, et nous avons développé des composants « maison » implémentant l'analyse des dépendances applicatives sur les langages les plus prioritaires dans notre contexte (i.e. PHP, Java, Node.js, Golang), de la même manière que [22].

Contrairement aux SAST, nous ne sommes pas confrontés aux faux positifs : toute vulnérabilité remontée est réelle et est échelonnée avec un niveau de criticité. Il n'y a pas lieu de réaliser une analyse manuelle, et les résultats peuvent être compris par les équipes produits aisément. Par contre, il subsiste toujours une part de priorisation des vulnérabilités remontées car l'approche risque ne peut être prise en compte automatiquement. En effet, quid d'une vulnérabilité de

criticité haute, mais qui de part l'urbanisme du produit, ne peut être atteignable ?

Dans ce cas-là, deux approches sont possibles, soit avec une rigueur sur l'automatisation, en s'imposant un prérequis de ne plus avoir de vulnérabilité, c'est l'approche « qualité logicielle », soit en reposant sur les compétences sécurité pour qualifier l'exploitabilité de la vulnérabilité remontée, c'est l'approche « risque ». Nous considérons que la première approche est plus vertueuse et finalement plus simple à s'y conformer, puisque systématique et intégrable dans les processus d'automatisation.

En se greffant au GitLab, il est possible de rajouter des notes ou badges sur les projets GitLab en fonction des résultats des analyses sécurité avec un accueil très positif des équipes produits. Nous avons constaté que l'émulation entre équipes et la gamification étaient des moyens ludiques et efficaces pour déployer une nouvelle approche ou un nouveau service.

### **7.7. Automatisation de la vérification de conformité dans les usines à images**

Dans un contexte d'orchestration Docker (Docker Swarm, Kubernetes), les équipes produits construisent leurs images et les envoient vers une usine à image grâce leur chaîne d'intégration continue. Il leur est ensuite possible de déployer ces images dans des conteneurs qui seront alors orchestrés par l'infrastructure. Nous y voyons alors un intérêt tout particulier : l'usine à image est un passage obligé avant tout déploiement sur les infrastructures. Tout passage obligé est une aubaine pour assurer conformité et sécurité, les fameux « Security Gates » chers à l'OWASP.

Nous avons donc développé un outillage « maison » qui réalise une série de vérifications de conformité, dont celles présentées au paragraphe précédent, qui se greffe dans le processus d'envoi vers l'usine à image. Nous bénéficions alors d'un service en passage obligé sur lesquels nous avons la main pour effectuer tout un ensemble de vérifications. Si les vérifications échouent, lors le « job » GitLab du composant concerné est échoué avec une explication des non-conformités ainsi qu'un détail sur les opérations à effectuer pour franchir l'étape.

Le principal obstacle à cette approche est qu'elle nécessite une opération de configuration par l'équipe produit dans leur chaîne CI/CD pour utiliser l'outillage de vérification.

Le principal objectif est d'arriver à ce que cette étape de vérification soit systématiquement implémentée par les équipes produits, et là, deux approches sont possibles : imposer ou accompagner. Évidemment, imposer n'est envisageable que lorsque la maturité est suffisante et que tout le monde est convaincu de l'efficacité/apport de la solution, i.e. lorsque la solution est un standard et qu'il ne faut plus régresser. Mais avant d'y arriver, il faut réaliser des phases d'accompagnement successives auprès des équipes.

Aujourd'hui, plus de 80% des composants applicatifs éligibles passent par cette phase de vérification avant envoi dans l'usine à image.

### **7.8. Les défis actuels et à venir**

Une difficulté rencontrée est la multitude des services sécurité sans qu'ils aient une intégration/utilisation commune, en effet, si plusieurs services sécurité sont utilisés, alors les résultats de ces services sécurité seront analysables/consultables différemment. Cela implique un effort supplémentaire d'intégration et des efforts humains de montée en compétences sur chacun d'entre eux, sur des interfaces hétérogènes. L'idéal serait de disposer d'une interface commune

à tous les outils jouant le rôle d'abstraction/orchestration des couches inférieures, quelques initiatives existent à ce sujet comme [23], mais d'autres problématiques sont alors rencontrées, ne donnant pas lieu à de solution aisée de prime abord.

Nous n'avons pas abordé d'autres thèmes comme l'automatisation des déploiements dans les environnements Cloud qu'ils soient internes ou publics, avec des notions de « Infrastructure as Code » où il sera nécessaire de s'appuyer sur de l'automatisation de l'ensemble des processus pour à la fois faciliter les déploiements et enlever les douleurs ressenties par les équipes produits tout en ne régressant pas au niveau de la sécurité des produits et architectures.

