Le Développement Agile pour les Responsables de la Sécurité

www.quotium.com

Agile Software Security

Les concepts Agile et DevOps accélèrent la collaboration entre équipes et la mise en production de nouvelles fonctionnalités produit.

Néanmoins avec un développement Agile dynamique et épuré, il est difficile d'intégrer des activités nécessitant du temps de configuration, d'analyse et d'intervention de prestataires externes. Les outils et services autour de la sécurité applicative évoluent moins vite que les méthodes de développement.

Dans ce document, nous analysons comment une organisation peut développer des logiciels en adoptant une méthode Agile et obtenir un niveau de sécurité élevé.

Le Développement Agile pour les Responsables de la Sécurité

AGILE SOFTWARE SECURITY

Table des matières

INTRODUCTION

Comment vos données sensibles sont-elles gérées par vos applications ?

La plupart des attaques se focalisent sur le vol de données confidentielles, meilleur potentiel de revenus pour les attaquants. Les applications gèrent de plus en plus de données métiers et sont donc de ce fait une cible de choix pour les hackers.

Afin de suivre les demandes croissantes des utilisateurs, les entreprises se dirigent vers des méthodes de développements itératifs, d'intégration continue et d'automatisation des processus. La méthode traditionnelle « Waterfall » a fait ses preuves, mais elle comporte tellement d'étapes que le processus ne parvient pas à suivre l'évolution des besoins clients.

Agile est une méthode qui accélère la mise en production de nouvelles fonctionnalités et la collaboration entre équipes. Néanmoins avec un développement dynamique et épuré, il est difficile d'intégrer des activités nécessitant du temps de configuration, d'analyse et d'intervention de prestataires externes. Les outils et services autour de la sécurité applicative évoluent moins vite que les méthodes de développement.

Comment s'assurer que le code mis en ligne sous la pression des métiers soit systématiquement sécurisé ?

Nous analyserons ici le développement Agile par le prisme de la sécurité applicative et élaborerons les critères d'intégration de la sécurité dans ces nouvelles méthodes de développement.

La sécurité des applications en chiffres

- Les applications concentrent 90% des vulnérabilités. Pourtant 80 % des budgets sécurité sont encore consacrés aux infrastructures
- Une application contient 13 failles en moyenne
- 85% des applications utilisent des composants open source, 6 projets sur 10 ne suivent pas les mises à jour de sécurité de ces composants
- Durant les 6 premiers mois de 2014, 1331 incidents majeurs ont été rapportés dans le monde concernant le vol de plus de 502 millions de données personnelles

PRESENTATION DES PRINCIPES AGILE

Les méthodes Agile partent du principe que spécifier et planifier en détail l'intégralité d'un produit avant de le développer (approche prédictive) est contre-productif.

Le concept Agile pur est décrit dans le Manifeste Agile définit par les 4 affirmations suivantes :

- **Les individus et leurs interactions** plus que les processus et les outils
- **Des logiciels opérationnels** plus qu'une documentation exhaustive
- **La collaboration avec les clients** plus que la négociation contractuelle
- **L'adaptation au changement** plus que le suivi d'un plan

« Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée. »

Dans un cycle de développement Agile ce sont les scénarios utilisateurs (« User stories ») qui dictent les développements : le client-utilisateur raconte ce qu'il souhaite faire – de son point de vue métier. L'équipe sélectionne ensuite la liste des fonctionnalités réalisables dans une portion de temps courte appelée « itération ».

Chaque itération inclut des travaux de conception, de spécification, de développement et de test. A la fin de chacune de ces itérations, le produit partiel mais utilisable est montré au client. Si le client a priorisé avec soin son besoin, il peut saisir l'opportunité d'accélérer le « time to market » s'il estime que le produit en l'état (partiel) peut aller en production.

