< GOUVERNANCE SSDLC >

GROUPE DE TRAVAIL CYBERSECURITE AGILE

CAMPUS CYBER 2025

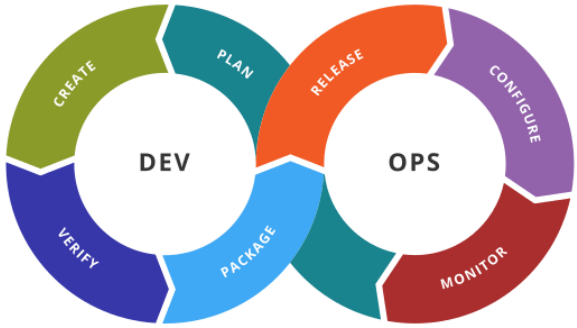
< SOMMAIRE >

1. CONTEXTE

En 2011, Marc Andreessen a publié un article dans le Wall Street Journal intitulé « Why Software is Eating The World »[1]. Cet article est considéré comme une des références pour comprendre l'évolution technologique actuelle. L'auteur y expliquait notamment comment le logiciel allait remettre en cause les acteurs historiques déjà en place sur leurs marchés respectifs. Le logiciel a cette vertu : il permet d'innover très rapidement, mais cela a aussi pour corollaire une obligation d'innover en permanence pour continuer à se différencier de ses concurrents.

En regardant la décennie passée, force est de constater que ce besoin de vitesse d'exécution a été au cœur de toutes les transformations : l'agilité, son déploiement à large échelle, le cloud, les architectures à base de micro-services, le CI/CD, le DevOps, le SRE[2] ont pour point commun de contribuer à la création de valeur, à un rythme soutenu et soutenable, tout en s'assurant une bonne qualité d'exécution.

Le DevOps est souvent associé à cette image :



Source : The DevOps Platform for agile business (gitlab.com)

Or, cette image correspond à un sous-ensemble de DevOps : le cycle de vie du développement logiciel (SDLC en anglais, pour Software Development LifeCycle) dans un contexte DevOps.

DevOps ne se limite pas au SDLC, c'est avant tout un ensemble de pratiques , une culture qui favorise la collaboration et la communication entre les développeurs et les exploitants, qui tire parti de l'automatisation du SDLC, qui met en place des mécanismes d'amélioration continue, et qui favorise le développement de la valeur pour l'utilisateur final via le déploiement de petits incréments combinés à une boucle retour rapide.

Le DevOps agit alors comme moyen à la mise en œuvre des méthodologies Agile, facilitant la mise en place d'itérations de développement courtes avec un impact positif sur le Time-To-Market.

Face à cette accélération dans les cycles de developpement, la sécurité fait face à un besoin nouveau de célérité, au risque de réduire ou nullifier celle acquise à l'aide du DevOps et des pratiques Agiles. En effet, effectuer les activités de sécurité en fin de projet est un anti-pattern Agile, avec des temps de validations pouvant parfois dépasser la durée d'un sprint.

Face à ce constat, a été largement introduit le concept de « Shift-Left Security Testing ».
Le shift-left est un concept vieux de plus de 20 ans issu des pratiques du Testing, selon lequel plus un test est effectué tôt (à gauche de la ligne temporelle du projet), plus un bug identifié est simple et rapide à corriger. Le concept a par la suite été étendu à la sécurité, principalement comme une réponse à la nécessité d'intégrer la sécurité dans les cycles de développements Agile plutôt qu'à leurs fins.

Si le concept est facile à comprendre, il reste toutefois très théorique quant à son implémentation. Heureusement, il existe le DevSecOps : de la même manière que le DevOps vient supporter les développements Agile, le DevSecOps vient supporter à la fois l'Agilité mais aussi les activités de sécurité qui y sont « shift-lefté » en outillant et systématisant les tests de sécurité pendant la phase de developpement.

Toutefois, l'outillage ne suffit pas à répondre seuls aux objectifs de sécurité, des connaissances sécurité étant nécessaire pour les utiliser et traiter leurs sorties. Or la réalité dans les organisations est que le nombre d'experts en cybersécurité est bien moindre que le nombre de développeurs ou d'exploitants. De fait, l'enjeu pour les entreprises est d'arriver à mettre à l'échelle l'accompagnement sécurité des projets avec un nombre de ressources sécurité limité. Cela passe par la mise en place d'une gouvernance adaptée qui, pour Gleen Willson[3], repose sur trois piliers :

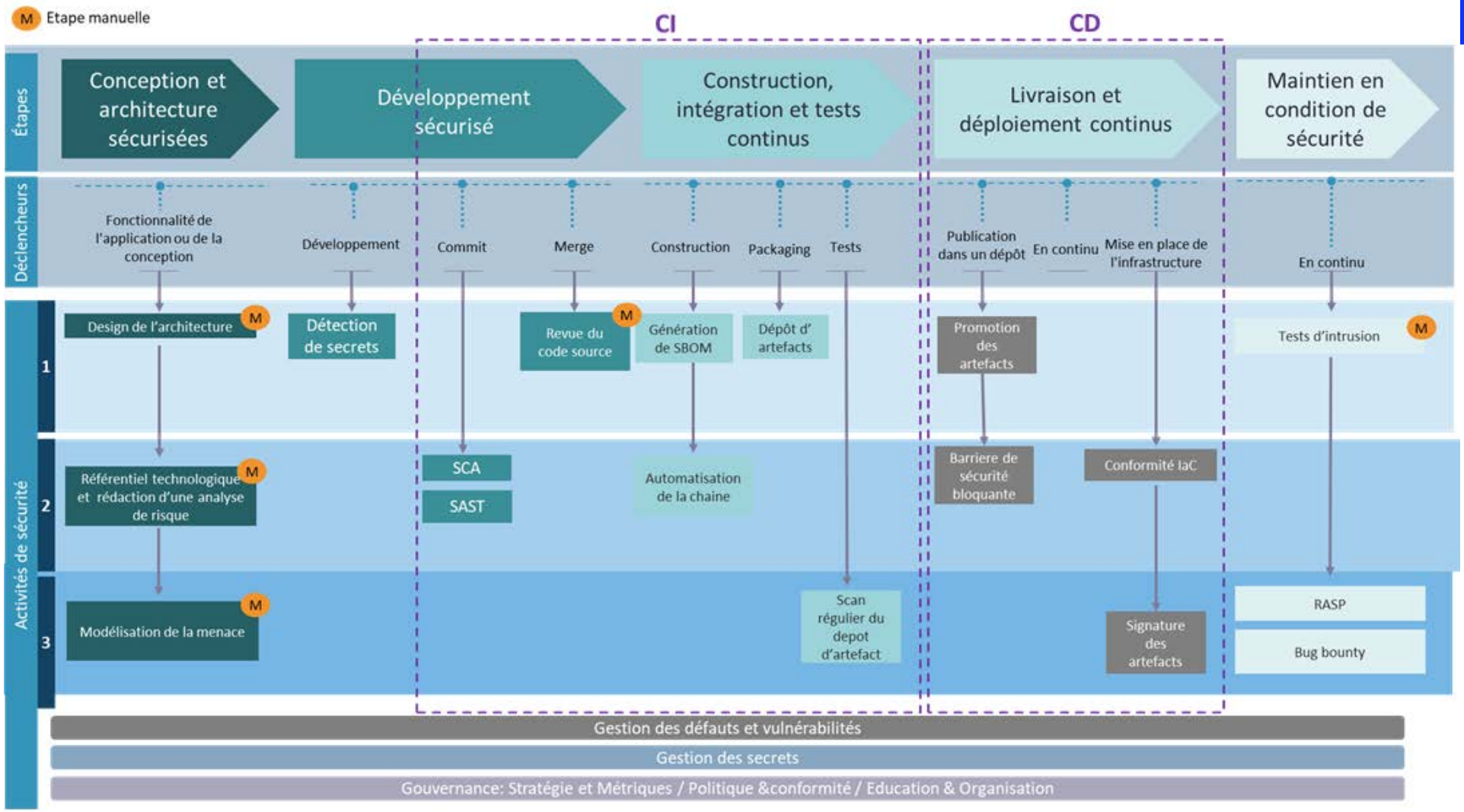
1. La formation : il faut à la fois former les experts en cybersécurité et sensibiliser les développeurs et les exploitants. De plus, la présence d'un Security Champion* dans les équipes facilite le partage d'informations, permet de garantir l'exécution des activités de sécurité et met en place un lien avec l'équipe centrale de gouvernance sécurité pour les grandes organisations (répondant ainsi au besoin de scalabilité évoqué précédemment) ;
2. La sécurité dès la conception (ou Security By Design en anglais), car intégrer la sécurité dès le début est plus efficace, et coûte moins cher ;
3. L'intégration et systématisation des activités de sécurité au sein duSoftware Developement LifeCycle, c'est-à-dire avoir un SDLC sécurisé ou Secure SDLC (SSDLC) en anglais, objet de ce document.