Dans tous les cas, le principal défi est de maintenir et de développer un niveau de culture sécurité au sein des équipes produits avec le support d'un réseau local d'acteurs sécurité.

## 8. Conclusion

En partant du postulat d'un engagement managérial fort, une organisation peut alors en découler. Ces dernières années, le changement de culture a été marqué par l'avènement de l'agilité et du DevOps. Ce sont à la fois des risques et des opportunités, il s'agit alors d'en accompagner le mouvement avec une organisation au service d'objectifs clairs sur la responsabilisation des équipes produits avec tout le support nécessaire des acteurs sécurité « terrain » et « centralisés ». L'automatisation en découle alors tout naturellement puisqu'il s'agit de passer à l'échelle et de pouvoir suivre la cadence imprimée par de nouvelles fonctionnalités produit déployées en continu. C'est alors un cercle vertueux qui s'engage avec des tâches manuelles plus pertinentes et gratifiantes, en s'appuyant sur un socle d'automatisation qui évoluera dans le temps au fur et à mesure des paliers de la montée en maturité des différentes équipes DevOps, accompagnées d'une équipe de sécurité transverse en position de facilitateur/accompagnateur de cette démarche. L'automatisation est un élément clé de réussite, mais les prérequis en terme de culture et de moyens sont nécessaires pour y parvenir.

## Références

- [1] Alexa, Alexa top sites france, 2021. URL : <https://www.alexa.com/topsites/countries/FR>.
- [2] YesWeHack, Yeswehack bug bounty programs, 2021. URL : <https://yeswehack.com/programs>.
- [3] Wikipedia contributors, Diginotar — Wikipedia, the free encyclopedia, 2021. URL : <https://en.wikipedia.org/w/index.php?title=DigiNotar&oldid=1021304505>, [Online ; accessed 18-May-2021].
- [4] DarkReading, Sony data breach cleanup to cost \$171 million, 2021. URL : [https://www.darkreading.com/attacks-and-breaches/sony-data-breach-cleanup-to-cost-\\$171-million/d/d-id/1097898](https://www.darkreading.com/attacks-and-breaches/sony-data-breach-cleanup-to-cost-$171-million/d/d-id/1097898).
- [5] ANSSI, Protection des oiv en france, 2021. URL : <https://www.ssi.gouv.fr/entreprise/protection-des-oiv/protection-des-oiv-en-france>.
- [6] U. Européenne, General data protection regulation, 2021. URL : <https://gdpr.eu>.
- [7] PCI, Pci security standards council, 2021. URL : <https://www.pcisecuritystandards.org>.

- [8] Microsoft, Security development lifecycle, 2021. URL : <https://www.microsoft.com/en-us/securityengineering/sdl>.
- [9] OWASP, Software assurance maturity model, 2021. URL : <https://owaspsamm.org>.
- [10] BSIMM, Building security in maturity model, 2021. URL : <https://www.bsimm.com>.
- [11] Atlassian, Atlassian, 2021. URL : <https://www.atlassian.com/fr/agile>.
- [12] ANSSI, Agilité et sécurité numérique : méthode et outils à l'usage des équipes projet, 2021. URL : <https://www.ssi.gouv.fr/guide/agilite-et-securite-numeriques-methode-et-outils-a-lusage-des-equipes-projet>.
- [13] Atlassian, Atlassian, 2021. URL : <https://www.atlassian.com/fr/agile/devops>.
- [14] Agile Authors, Manifesto for agile, 2021. URL : <https://agilemanifesto.org>.
- [15] B. T. Klein, The devops : A concise understanding to the devops philosophy and science, 2021. URL : <https://www.osti.gov/biblio/1785164>.
- [16] D. D. Zovi, Black hat usa 2019 keynote : Every security team is a software team now, 2019. URL : <https://www.youtube.com/watch?v=8armE3Wz0jk>.
- [17] A. De Bock, V. Nadal, Integrating mobile applications security into software development lifecycle, 2021. URL : <https://www.cesar-conference.org/>.
- [18] L. Butti, O. Moretti, Analyse statique de code dans un cycle de développement web, retour d'expérience, 2012. URL : <https://speakerdeck.com/0xd012/gs-days-2012-analyse-statique-de-code-dans-un-cycle-de-developpement-web-retour-dexperience>.
- [19] CVE, Common vulnerabilities and exposures, 2021. URL : <https://cve.mitre.org>.
- [20] CoreOS, Clair, 2021. URL : <https://github.com/quay/clair>.
- [21] Yebinama, Paclair, 2021. URL : <https://github.com/yebinama/paclair>.
- [22] OWASP, Dependency check, 2021. URL : <https://owasp.org/www-project-dependency-check>.
- [23] DefectDojo, Defectdojo, 2021. URL : <https://github.com/DefectDojo/django-DefectDojo>.