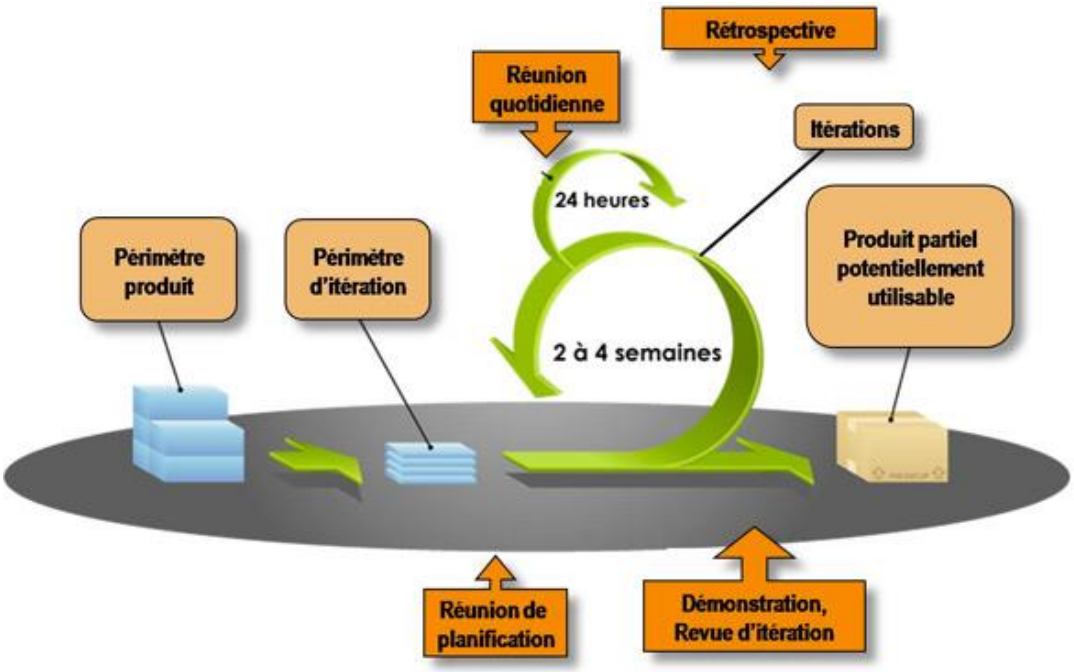

Fig1 Du scénario utilisateur au produit partiel – La méthode Scrum

Différentes méthodologies de mise en œuvre ont vu le jour autour du concept Agile: Scrum, Extreme Programming (XP), Rational Unified Process (RUP), Feature Driven Development, Kanban etc... Leur définition est donnée dans le glossaire en annexe.

Même si la plupart des entreprises n'adoptent pas une méthode Agile sous sa forme la plus pure, nombreuses sont celles qui appliquent une méthode qui repose sur des principes Agile.

Ces principes sont résumés ci-dessous.

1) Développement Lean

La méthode Agile applique les principes du Lean manufacturing au processus de développement logiciels. La coopération quotidienne entre les développeurs et les parties prenantes permet de réduire les gaspillages, de donner de l'autonomie à l'équipe et de livrer le projet aussi rapidement que possible et au meilleur coût. La mesure essentielle de la livraison est un logiciel opérationnel qui fonctionne comme prévu.

2) La responsabilité est confiée aux développeurs

Les équipes de développement se voient confier l'entière responsabilité du produit et bénéficient d'une confiance totale. La meilleure méthode de communication repose sur un échange direct entre toutes les parties prenantes. Les différents acteurs travaillant souvent dans un même espace.

3) Exigences orientées client : User Stories

Les projets sont réalisés en collaboration avec le client qui détermine les exigences et le périmètre du projet. Un récit utilisateur ou « user story » est un énoncé court et rédigé de manière simple qui explique ce qu'un utilisateur doit faire pour chaque fonctionnalité du projet. La liste des cas utilisateurs représente ce que l'on appelle un « backlog », liste des fonctionnalités produit à développer.