L'ensemble des outils et des pratiques permettant de sécuriser son SDLC est décrit dans le document « Cartographie des Technologies »[4] publié par le Campus Cyber.

* Terme consacré par la littérature. Celui-ci peut être mal interprété car l'objectif n'est pas tant d'avoir un expert en cybersécurité, mais un sachant qui est au carrefour entre son équipe, les autres Security Champions et potentiellement l'équipe de gouvernance sécurité, afin de faciliter le partage d'informations et de l a connaissance.

< CONTEXTE >

L'ensemble des étapes et des activités est listé dans le schéma ci-dessous :



< CONTEXTE >

Historiquement, l'intégration de la sécurité au sein du SDLC a commencé par l'utilisation d'outils de SAST, continuité logique à l'utilisation d'outils d'analyse de la qualité du code fonctionnant sur un modèle proche, qui étaient largement utilisés par les projets ; si bien que certains outils de qualité sont venus enrichir leurs règles de détection pour identifier des vulnérabilités.

Aujourd'hui, les entreprises sont nombreuses à prioriser leur investissement sur la mise en place d'outils qui peuvent générer des Software Bill Of Materials (inventaire de dépendances) afin d'utiliser des outils de Software Composition Analysis (SCA).

Leur objectif est de prévenir des attaques via les vulnérabilités publiquement connues dont les projets héritent via les frameworks ou bibliothèques utilisées et fournies par des tiers. S'il n'a pas de cible très précise, un attaquant a tout intérêt à attaquer via ce vecteur d'attaque, car d'une part le code tiers représente une part très importante du code déployé*, et d'autre part il est utilisé par une multitude de cibles potentielles.
Ces enjeux ont également été bien identifiés dans la directive NIS2[7] qui sera mise en application en France en octobre 2024.

Pour une entreprise, il n'est jamais facile d'évaluer sa maturité en termes de sécurisation du SDLC, ou d'identifier les actions les plus prioritaires à lancer. Une approche via un modèle de maturité facilite cette démarche, elle permet à la fois de s'évaluer, et d'identifier les actions à mener. Ce document a été rédigé dans cet état d'esprit.
Il s'est à la fois inspiré du modèle de maturité SAMM de l'OWASP[8], du document « Cartographie des Technologies » publié par le Campus Cyber, et de l'expérience des professionnels ayant participé à sa création.

*L'entreprise vdoo[5] indique que presque tous les projets utilisent de l'open source, et que l'open-source représente 85% à 97% de l'ensemble du code source en termes de volume de code. L'entreprise Synopsys partage des chiffres similaires[6].

2. PRIORISATION

2.1 NIVEAU INITIAL ET PREREQUIS

Comme base, il est primordial d'avoir à disposition un inventaire applicatif aussi exhaustif que possible et d'avoir défini dans l'entreprise une classification des applications en fonction de la criticité métier et de l'IT générale.

2.2 DÉFINITION DES TROIS NIVEAUX

Tématique	Niveau 1	Niveau 2	Niveau 3
Niveau de Sécurité	Minimum de sécurité	Bon niveau de Sécurité Applicative	Sécurité globale et appliquée
Couverture	Systémisation de l'activité sécurité (manuelle ou peu d'automatisation)	Systémisation de l'activité Sécurité (automatisation en place)	100% des applications à risque et sous règlementation couvertes
Gouvernance	Identification des objectifs et moyens de mesurer l'efficacité des activités de sécurité	Etablir un plan d'actions commun stratégique pour la sécurité logicielle au sein de l'organisation	Adapter les efforts de sécurité en fonction des indicateurs organisationnels et la criticité des actifs
Responsabilité	Equipe Sécurité	Equipes Dev/AppSec & Ops/Securité	Equipes Dev/Sec/Ops
Périmètre	Applications critiques business / IT, critiques et vitales	Augmentation de la couverture (50% parc), classification des applications (+ critique vers - critique)	Parc applicatif couvert et profil de risques définis entièrement
Processus	Processus à définir ou définis (sans implémentation concrète)	Procédures définies et outillées	Standardiser les pratiques à échelle
Organisation*	A définir ou définie avec peu de ressources Sécurité ET définition du rôle de Security Champion	Définie avec des ressources Sécurité identifiées par les équipes Dev et Ops ET définition du rôle de SME AppSec	Communautaire impliquant les Security Champions, les SMEs et les équipes Sécurité

3. CONCEPTION ET ARCHITECTURE

3.1 OBJECTIFS

Lors de cette étape de définition de conception de l'applicatif ou de la fonctionnalité (feature) complémentaire et de l'architecture associée, l'objectif pour les équipes de sécurité est :

Établir les niveaux de risques fonctionnels et non-fonctionnels
Définir les exigences de sécurité et l'architecture applicative adaptées aux contextes de l'entreprise et aux spécificités métier

3.2 PREREQUIS

Pour réaliser cette étape, il est nécessaire de pouvoir s'appuyer sur différents éléments afin de s'assurer que les exigences de sécurité soient prises en compte :

- Un parrainage du top management : l'implication du top management permet d'assurer l'engagement des équipes à la prise en compte des exigences sécurité tout au long du projet.
- Définition de piles techniques et briques transverses de sécurité à l'entreprise (Authentification, réseaux, etc.)

3.3 IMPLEMENTATION

NIVEAU 1

Thématiques	Activités	Livrables
Exigences de sécurité	Définition du type de données et besoins de sécurité associé (Disponibilité/Intégrité/ Confidentialité/Traçabilité)	Profil minimal de sécurité en termes de DICT couvrant la donnée et l'exposition sur Internet
Architecture	Revue de l'architecture applicative	Document HLD/LLD Modèle d'autorisations : RBAC, etc.
Technologies	Adapter les choix technologiques en fonction du risque	Recommandations des choix techniques et de frameworks
Processus	Définition du RACI au sein du projet pour adresser les activités sécurité	RACI des activités sécurité
Contrôles sécurité	Définition des contrôles de sécurité en se basant sur les exigences standards et règlementaires	Liste des contrôles minimaux applicables

NIVEAU 2

Thématiques	Activités	Livrables
Exigences de sécurité	Rédaction d'une analyse de risques pour définir les mesures spécifiques à mettre en œuvre	Profil de sécurité (Finance/HR/etc.) applicable à l'application et association à un niveau d'accompagnement
Architecture	Choix des piles techniques et briques transverses de sécurité	Liste des composants de sécurité à utiliser
Technologies	Définir un référentiel des technologies pour le processus de développement	Référentiel des technologies recommandées
Processus	Sélection des outils de sécurité utilisés dans la chaine CI/CD	Liste des outils sélectionnés
Contrôles sécurité	Définir un modèle de menaces global à l'organisation et identifier les cas de fraudes et d'abus métier à surveiller	Liste d'exigences de sécurité complémentaires à appliquer

NIVEAU 3

Thématiques	Activités	Livrables
Exigences de sécurité	Mettre en place un framework global d'exigences de sécurité pour les projets et sur l'utilisation des dépendances tierces	Document et processus standardisés Liste des exigences implémentables pour les développeurs
Architecture	Définition d'architectures de référence évolutives	Modèles d'architecture
Technologies	Formalisation et déploiement de technologies validées	Référentiel des technologies imposées, du processus d'exception et des services transverses (IDP, gestion des logs)
Processus		
Contrôles sécurité	Définition d'un Threat modeling au niveau du projet en s'appuyant sur une méthodologie globale	Modèle de menaces projet Processus de mise à jour du modèle de menaces global

3.4 BONNES PRATIQUES

Les bonnes pratiques de sécurité associées à un projet sont aussi applicables au sein du SSDLC :

- Ajout d'exigences de sécurité dans les différents contrats des fournisseurs.
- Mise en place de KPI sécurité pour s'assurer de la mise en œuvre des étapes, la réalisation des livrables définis et valider l'efficacité du modèle.
- Mise en place de formations et d'outils facilitant l'implémentation de pratiques standardisées pour les différents acteurs.