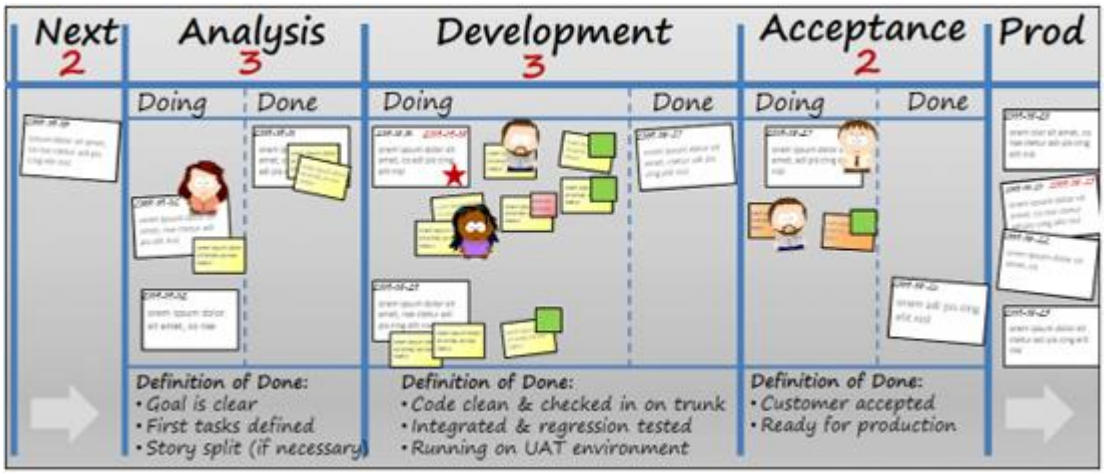

Fig 2 Exemple de tableau de bord Agile

4) Les exigences évoluent au cours du processus

Les équipes Agile savent que les exigences vont changer tout au long du processus en réponse au retour client sur les précédentes itérations. Ceci implique le gaspillage des premiers investissements.

Les exigences sont simplement définies en amont pour identifier le périmètre du projet. Les exigences définitives seront déterminées en cours de processus lorsque l'équipe toute entière aura parfaitement compris ce que doivent être les résultats finaux.

5) Automatisation des tests : Tests permanents

La mise à jour constante du code permet d'effectuer des modifications au fil de l'eau, en adaptant le produit final au besoin. Le cycle de vie Agile augment le nombre de release et fait donc appel à un besoin de tests continus remplaçant les différentes étapes en cascades.

Un processus Agile a besoin de l'automatisation d'une part importante de test afin :

- d'avoir un retour rapide sur la qualité de l'itération
- l'assurance de l'absence de bug de régression
- d'éliminer les tâches répétitives
- permettre à l'équipe de se focaliser sur la validation de fonctionnalités spécifiques.

La méthode XP (Extreme Programing) par exemple consiste à fusionner le code de chaque développeur avec la source plusieurs fois par jour afin de vérifier que le code de tous les développeurs s'intègre correctement. Un build plus complet est programmé la nuit pour vérifier que le produit fonctionne. Les taches effectuées dans ce build complet sont par exemple :

- Validation du code;
- Exécution des tests unitaires;
- Construction des releases;
- Déploiement de l'application sur l'environnement d'intégration;
- Exécution automatique de tests fonctionnels (script sélénium), de non régression, d'intégration, de performance, de sécurité etc….
- Génération d'un rapport de tests. Les bugs rencontrés sont insérés dans les outils de gestion de bugs

L'équipe Agile aura donc le matin suivant un vue globale de la qualité du produit en cours.

6) Un processus d'auto-amélioration

Les équipes doivent régulièrement s'arrêter pour réfléchir sur les problèmes rencontrés dans le processus et trouver les moyens de les résoudre.

Dans la méthodologie Scrum par exemple, une rétrospective est faite à la fin de chaque itération. L'équipe analyse ce qui s'est passé durant ce sprint, afin de s'améliorer pour le prochain. L'adaptation et la réactivité de l'équipe de développement est facilitée par son auto-organisation.

GARANTIR LA SECURITE DES APPLICATIONS DANS AGILE

Les outils de tests de sécurité traditionnels (analyseur de code, scanneur de vulnérabilités) ainsi que les sessions de pentesting sont des activités séquentielles consommatrices de temps qui nécessitent une expertise sécurité et outil :

- Pour chaque rapport que vous allez obtenir, les faux positifs et faux négatifs devrons être traité et un processus d'analyse effectué pour corréler les résultats.
- Peu de personnes ont la capacité de savoir ce que fait l'application ou ont la capacité de comprendre la faille pour déterminer où se situe le problème dans le code.
- Problèmes liés aux tierces parties : vous ne disposez pas nécessairement de l'ensemble du code source
- Les résultats sont très centrés sur le code, pas sur l'application elle-même ou le flux métier impacté. Si l'on vous informe d'une faille par injection SQL au sein d'une fonction particulière dans le code, qui saura où elle se manifeste dans l'interface ? Ce code peut également être redondant.

Cela signifie que les outils qui mettent du temps à être exploités s'avèrent inefficaces dans ces environnements, tout comme les outils qui requièrent une interprétation manuelle des résultats. Ils ralentissent la tâche des concepteurs, en les contraignant à travailler en dehors du cadre Agile.

Ces outils sont donc souvent déployés dans un processus d'audit parallèle au développement mais **comment être sûr que chaque version de code livré en production de façon itérative soit sécurisée?**

Reprenons les caractéristiques du concept Agile afin de déterminer les principes clefs nécessaires à cette intégration.

1) Définir des exigences claires en terme de sécurité

Dans les équipes Agile, les connaissances sont partagées et ne relèvent pas de la responsabilité d'une seule personne ou équipe. La qualité et la sécurité du code font partie intégrante du travail de groupe

Très souvent, les développeurs ont le sentiment que les procédures de sécurité sont redondantes et ne servent qu'à alourdir leur charge de travail, notamment lorsque la validation de la sécurité n'est effectuée qu'à la toute fin du projet.

Des exigences claires doivent donc être définies dès le début. L'équipe doit comprendre le niveau de sécurité souhaité, avoir conscience de l'importance d'atteindre ce niveau, ainsi que les tests qui doivent être menés pour atteindre ces résultats.

La première étape pour les développeurs est de répondre aux questions suivantes afin de bien définir les exigences :

- Quelles normes de sécurité doivent être suivies et validées par l'équipe? (Il peut s'agir de normes du secteur telles qu'OWASP Top 10, PCI-DSS ou des exigences internes de l'entreprise)
- L'utilisation d'API ou de librairies sécurisés spécifiques est-elles préconisée ?(ESAPI..)
- Quels sont les points à traiter en priorité dans le cadre de l'élaboration de tests de sécurité?
- Ces tests remplacent-ils les tests périodiques de pénétration et les audits de sécurité ou sont-ils associés à ces méthodes de test ?
- Combien de fois les développeurs doivent-ils tester la sécurité ? Qui est chargé de réaliser ces tests ?

Après avoir répondu à ces questions, l'équipe de développement sera à même de mieux comprendre les enjeux de la sécurité applicative dans leur processus sous l'angle de la coopération, plutôt que sous le prisme de la suspicion. Il est important de fournir non seulement des exigences, mais aussi un soutien en vue de la réalisation de l'objectif, que ce soit par la formation ou l'utilisation de frameworks sécurisés spécifiques etc…

2) La sécurité applicative : un travail de groupe

Les équipes de développement se posent souvent les questions suivantes :

- Qui doit valider la sécurité ? Est-ce que chaque développeur peut utiliser son propre code de test unitaire ou est-il préférable de définir des cas de test spécifiques?
- Quel sont les critères de sécurité à rajouter dans la définition du « Done » (Etape fini)
- Quelle doit être la fréquence des tests de sécurité et doivent-ils être effectués sur chaque élément du code ou après l'intégration ?
- À qui les résultats doivent-ils être transmis ? Au développement ou à la sécurité ?
- Qui est en charge de la validation ?

Pour qu'un flux de sécurité soit Agile il doit suivre les habitudes et les interactions de l'équipe. L'objectif est de créer un processus qui intègre la sécurité applicative dans chacune des phases, tout en appliquant les principes Agile rapides et Lean.

3) Créer des récits de sécurité

Les équipes Agile définissent leur besoins sous forme de «user-stories». La sécurité applicative devrait donc être traitée de la même manière afin de s'intégrer dans les exigences de développement.

Les récits de sécurité utilisateur ou plutôt les cas d'utilisation abusifs de ces « user stories » permettent de prendre en compte en amont la sécurité dans la conception. Cette étape d'intégration dans le « Product Backlog » est primordiale et évite la dispersion, sachant que par exemple en Scrum les Développeurs n'ont pas le droit de travailler sur autre chose que leur « Backlog ».