3.5 ANTIPPATERNS

Ces pratiques sont contre-productives pour la sécurisation du SDLC :

- Analyse de risques sans recommandation de sécurité associées.
- Suppression des Security stories au profit de nouvelles fonctionnalités.

4. THREAT MODELING (MODÉLISATION DE MENACES)

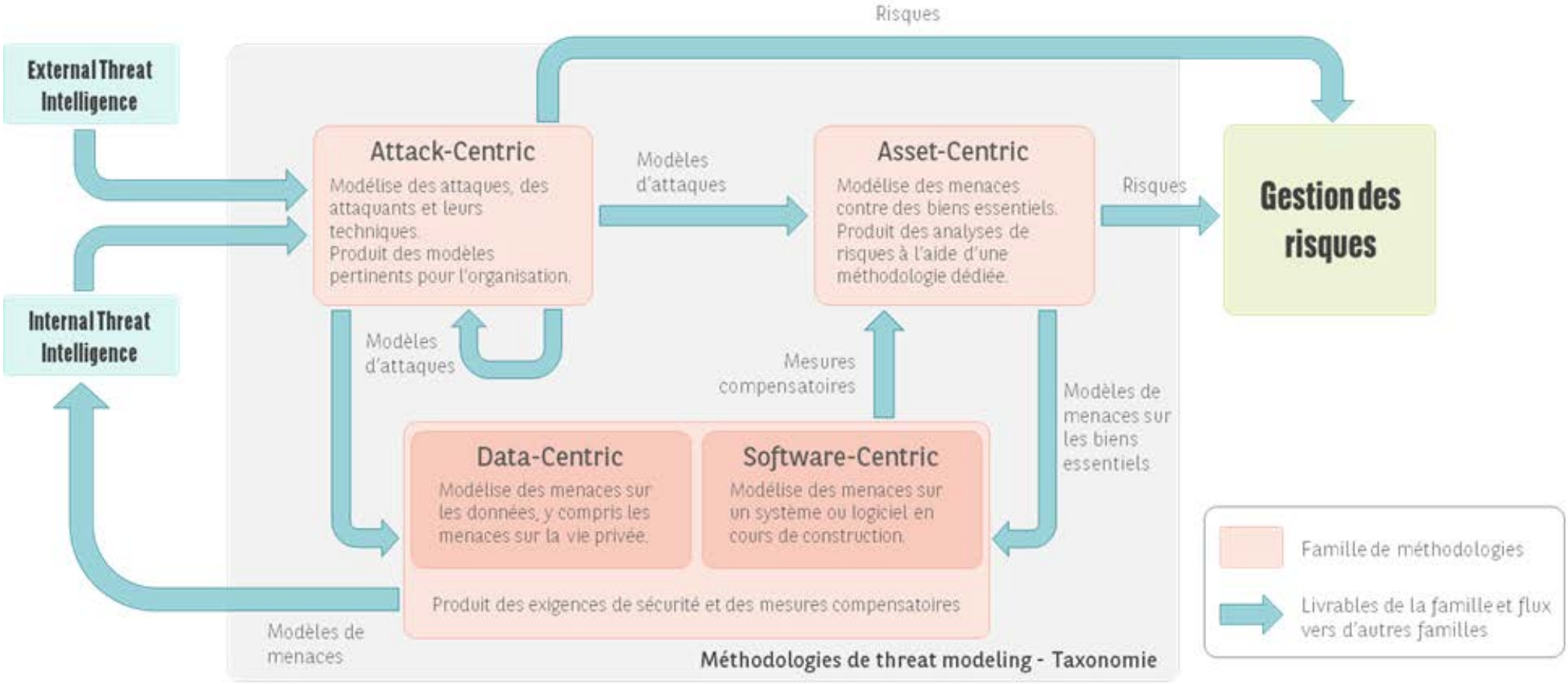
4.1 OBJECTIFS

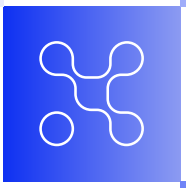
Le threat modeling est un terme générique pour décrire un ensemble d'activités qui partent du même besoin (identifier des menaces contre un système), mais dont les objectifs ne sont pas forcément compatibles.

En nous basant sur une taxonomie originellement proposée dans [1] , il est possible de classer les méthodologies de threat modeling en fonction de leurs objectifs :

- Les méthodologies « attack-centric », ou « attacker-centric », dont le but est de créer des modèles d'attaques et d'attaquants qui sont spécifiques à l'organisation qui applique la méthode.
- Les méthodologies « asset-centric », ou « impact-centric », dont le but est d'identifier les menaces qui portent sur les biens essentiels et d'évaluer les risques. Ces méthodologies sont très proches des méthodes d'analyse de risques « traditionnelles », mais intègrent certains aspects spécifiques du threat modeling, comme par exemple l'utilisation de diagrammes de flux comme éléments principaux.
- Les méthodologies « data-centric » et « software-centric », qui sont groupées dans la même famille, et dont l'objectif est d'analyser les menaces qui pèsent un système en cours de conception afin d'identifier des mesures de sécurité spécifiques et pertinentes et d'éviter des vulnérabilités liées au design du système.

Chacune de ces méthodologies peut être implémentée indépendamment par une organisation, mais il est également possible d'utiliser les spécificités de chacune afin de couvrir un spectre plus large de menaces et d'activités. Chaque méthodologie produira des livrables qui pourront être consommés par d'autres méthodologies, ou des processus externes :





Dans le cadre du SSDLC la pratique du threat modeling utilise des méthodologie software-centric afin de répondre à ces objectifs :

- Identifier les menaces et les attaques spécifiques au contexte applicatif.
- Identifier au plus tôt les vulnérabilités dans le design applicatif pour le corriger, par opposition à des vulnérabilités d'implémentation.
- En déduire des mesures de sécurité pertinentes à ce contexte.

Parmi les méthodes de threat modeling les plus utilisées dans la recherche de vulnérabilités logicielles nous pouvons citer :

- STRIDE, et plus particulièrement les variantes STRIDE-per-element ou STRIDE-per-interaction.
- DREAD.
- RTMP (Rapid Threat Model Prototyping).

La gestion des risques liés à des vulnérabilités ou des faiblesses de l'application ne fait pas partie des objectifs, mais s'alimentera des sorties des exercices de threat modeling.

4.2 PREREQUIS

La pratique du threat modeling nécessite un certain niveau de maturité au sein de l'organisation, et devrait être menée en collaboration entre les équipes AppSec et produit.

Dans ce modèle d'organisation, le niveau 1 de maturité du threat modeling ne peut dans la plupart des cas être entrepris qu'à partir du niveau 2 de maturité globale (« Bon niveau de Sécurité Applicative »), lorsque la responsabilité des activités sécurité commence à être partagée avec les équipes produit.

- Posséder une bonne connaissance du secteur, en particulier des attaques qui le ciblent et les modus operandi des groupes d'attaquants.
- Un modèle de security champions doit être en place.