La surface d'attaque est représentée par tous les éléments techniques du projet qui viennent en contact avec les utilisateurs ou l'extérieur. C'est effectivement le point d'entrée de la plupart des attaques. L'interaction avec les utilisateurs permet une meilleure

connaissance des processus métier impactés afin de pouvoir définir ces « cas abusifs » et anticiper des problèmes de sécurité que peuvent engendrer ces flux.

4) Travailler avec les processus existants

Les équipes de développement disposent de leurs propres processus qu'elles utilisent au cours du cycle de vie du projet. Comme presque tout ce qui a trait à Agile, ces processus sont censés être légers et épurés.

Le Manifeste Agile stipule : « les individus et les interactions par-delà les processus et les outils. » S'agissant de la sécurité, vous ne pouvez pas négliger entièrement les outils, mais vous devez être sûr que ***les outils que vous utilisez facilitent les interactions entre les membres de l'équipe*** en élaborant un langage commun compris par tous, et qu'ils n'entravent pas le processus.

Privilégiez les outils qui produisent des résultats dans un langage que les développeurs comprennent, en précisant l'emplacement de la faille dans le code et en fournissant, de préférence, des explications claires et une démonstration de la faille afin d'éviter les arguments sur la priorité du correctif.

Plutôt que d'élaborer un processus à part pour la sécurité qui peut aboutir à des retours en arrière à la fin du projet, la sécurité est intégrée à chaque itération du processus.

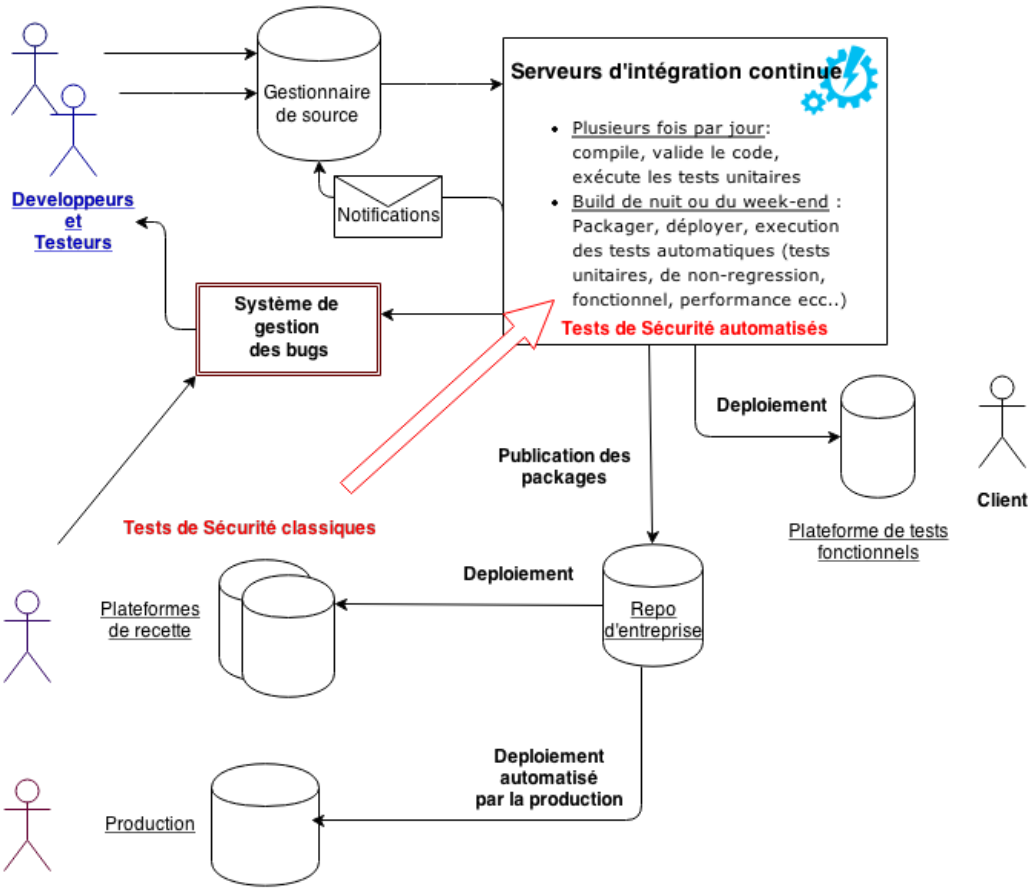

Fig 3 La sécurité dans le flux d'intégration continue

5) Créer un flux de sécurité Agile

La méthode Agile ne se contente pas d'apporter des modifications régulières au code, elle les encourage. S'intégrer avec les processus et outils des développeurs signifie par exemple que :

- Si une équipe utilise un logiciel de gestion des bugs, les problèmes de sécurité doivent être également **traités comme des bugs** à l'aide des mêmes interfaces.
- Si des tests de régression et d'automatisation sont réalisés, optez pour des outils de sécurité qui peuvent être **intégrés aux processus de tests automatiques** déjà en place.