4.3 IMPLÉMENTATION

NIVEAU 1

Thématiques	Activités	Livrables
Acteurs du threat modeling	Les ateliers de threat modeling sont menés par les équipes sécurité applicative. Les architectes applicatifs et les security champions sont des participants obligatoires.	Threat models pour l'application
Formation	Utilisation de formations générales disponibles sur le marché. Utilisation d'un guide interne de sensibilisation pour les populations qui ne sont pas ciblées directement par la formation.	Portail de formation accessibles a minima aux équipes sécurité et aux architectes applicatifs. Guide de sensibilisation au threat modeling.
Population formée	Les équipes sécurité sont formées à un niveau certifiant. Les architectes applicatifs et les security champions sont sensibilisés.	100% des équipes sécurité applicative formées et certifiées. 100% des architectes applicatifs et des security champions des applications les plus critiques sensibilisés.
Intégration SSDLC	Le SSDLC comporte une activité de threat modeling recommandée lors de la phase de design pour les applications les plus critiques.	Document SSDLC.
Méthodologie	Utilisation d'une méthode générique telle-quelle lors des ateliers.	Méthodologie générique Guide méthodologique
Ressources	Pas de mutualisation de ressource à ce stade.	N/A
Outillage	Utilisation de tableau blanc lors des ateliers, et d'outils génériques pour réaliser les diagrammes et suivre les menaces identifiées.	N/A

NIVEAU 2

Thématiques	Activités	Livrables
Acteurs du threat modeling	Les ateliers de threat modeling sont menés par les équipes produits (architectes applicatifs et security champions), les équipes sécurité applicatives sont présentes pour guider / réorienter / conseiller, mais ne sont pas leaders dans les ateliers.	Threat models pour l'application
Formation	Définition de parcours de formation adaptés à la culture d'entreprise avec du contenu basé sur un retour d'expérience des formations passées.	Parcours de formation organisé en trois niveaux : introduction, intermédiaire et avancé.
Population formée	Les équipes sécurité sont formées à un niveau certifiant. Les architectes applicatifs et les security champions intervenant sur les applications les plus critiques sont formés au premier niveau. Les architectes applicatifs et les security champions des autres applications, ainsi que tous les développeurs, sont sensibilisés.	100% des équipes sécurité applicative formées et certifiées. 100% des architectes applicatifs et des security champions des applications les plus critiques formés au premier niveau. 100% des architectes applicatifs et des security champions des autres applications sensibilisés. 100% des développeurs sensibilisés.
Intégration SSDLC	Le SSDLC comporte une activité de threat modeling obligatoire lors de la phase de design des applications les plus critiques, et recommandée pour les autres.	Dccument SSDLC.
Méthodologie	Utilisation d'une méthodologie adaptée à la culture de l'organisation, basée sur une ou plusieurs méthodologies génériques du marché affinées sur la base d'exercices passés.	Documentation de la méthogologie Fiches pratiques Threat models de référence réalisés avec cette méthodologie
Ressources	Maintien de ressources communes mais faiblement communiquées ou faiblement maintenues à jour.	Catalogue d'acteurs de menaces ou de scénarios de risques commun, connu principalement des équipes sécurité
Outillage	Utilisation de tableau blanc lors des ateliers, et d'outils génériques pour réaliser les diagrammes et suivre les menaces identifiées.	Outil de threat modeling

< THREAT MODELING >

NIVEAU 3

Thématiques	Activités	Livrables
Acteurs du threat modeling	Les ateliers de threat modeling sont menés en totale autonomie par les équipes produits (architectes applicatifs, security champions). Les équipes sécurité applicative sont disponibles sur demande pour y participer mais n'y sont pas inclus d'une manière systématique. Les équipes sécurité applicative revoient les threat models en utilisant de l'échantillonnage, priorisé par criticité des applications.	Threat models pour l'application
Formation	N/A, identique au niveau 2	N/A
Population formée	Les équipes sécurité sont formées à un niveau certifiant. Les architectes applicatifs et les security champions intervenant sur les applications les plus critiques sont formés au deuxième niveau. Les architectes applicatifs et les security champions des autres applications, ainsi que tous les développeurs, sont formés au premier niveau.	100% des équipes sécurité applicative formées et certifiées. 100% des architectes applicatifs et des security champions des applications les plus critiques formés au deuxième niveau. 100% des architectes applicatifs et des security champions des autres applications formés au premier niveau. 100% des développeurs formés au premier niveau.
Intégration SSDLC	Le SSDLC comporte une activité de threat modeling obligatoire pour l'ensemble des applications.	Document SSDLC.
Méthodologie	Identique au niveau 2	Identique au niveau 2
Ressources	Mutualisation de ressources consommables directement par l'outil de threat modeling, avec propriétaires identifiés qui les maintiennent à jour.	Catalogue central de composants standards avec threat models validés et maintenus, pour intégration dans les projets locaux et composition avec les éléments applicatifs Catalogue central de menaces pertinents pour l'organisation, qui sont obligatoirement considérés lors de la création d'un nouveau modèle
Outillage	Utilisation d'un outil de threat modeling capable d'utiliser des modèles spécifiques au métier, stockés dans un catalogue central, de suivre les menaces générées sur leur cycle, et de s'interfacer avec les solution de gestion de risques. La solution supporte de plus le threat modeling itératif nativement.	Outil de threat modeling évolué

4.4 BONNES PRATIQUES

- Mettre une limite temporelle sur les ateliers, et sur les activités individuelles lors des ateliers.
- Ne pas rester bloqué si plus aucune menace n'est identifiée pour un actif ou une catégorie, et passer à la suite.
- Ne pas chercher à trouver à tout prix une menace pour un actif et/ou une catégorie afin de « cocher la case » ; passer à la suite.
- Les menaces doivent être pratiques et reposer sur des vulnérabilités, et un changement de design ou des mesures de sécurité spécifiques doivent pouvoir la bloquer.
- Pour faciliter la découverte de menaces, lister un acteur malveillant et ses objectifs.

4.5 ANTIPATTERNS

- Lister des vulnérabilités au lieu des menaces : les menaces permettent de trouver des vulnérabilités.
- Lister des menaces trop génériques pour être pertinentes / corrigeables (par exemple « un attaquant exploite une vulnérabilité pour… »).
- Considérer que faire un pentest, un scan SAST ou un scan SCA est une remédiation à une menace ; sélectionner ces contre-mesures indique une menace mal identifiée (voire une vulnérabilité).

5 DEVELOPPEMENT

5.1 OBJECTIF

Dans cette étape, l'objectif est d'intégrer les exigences de sécurité dans le processus de développement et d'implémentation du besoin métier pour l'application.

5.2 PREREQUIS

Pour mener à bien cette étape, certains points sont considérés comme des prérequis :

- Sensibiliser/former les développeurs à la sécurité (type OWASP Top 10) et aux architectures / outils / composants / frameworks existants.
- Liste des exigences de sécurité et mesures à implémenter.
- Mise en place d'une chaine CI/CD, d'un outil de gestion de code source et d'un logiciel de dépôt d'artefacts.
- Politique de sécurité des applications décrivant les activités de sécurité à mener dans le cadre du developpement.

5.3 IMPLÉMENTATION

NIVEAU 1

Thématiques	Activités	Livrables
Contrôles sécurité	Premier niveau d'analyse de sécurité : revues par les pairs, détection des secrets dans le code	Actions/tickets de remédiation
Pilotage	Intégration de la sécurité dans la méthodologie projet	Activités sécurité dans le backlog Définition des critères d'acceptance de la security story

NIVEAU 2

Thématiques	Activités	Livrables
Processus	Accompagnement des équipes projet par un référent sécurité	RACI des activités de sécurité
Contrôles sécurité	Intégration des outils de sécurité (SCA, SAST et DAST) Revue de configurations de base des outils afin de minimiser les faux positifs Mise en œuvre de barrières de sécurité en mode informatif	Rapports d'outils type SCA/SAST/DAST/IAST Priorisation des remédiations des vulnérabilités détectées.
Pilotage	Définition de KPI permettant le suivi de l'évolution des vulnérabilités et des défauts.	Liste des KPI

NIVEAU 3

Thématiques	Activités	Livrables
Processus	Définition des rôles/ RACI incluant les activités de sécurité : responsabilité des acteurs Dev et Ops dans l'implémentation des exigences de sécurité, dans la gestion et l'utilisation des outils sécurité dans la chaine logicielle	
Contrôles sécurité	Optimisation de la configuration des outils spécifiquement pour l'application Mise en œuvre de barrières de sécurité en mode bloquant	Configuration des outils et des barrières de sécurité
Pilotage	Centralisation des KPI Pilotage de la politique de formation en se basant sur les vulnérabilités les plus détectées	Tableau de bord du suivi des KPI

5.4 BONNES PRATIQUES

Pour faciliter l'implémentation de ces actions, certaines bonnes pratiques existent :

- Mise en place d'un modèle de Security champion au sein des équipes de développements de l'organisation.
- Intégration de l'outil de sécurité dans l'IDE pour que les vulnérabilités soient remontées au plus tôt aux développeurs.
- Création de guides de développement sécurisé (générique et/ou par langage).
- Mettre en œuvre un outil de SCA avant un outil de SAST car les enjeux de sécurité des composants tiers utilisés sont plus importants, comme représentant plus de 70% du livrable, que le code livré qui pourra être testé au travers d'autres outils ou par les tests d'intrusions.