Les failles de sécurité d'une application sont des bogues logiciels. La sécurité des applications doit donc être traitée comme un problème de qualité

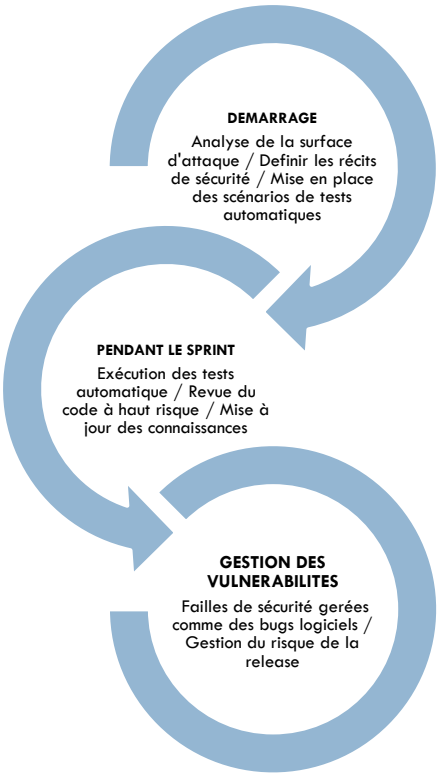

Fig 4 Exemple des activités de sécurité au cours d'un sprint

6) Fournir un programme de formation

De nombreux développeurs ne sont pas suffisamment formés sur les concepts de code sécurisé et les problématiques liées.

Avant de déléguer une tâche à une équipe de développement, vous devez fournir assez d'informations pour simplifier le processus.

Former les développeurs aux problèmes de sécurité liés à leur langage de programmation :

- Quelles sont les bonnes pratiques de programmation sécurisée ?
- Sensibilisation aux différents types de vulnérabilités (OWASP Top10)

Utilisez des outils de sécurité qui permettent de former leurs utilisateurs dans le cadre du processus. Cette formation sera non seulement bénéfique pour le projet en cours, mais aussi pour les futurs projets développés selon une méthode similaire.

C'est l'un des avantages de la méthode Agile, une amélioration constante grâce au retour d'expérience des phases précédentes.

La formation continue doit permettre par exemple à un développeur dont le code présente une faille de se former rapidement sur cette dernière ou sur d'autres failles spécifiques qui concernent l'application sur laquelle il travaille.

N'ayez pas peur de commettre des erreurs et progressez à votre rythme. Le développement Agile consiste à se former à chaque étape du projet. Toute itération comporte des améliorations et des modifications avant d'aboutir au résultat final. Le processus de sécurité des applications se déroule de la même manière, avec son lot d'améliorations et de modifications.

CONCLUSION

Le développement d'un logiciel sûr passe par la réalisation de test et l'adoption de mesures de sécurité adéquates. Il est possible avec un peu de flexibilité d'intégrer la sécurité applicative dans votre système de développement Agile.

Voici nos 6 conseils pour une mise en œuvre réussie de la sécurité du code dans les projets Agile:

1) **Améliorer la communication** et la compréhension entre équipe, en faisant participer la sécurité aux sessions de réunions de développement

2) **Établir des exigences claires** en terme de normes et de besoin de validation. Inclure ces exigences dans le concept de validation produit et de la définition du « Done ».

3) **Intégrer des récits de sécurité aux «user stories»** utilisées comme base de travail par les équipes Agile.

4) **Utiliser les flux et outils de développement existants**. Le risque inhérents aux vulnérabilités doit être compréhensible par les développeurs et remonté dans les outils de bugs et géré comme tel.

5) **Mettre en œuvre un processus sécurité en l'intégrant dans les tests automatisés** pour assurer une sécurité continue dans le cadre du développement continu; l'automatisation des tests fonctionnels a une raison d'être, il en va de même pour la sécurité.

6) **Permettre aux développeurs de comprendre leurs erreurs** et de se former en appliquant les correctifs nécessaires. Ne pas avoir peur d'apprendre et de modifier, cela fait partie de la méthode agile.

Même si les tests de sécurité ont un coût, celui qui peut être associé à des tests inadaptés sera généralement bien plus important.

Pour les organisations qui sont à la recherche d'un moyen d'intégrer les tests de sécurité de l'application dans leur cycle de vie agile, notre produit Seeker peut contribuer à ce processus de diminution du fossé entre les équipes de la sécurité et celles du développement.

LA SECURITE DES APPLICATIONS COMME COMPOSANTE NATURELLE DU CYCLE DE DEVELOPPEMENT: PRESENTATION DE SEEKER

La plupart des équipes de développement ne sont pas composées d'experts en sécurité. Seeker a été développé être utilisé par des non expert et leur permettre d'exécuter des tests de sécurité complet de leurs applications selon les normes les plus strictes.

Seeker offre un processus continu d'évaluation et de correction des vulnérabilités, permettant à tous les acteurs de travailler ensemble pour créer des applicatifs sécurisés

- **Rapide** : Seeker vérifie l'impact d'un code défaillant sur les processus métier et les données critiques en l'exploitant de façon automatique (élimination des faux positifs). Les développeurs gagnent du temps en concentrant leurs efforts sur la correction des vulnérabilités qui représentent une réelle menace pour l'entreprise. Les responsables de projet savent exactement quels sont les risques présentés par chaque version et peuvent prendre les décisions adéquates.

- **Intégré** : Les cycles de test Seeker sont courts et peuvent être inclus dans le processus agile « Build & Test ». Seeker s'intègre aux frameworks de tests automatiques, tels que Sélénium, pour créer un scénario d'attaque. Il peut être lancé depuis les serveurs d'intégration lors des tests fonctionnels automatisés. Les résultats sont fournis dans des formats déjà utilisés pour les rapports sur les bugs, tels que les systèmes de ticket.

- **Simple :** Seeker ne nécessite aucune expertise en matière de sécurité de la part de ses utilisateurs. Les risques sont expliqués dans un langage simple, en permettant aux parties prenantes de prendre des décisions. Il apporte de la simplicité au cycle de développement : il s'adapte facilement aux processus déjà en place.

ANNEXE : GLOSSAIRE DES TERMES LIES AUX METHODOLOGIES AGILE

Backlog

Liste ordonnée de toutes les choses à faire. Il y a le backlog de produit qui énumère les exigences avec le point de vue du client et le backlog de sprint qui contient les tâches de l'équipe. En anglais backlog. En français, il y a eu des tentatives pour traduire en cahier, en retard cumulé, mais l'usage de loin le plus courant est de conserver backlog

Build

En ingénierie logiciel un "build" est une version compilé d'un logiciel

Done

L'équipe affiche de façon visible une liste de critères génériques qui conditionnent le fait de pouvoir considérer un incrément comme "fini". Faute de remplir ces critères en fin de Sprint ou d'itération le travail réalisé n'est pas comptabilisé dans la vélocité.

Intégration continue

Lorsqu'une tâche est terminée, les modifications sont immédiatement intégrées dans le produit complet. On évite ainsi la surcharge de travail liée à l'intégration de tous les éléments avant la livraison. Les tests facilitent grandement cette intégration : quand tous les tests passent, l'intégration est terminée.

Kaban

L'approche Kanban consiste globalement à visualiser le Workflow (Le processus de traitement d'une tâche). On met en place un tableau de bord des items (demandes). Chaque item est placé à un instant donné dans un état. L'item évolue jusqu'à ce qu'il soit soldé. Chaque état du tableau peut contenir un nombre maximum prédéfini de tâches simultanées (défini selon les capacités de l'équipe) : on limite ainsi le WIP (Work In Progress). Il est primordial, pendant l'exécution des tâches, de mesurer le "lead-time". Il s'agit du temps moyen pour compléter un item. Cette durée sera progressivement de plus en plus courte et prévisible.