5.5 ANTIPATTERNS

Ces pratiques sont contre-productives pour l'implémentation de la sécurité lors des développements :

- Avoir une politique de sécurité avec des exigences élevées sans avoir de solutions techniques ou de support pour aider les équipes de développement à implémenter ces exigences.
- Mettre à disposition des développeurs des outils ou systèmes avec des configurations par défaut non adaptées au niveau de sécurité.

6 INTEGRATION CONTINUE

6.1 OBJECTIF

Assurer l'intégration de la sécurité tout du long du cycle y compris lors de la construction des composants de l'application et avec la réalisation de tests en continu.

6.2 PREREQUIS

- Mise en place d'une chaine CI/CD, d'un outil de gestion de code source et d'un logiciel de dépôt d'artefacts;

6.3 IMPLEMENTATION

NIVEAU 1

Thématiques	Activités	Livrables
Processus	Intégration de contrôles de sécurité dans la chaine CI	Documentation des processus de la chaine CI
Dépendances logicielles	Génération de la Software Bill Of Materials (SBOM)	Fichiers SBOM
Outillage	Standardisation des outils composant la chaine CI	Documentation accessible et à jour
Secrets	Vérification manuelle de l'absence de secrets dans le depot de code source	Révocation des secrets détectés
Artefacts	Création d'un dépôt centralisé d'artefact	Solution de gestion d'artefacts

NIVEAU 2

Thématiques	Activités	Livrables
Processus	Automatisation de la chaîne de build complète	Documentation de la chaine automatisée
Dépendances logicielles	Identification des vulnérabilités contenus dans les artefacts (SCA) Priorisation de la correction des vulnérabilités basée sur la criticité des CVEs	Rapports contenant les éventuelles CVEs Actions de corrections Suivi de corrections des vulnérabilités les plus critiques
Outillage	Intégration d'un outil SCA et de scan de secrets a minima	Documentation d'utilisation et d'intégration
Secrets	Automatisation de l'absence de secrets dans le dépôt de code source et d'artefact	Résultats de scan de secrets dans les sources et les configurations
Artefacts	Génération de checksums pour vérification de l'intégrité	Artefact + checksum associés

NIVEAU 3

Thématiques	Activités	Livrables
Processus	Contrôles de sécurité obligatoires définis et interruption du process en cas de non-conformité	Documentation et templates standards par niveau de sécurité de l'application
Dépendances logicielles	Implémentation d'un registre de dépendances autorisées Contextualisation de la criticité des vulnérabilités détectées	Méthodologie de l'évaluation de la criticité d'une vulnérabilité
Outillage	Analyse régulière dans le depot centralisé d'artefact et notification du propriétaire du projet associé	Documentation d'utilisation et d'intégration
Secrets	Mise en place d'un outil de gestion de secrets	Documentation d'utilisation et d'intégration Dossier d'architecture du gestionnaire de secrets Politique de gestion des secrets et des certificats
Artefacts	Signature de tous les artefacts	Artefact signé avant dépôt

6.4 BONNES PRATIQUES

La mise en place d'une solution permettant de générer des fichiers SBOM applicatifs peut permettre d'initier un référentiel unique des librairies et licences tierces utilisées afin d'identifier rapidement l'impact d'une zero-day sur une librairie déterminée (i.e.: log4j pour la vulnérabilité Log4Shell).

La solution dependency-track par exemple permet de mettre en place une analyse centralisée.

BONNES PRATIQUES POUR LES SBOM

Afin de générer des SBOM de qualité, il est néccessaire de s'assurer que l'outil utilisé pour la génération couvre les langages utilisés par l'application.

Il est aussi recommandé de créer une SBOM à chaque modification des composants de l'application.

Pour avoir une meilleure couverture des dépendances logiciels présentes dans l'application, il est recommandé de fournir les « lock files » (package-lock.json, yarn.lock, ect…) car ces derniers contiennent des informations cernant les packets installés.

BONNES PRATIQUES SUR L'UTILISATION DES ENVIRONNEMENTS

L'utilisation d'environnement spécifique à l'intégration continue (CI) séparé des environnements de déploiement (CD) est primordiale afin d'empêcher une compromission des artefacts de déploiement ou des secrets de déploiement.

6.5 ANTIPATTERNS

L'utilisation par défaut de la dernière version d'un composant logiciel (tag-latest) n'est pas forcément une bonne pratique car peut induire des erreurs en cas de montée de version de la part de l'éditeur, de la communauté, du développeur. Cette pratique peut impliquer une adaptation dans le code/la configuration qui ne sera visible que lors d'un prochain build qui sera en échec.

7 LIVRAISON ET DEPLOIEMENT CONTINUS

7.1 OBJECTIF

Dans cette étape, les composants de l'application développée doivent être livrés et déployés de manière sécurisée en production.

7.2 PREREQUIS

- Définition des exigences de sécurité au niveau de l'infrastructure, de leurs configurations et des seuils d'acceptabilité des risques pour le déploiement en production.
- Mise en place d'une chaine CD avec les outils permettant le déploiement en production.
- Définition d'un standard listant les différents environnements applicatifs.

7.3 IMPLÉMENTATION

Définition des processus et des critères d'acceptation type de barrières de sécurité.

NIVEAU 1

Thématiques	Activités	Livrables
Processus	Test d'intrusion avant la 1ere mise en production selon la criticité des applications Barrières de sécurité obligatoires et non bloquantes pour le CD	Implémentation des barrières dans l'outils CI/CD
Outillage	Mise en place d'un outil déploiement automatisé	
Artefacts	Mécanisme manuel de promotion des artefacts en fonction de l'environnement de déploiement	Processus et documentation associés

< LIVRAISON ET DEPLOIEMENT CONTINUS >

NIVEAU 2

Thématiques	Activités	Livrables
Processus	Barrières de sécurité obligatoires et interruption du process en cas de non-conformité pour les applications critiques	Implémentation des barrières dans l'outils CI/CD et définition de dérogation
Outillage	Outil de vérification de la conformité du code IaC	Rapport des non-conformités
Artefacts	Mécanisme automatique de promotion des artefacts en fonction de l'environnement de déploiement	Implémentation de la promotion dans le dépôt des artefacts

NIVEAU 3

Thématiques	Activités	Livrables
Processus	Barrières de sécurité obligatoires et interruption du process en cas de non-conformité pour toutes les applications	Implémentation des barrières dans l'outils CI/CD et définition de dérogation
Outillage	Génération automatique de l'artefact éligible à la production	
Artefacts	Vérification de la signature des artefacts	

8 MAINTIEN EN CONDITIONS DE SÉCURITÉ

8.1 OBECJTIF

L'objectif est de maintenir les besoins de sécurité en production des applications. Cela peut aussi bien concerner le maintien à jour des composants et logiciels et le renforcement des environnements de production, la surveillance applicative, la gestion des incidents et la protection des données.

8.2. PRÉREQUIS

Pour réaliser le maintien en conditions de sécurité il est nécessaire d'avoir un inventaire complet des applications à surveiller, de leurs infrastructures et des composants utilisés par celles-ci.

8.3 IMPLÉMENTATION

NIVEAU 1

Thématiques	Activités	Livrables
Durcissement	Application des recommandations de sécurité sur les composants clés selon disponibilité WAF	Liste des composants clés pour chaque stack technique et statut
Mise à jour des composants	Mise à jour de l'infrastructure et des composants applicatifs selon les ressources disponibles	Composants mis à jour
Protection des données	Identification des données stockées et traitées	Verification de la conformité legale des données de productions
Gestion du legacy et décommissionnement	Identifier les applications, serveurs composants, etc. non utilisés, en fin de vie, plus maintenus	Etude d'impact et traitement de suppression ou de mise à jour si possible
Contrôles de sécurité	Tests d'intrusion	Rapports, plan d'actions et planification de remédiations
Défense en profondeur	WAF en configuration standard	Logos

NIVEAU 2

Thématiques	Activités	Livrables
Durcissement	Définition de guides standards et de leurs responsables	Standards et RACI
Mise à jour des composants	Mise à jour régulière de l'infrastructure et des composants applicatifs	Composants mis à jour
Protection des données	Définition d'un catalogue de données incluant les types, les niveaux de sensibilité, les types de traitement et les localisations de stockage	Standard de classification des données
Gestion du legacy et décommissionnement	Définition d'un processus de décommissionnement	Document standard décrivant le processus de suppression, les versions supportées et les mises à jour
Contrôles de sécurité	Scans de vulnérabilités infrastructure et applicatif type SCA, DAST, fuzzing	Priorisation et planification des traitements (remédiation)
Défense en profondeur	WAF configuré par application après mode d'apprentissage	Configuration et logs/incidents

NIVEAU 3

Thématiques	Activités	Livrables
Durcissement	Vérification régulière et automatisée du durcissement et mise à jour des standards	Rapports de conformité et statuts Revue annuelle
Mise à jour des composants	Suivi opérationnel et contrôle de l'ensemble des composants et leurs versions	Mise en place d'une solution ou de tableaux de suivi avec des SLAs pour l'ensemble du portfolio applicatif Traitement actif des non-conformités
Protection des données	Définition d'une politique de protection des données et suivi opérationnel de sa conformité Installation d'outils permettant la prévention des pertes de données (DLP), le suivi des contrôles d'accès et la détection des comportements anormaux	Politique de protection des données de l'entreprise Rapports de conformité et planification des traitements Organisation (DPO, référent données/conformité/qualité)
Gestion du legacy et décommissionnement	Suivi régulier de l'état des cycles de vie et statut de support de tous les logiciels et composants d'infrastructure	Plan de support et revue annuelle a minima
Contrôles de sécurité	Mise en place et campagnes de Bug bounty	Processus, politique de divulgation et partenariat plateforme Priorisation et planification des traitements (remédiation)
Défense en profondeur	Runtime Application Self Protection (RASP)	Alertes de sécurité

8.4 BONNES PRATIQUES

- WAF préconfiguré *as code* au déploiement.

8.5 ANTIPATTERNS

Parmi les choses à éviter, la mise en place d'un DAST sans mettre en place un WAF, permettant de sécuriser l'exposition applicative, est contre-productive. L'implémentation d'un WAF sans prévoir un monitoring des alertes remontées est elle aussi improductive.

9 GESTION DES VULNÉRABILITÉS

9.1 OBJECTIFS

Définir un process global de gestion de vulnérabilité dans le cycle de développement logiciel.

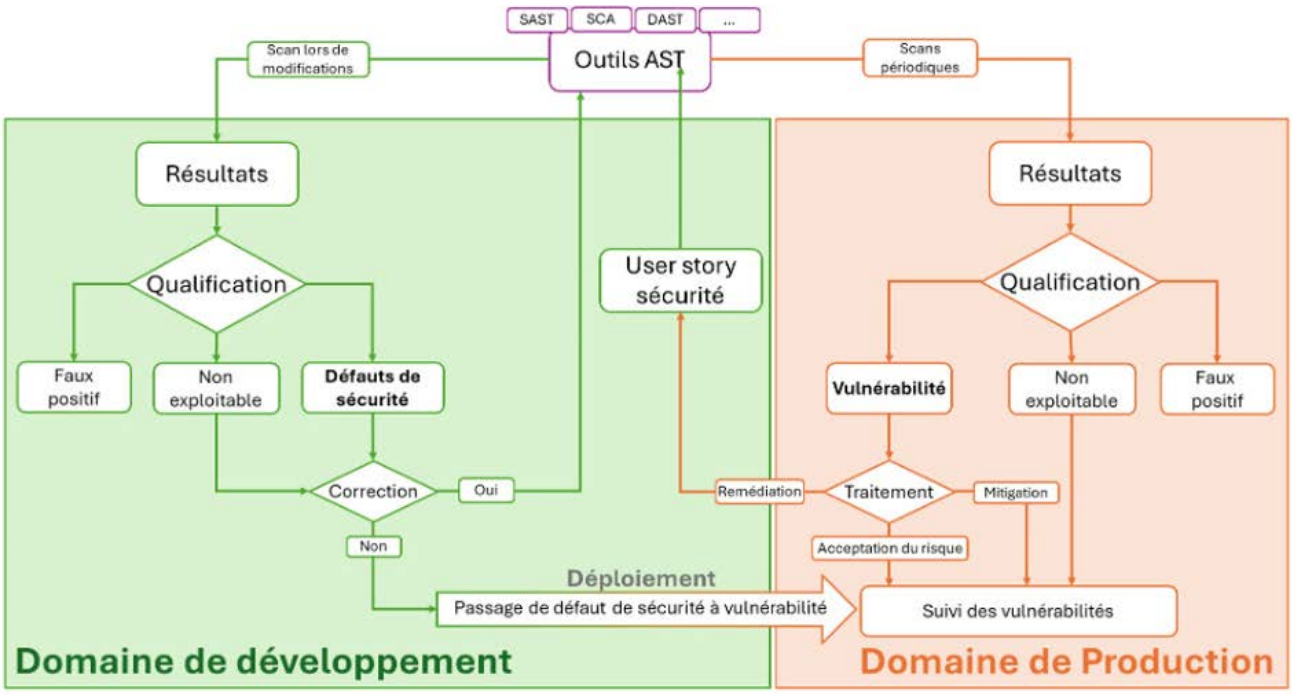
Deux types de failles de sécurité :

- Vulnérabilité : faille pour une application en production
- Défaut de sécurité : faille détectée dans le cycle de développement d'une application

Trois types de process de gestion de failles de sécurité :

- Process de gestion de la Dette de sécurité, l'ensemble des vulnérabilités pour une application déjà en production.
- Process de gestion des nouveaux défauts de sécurité détectés pendant le cycle de vie Agile et des vulnérabilités.
- Process de gestion des vulnérabilités détectées après la mise en production avec des analyses continues de la production.

L'équipe de développement ne doit gérer que les failles du code de son application et de ses dépendances et non les failles de composants livrés par d'autres équipes.



9.2 ACTIVITÉS

Détection des failles de sécurité

- Outils SCA, SAST, DAST, Test d'intrusion, Bug bounty, Veille securitaire.
- Stratégie d'orchestration des outils : au moment de la mise en place de la stratégie de gestion des vulnérabilités, lors du cycle de vie Agile et de façon continue sur la production.

Qualification et priorisation

- Qualification : Définition du statut d'une remontée en défaut de sécurité, en vulnérabilité ou en faux positif et réévaluation de sa criticité en fonction de son exploitabilité et du contexte applicatif.
- Priorisation : Définition des priorités, à partir d'une stratégie de remédiations et de l'organisation de l'équipe.
- Critères de priorisation : criticité de la vulnérabilité (CVSS), exposition de l'application Internet ou interne, profil de sécurité DICT de l'application

Traitement

- Remédiation : correction du défaut.
- Mitigation : atténuation par la mise en place de mesures de sécurité supplémentaires.
- Acceptation du risque en cas de non-correction de la vulnérabilité.

Barrière de sécurité

- Barrière bloquante : déploiement bloqué si détection de vulnérabilités majeures.
- Barrière non bloquante : déploiement de l'application avec génération d'un rapport détaillant les vulnérabilités.

Gestion des exceptions

- Définition du SLA pour la remédiation : temps maximal pour la correction.
- Non affichage dans la barrière des vulnérabilités dont le risque a été accepté et qui respecte le SLA.

9.3 BONNES PRATIQUES

- Garder une trace des justifications des exceptions et du porteur du risque accepté par le métier
- Reintroduire dans le sprint les vulnérabilités afin d'y remédier avant l'échéance de leur SLA
- Réaliser un audit par echantillonage des faux positifs
- Mettre en place des tests permettant de s'assurer de la qualification dans le temps des vulnérabilités non exploitables.