Scrum

La méthode s'appuie sur le découpage d'un projet en boîtes de temps, nommés « sprints ». Les sprints peuvent durer entre quelques heures et un mois (avec une préférence pour deux semaines). Chaque sprint commence par une vérification de la planification opérationnelle. Le sprint se termine par une démonstration de ce qui a été achevé et contribue à augmenter la valeur d'affaires du produit. Avant de démarrer un nouveau sprint, l'équipe réalise une rétrospective : elle analyse ce qui s'est passé durant ce sprint, afin de s'améliorer pour le prochain. L'adaptation et la réactivité de l'équipe de développement est facilitée par son auto-organisation.

Tests unitaires

Avant de mettre en œuvre une fonctionnalité, le développeur écrit un test qui vérifiera que son programme se comporte comme prévu. Ce test sera conservé jusqu'à la fin du projet, tant que la fonctionnalité est requise. À chaque modification du code, on lance tous les tests écrits par tous les développeurs, et on sait immédiatement si quelque chose ne fonctionne plus.

Test Driven Development (TDD) ou en français développement piloté par les tests

Technique de développement qui préconise d'écrire les tests unitaires avant d'écrire le code source d'un logiciel.

Le TDD est souvent associé, particulièrement dans la méthode XP, à la programmation en binôme. Pour expliquer les rôles des deux acteurs impliqués, il est généralement utilisé la métaphore du « rallye automobile ». Pour chaque fonctionnalité à réaliser, l'un des développeurs, nommé copilote, code le test. Ensuite, le pilote code la fonctionnalité elle-même. À chaque nouvelle fonctionnalité ajoutée, l'ensemble test-code ainsi réalisé sera de nouveau exécuté et inclus dans la liste des tests unitaires, réduisant ainsi au maximum le risque de régression.

Sprint

Bloc de temps aboutissant à créer un incrément du produit potentiellement livrable. C'est le terme utilisé dans Scrum pour itération. En anglais sprint, comme en français. Les itérations seront toujours de la même durée.

XP

L'Extreme Programming repose sur des cycles rapides de développement (des itérations de quelques semaines) dont les étapes sont les suivantes :

- une phase d'exploration détermine les scénarios client qui seront fournis pendant cette itération
- l'équipe transforme les scénarios en tâches à réaliser et en tests fonctionnels
- chaque développeur s'attribue des tâches et les réalise avec un binôme
- lorsque tous les tests fonctionnels passent, le produit est livré

Le cycle se répète tant que le client peut fournir des scénarios à livrer. Généralement le cycle de la première livraison se caractérise par sa durée et le volume important de fonctionnalités embarquées. Après la première mise en production, les itérations peuvent devenir plus courtes (une semaine par exemple).

Les principes de cette méthode ne sont pas nouveaux : ils existent dans l'industrie du logiciel depuis des dizaines d'années et dans les méthodes de management depuis encore plus longtemps.

Seeker en bref

Seeker est une solution IAST (Interactive Application Security Testing) qui permet aux entreprises de développer des logiciels sécurisés.

Seeker détecte avec une grande précision les vulnérabilités lors des processus de développement et de test. À l'aide d'une technologie unique, il analyse le code et les flux de données pendant l'exécution de l'application afin de mieux comprendre le contexte dans lequel se trouvent les vulnérabilités.

Seeker évalue l'impact de l'exploitation du code vulnérable et fournit une explication claire des risques. Seeker élimine totalement les faux positifs. Cette approche permet de confirmer ou d'infirmer l'exploitabilité et la criticité des vulnérabilités détectées sans intervention humaine. Seeker peut être entièrement automatisé et fonctionne parfaitement dans les environnements Agile et d'intégration continue.

Quotium en bref

Quotium Technologies est spécialisé dans le développement de solutions logicielles innovantes pour des applications métiers sécurisées et robustes tout au long de leur cycle de vie. En 2011, Quotium a été le pionnier dans le développement de la technologie d'analyse du code applicatif pendant son exécution, pour y détecter et corriger les vulnérabilités techniques et logiques.

Pour plus d'information www.quotium.com ou info@quotium.com

Suivez nous sur Twitter et Linkedin !

http://twitter.com/quotium

https://www.linkedin.com/company/quotium-technologies