9.4 ANTIPATTERNS

- Barrière bloquante sans processus complet de gestion des exceptions.

< ANNEXES >

< RÉFÉRENCES >

[1] ANDREESEN Marc: Why Software Is Eating The World. 2011. Disponible à l'adresse : https://www.wsj.com/articles/SB10001424053111903480904576512250915629460

[2] Google : What is Site Reliability Engineering (SRE)? Disponible à l'adresse : https://sre.google/

[3] WILSON Glenn. DevSecOps : A leader's guide to producing secure software without compromising flow, feedback and continuous improvement, 2020

[4] Campus Cyber. Cartographie des Technologies, 2025.

[5] VDOO. The Practical Guide for Understanding & Preventing Software Supply Chain Attacks. 2021. Disponible à l'adresse : https://www.energy.gov/sites/default/files/2021-06/Shane%20DeLair%20Vdoo-A1.pdf

[6] SYNOPSYS. Open Source Security And Risk Analysis Report. 2023. Disponible à l'adresse : https://www.synopsys.com/content/dam/synopsys/sig-assets/reports/rep-ossra-2023.pdf

[7] ANSSI. La directive NIS2. Disponible à l'adresse : https://cyber.gouv.fr/la-directive-nis-2

[8] OWASP. Software Assurance Maturity Model. Disponible à l'adresse : https://owasp.org/www-project-samm/

MODÈLE DE MATURITÉ

OWASP SAMM v2
https://owaspsamm.org/assessment/

OWASP DSOMM
https://dsomm.owasp.org/

CIS
https://www.cisecurity.org/controls/application-software-security

Datadog
https://www.datadoghq.com/blog/devsecops-maturity-model-self-assessment/

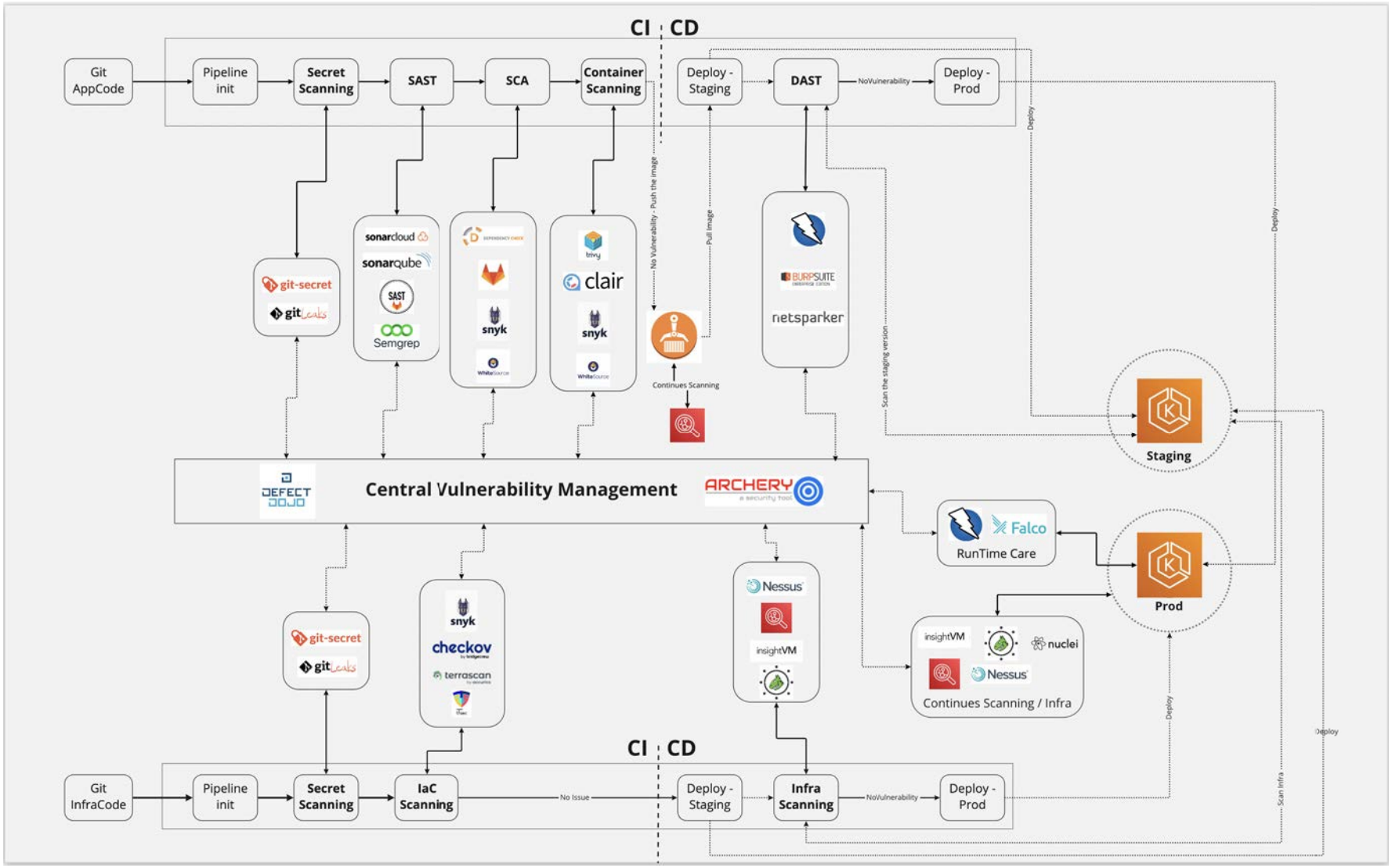
Gitlab
https://learn.gitlab.com/c/devsecops-assessment?x=u5RjB_

BSIMM
https://www.synopsys.com/software-integrity/software-security-services/bsimm-maturity-model.html

Documentation SSDLC
https://owasp.org/www-pdf-archive/Jim_Manico_(Hamburg)_-_Securiing_the_SDLC.pdf

< RÉFÉRENCES >

MODÈLE DE PIPELINE



https://owasp.org/www-project-devsecops-guideline/latest/

< RÉFÉRENCES >

MODÈLE DE PIPELINE

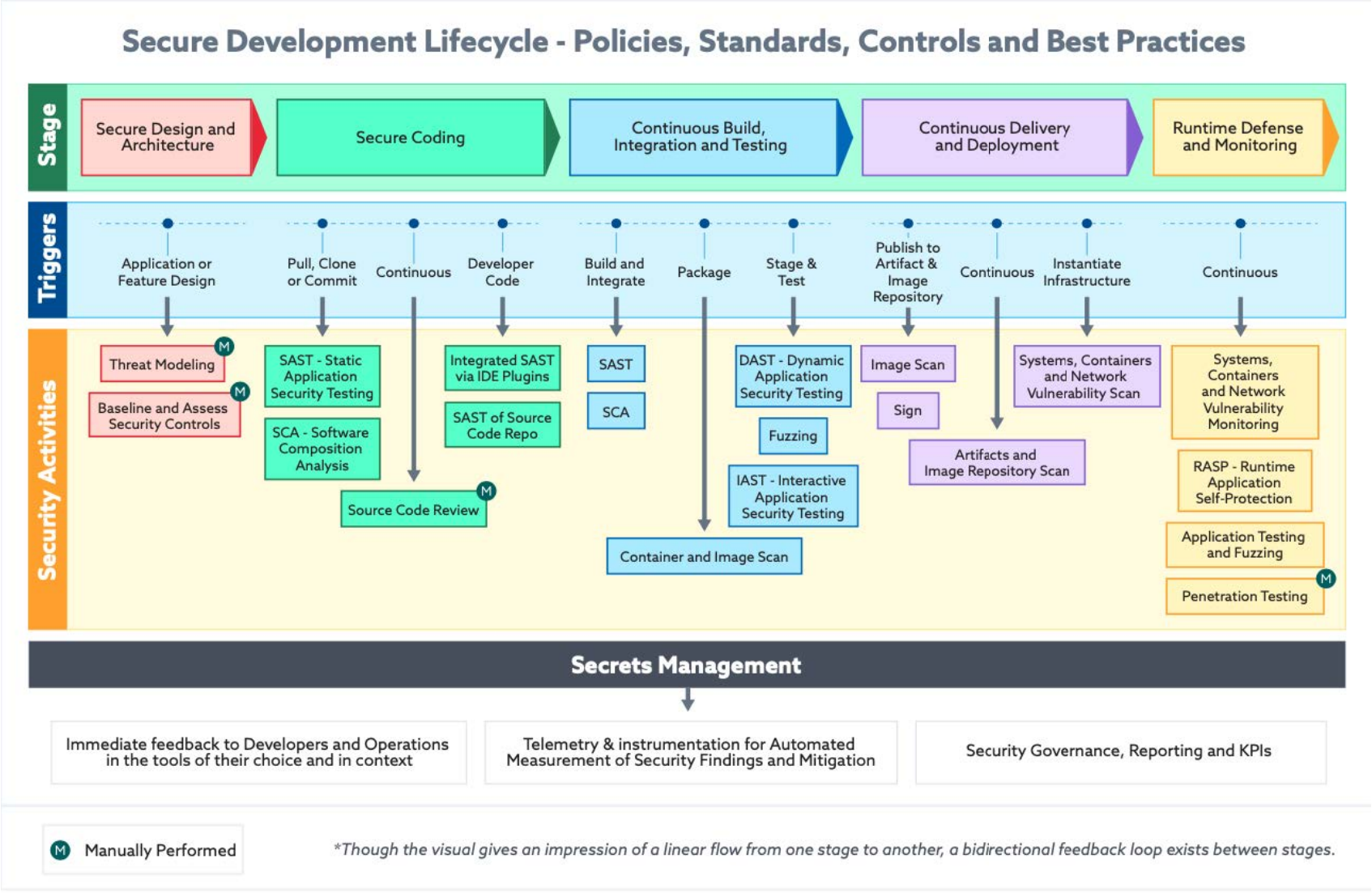
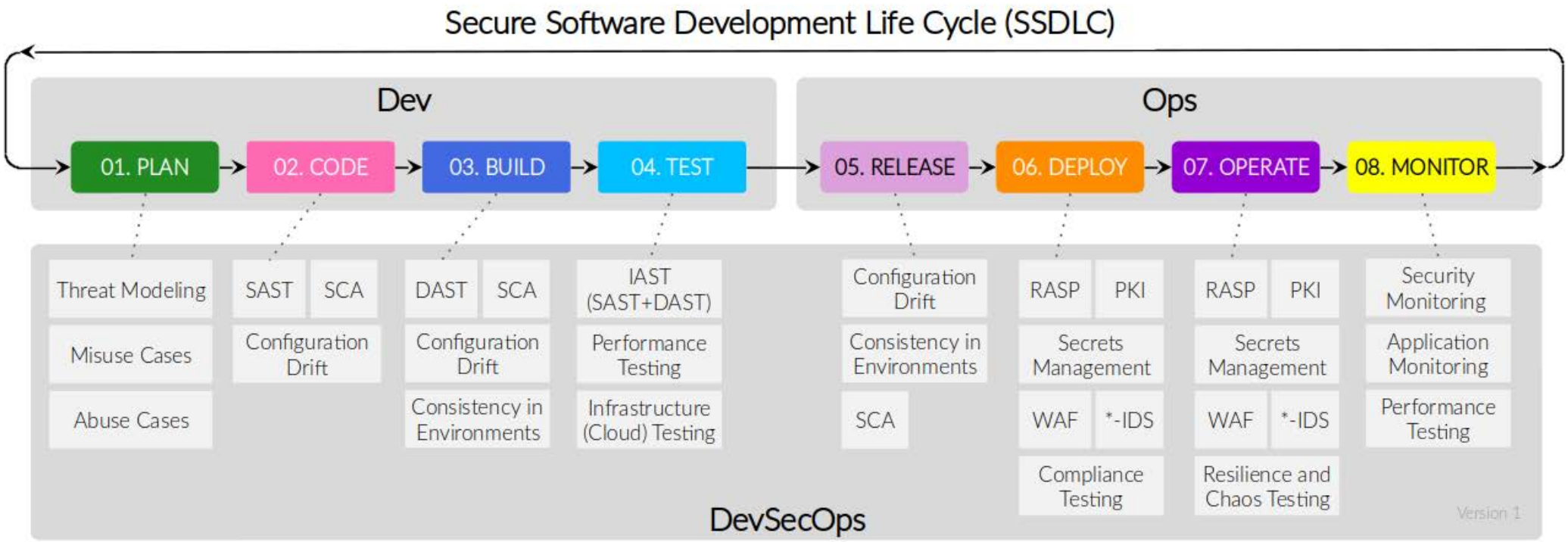


Figure 1: The CSA DevSecOps Delivery Pipeline

https://devsecops.pagerduty.com/secure_sdlc/

< RÉFÉRENCES >

MODÈLE DE PIPELINE



https://holisticsecurity.io/2020/02/10/security-along-the-sdlc-for-cloud-native-apps/

< GLOSSAIRE >

DAST DAST signifie Dynamic Application Security Testing en anglais. Contrairement à un outil de SAST, un outil de DAST évalue l'application depuis l'extérieur en cours d'exécution, en simulant des attaques pour identifier les vulnérabilités de sécurité.

IAST IAST signifie Interactive Application Security Testing en anglais. C'est une méthode de test de sécurité qui combine l'approche SAST, et l'approche DAST pour identifier les vulnérabilités pendant que l'application est exécutée. Cela se traduit souvent par un agent qui surveille passivement le trafic pour déterminer si le code source d'une application est susceptible d'être exploité.

RASP RASP signifie Runtime Application Self-Protection en anglais. C'est une technologie de sécurité qui s'intègre à une application ou à son environnement d'exécution pour détecter et bloquer les menaces en temps réel. Contrairement à d'autres technologies, elle est utilisée dans un environnement de production.

SAST SAST signifie Static Application Security Testing en anglais. Les outils de SAST analysent le code source d'une application, ou ses composants binaires, pour identifier des vulnérabilités de sécurité. Un test de type SAST est effectué sans exécuter le programme, c'est pourquoi il est qualifié de test statique.

SBOM SBOM signifie Software Bill of Materials en anglais. Une SBOM est une liste exhaustive des composants utilisés dans la création d'un produit logiciel, y compris les bibliothèques tierces, les modules, les frameworks et les packages. La SBOM doit identifier de manière unique chaque composant et chaque version de chaque composant logiciel utilisé.

SCA SCA signifie Software Composition Analysis en anglais. C'est une pratique qui consiste à analyser automatiquement les composants logiciels tiers, tels que les bibliothèques open source et les frameworks, pour identifier et gérer les risques de sécurité, et les risques de conflits entre licences open-source. Le SCA s'appuie sur une SBOM afin de pouvoir effectuer son analyse.

SDLC SDLC signifie Software Development Life Cycle en anglais, ou cycle de vie du développement logiciel en français

SLA SLA signifie Service Level Agreement en anglais ou accord de niveau de service en français. C'est un contrat ou une partie d'un contrat entre un fournisseur de services et son client qui définit le niveau de service attendu.

Vulnérabilité Défaut de sécurité livré en production (exploitable ou non)

WAF WAF signifie Web Application Firewall en anglais. C'est un dispositif de sécurité spécifiquement conçu pour surveiller, filtrer et bloquer le trafic HTTP entrant et sortant d'une application web.

< REMERCIEMENTS >

Ce livrable a été produit dans le cadre du groupe de travail Cybersécurité Agile - Stream Gouvernance.

Contributeurs et rédacteurs (par ordre alphabétique) :

Benjamin CHOBERT, BNPP
Flavien DUMUR
Vincent FOURESTIER, Banque de France
Erwan NEDELLEC, Orange
Gregory OESTREICHER, BNPP
Lucas RANGEARD, CapGemini

< Studio des Communs >

CAMPUS CYBER

POUR EN SAVOIR PLUS : **WIKI.CAMPUSCYBER.FR**
ADRESSE MAIL DE CONTACT : **COMMUNAUTES@CAMPUSCYBER.FR**
5 - 7 RUE BELLINI 92800, PUTEAUX

Ce projet a été financé par le gouvernement dans le cadre du Programme d'investissements d'avenir.