DevSecOps

dynamique, rapide, efficace
et sécurisé

swisscom

Mentions légales

Editeur	Swisscom SA
Auteur	Group Security
Rédaction	René Mosbacher, Faktor Journalisten AG, Zurich
Illustration	Agence Nordjungs, Zurich
Copyright	© Mars 2019 by Swisscom SA, Group Security, Berne

Table des matières

1 Avant-propos

Aujourd'hui, nous vivons des vies connectées. Internet n'a ni géographie ni frontières. En créant le réseau Internet, l'humanité a ouvert une boîte de Pandore où les frontières tangibles et les ennemis identifiables n'existent plus.

De fait, nous sommes la première génération de l'Histoire qui risque davantage d'être victime d'un crime dans le monde virtuel que dans le monde réel. C'est un changement majeur.

Dans ce nouveau monde, les entreprises sont également confrontées à de nouveaux risques. Les failles de sécurité dans les grandes entreprises font souvent la une des journaux. Aujourd'hui, toute entreprise est également une société de logiciels, et toute entreprise doit se préoccuper de la sécurité sur Internet. Aujourd'hui, la cybersécurité doit être un sujet permanent au plus haut niveau de l'entreprise.

Finalement, il n'existe que deux types de problèmes de sécurité: les problèmes techniques et les problèmes humains.

Les problèmes techniques peuvent se corriger avec des patchs. Mais pas les problèmes humains. Il n'existe pas de patch contre la bêtise. Vous pourrez leur répéter autant de fois que vous voulez, les utilisateurs cliqueront toujours sur n'importe quel lien, ils ouvriront toujours n'importe quelle pièce jointe, et ils saisiront toujours leur mot de passe sur n'importe quel site de phishing. Immanquablement.

En fin de compte, les vulnérabilités de nos systèmes correspondent simplement à des bugs dans le code. Pourquoi le code contient-il des bugs? Parce que les programmes sont programmés par des êtres humains et que les humains commettent des erreurs.

Mais comment pouvons-nous limiter le nombre d'erreurs commises par les programmeurs? Nous avons besoin d'une meilleure ingénierie sécurité. Nous avons besoin d'une meilleure culture en matière de sécurité. La sécurité doit être intégrée, au lieu de représenter un simple périmètre autour des applications et des données qu'elles traitent. La sécurité doit faire partie intégrante de l'ensemble du cycle de vie d'un projet logiciel.

La sécurité ne s'arrête pas à l'ingénierie logicielle et aux audits de sécurité. Quel que soit le type de système que vous concevez, vous devez toujours supposer qu'il y aura de toute façon une faille: quelqu'un s'introduira par un chemin auquel vous n'avez jamais pensé. C'est la raison pour laquelle les entreprises devraient se concentrer sur la détection des failles et la manière d'y réagir. La résilience est fondamentale.

Nous disposons tous de ressources et de budgets limités pour défendre nos réseaux. Comprendre l'ennemi nous permet de focaliser nos ressources sur les points clés.

Et dans notre lutte contre les pirates, ce sont les pirates qui représentent l'une de nos meilleures ressources. Autrement dit, nous avons besoin de bons pirates pour attraper les méchants pirates. C'est pour cela que les programmes Bug Bounty fonctionnent. Lancer un programme Bug Bounty demande beaucoup de travail, mais cela vous apporte de nouveaux points de vue sur la sécurité de vos systèmes. Personnellement, je me réjouis que de nombreuses grandes entreprises mettent désormais en place des programmes Bug Bounty ouverts, car j'ai ainsi une réponse facile à donner aux jeunes hackers impatients que je rencontre. «Alors comme ça tu veux pirater une entreprise? D'accord: va pirater Microsoft, Google ou Apple. C'est parfaitement légal, tant que tu le fais dans le cadre de leurs programmes Bug Bounty. Ils vont même te payer pour ça.»

Je travaille dans la sécurité depuis près de 30 ans. Notre travail est sans fin.

Je vous adresse mes meilleurs vœux de réussite dans le développement de systèmes de sécurité. Cette brochure sur le DevSecOps (développement-sécurité-exploitation) fournit un bel exemple de ce que nous pouvons faire concrètement pour améliorer la sécurité du monde qui nous entoure.

Merci pour votre travail.

Mikko Hypponen
Chief Research Officer at F-Secure

2 A propos de cette publication

2.1 Pourquoi ce livret?

Avec environ 20 000 collaborateurs, Swisscom est une grande entreprise issue du secteur des télécommunications. Au fil des années, de nouveaux domaines d'activité ont été intégrés et actuellement, Swisscom est également un prestataire classique de services informatiques et l'entreprise suisse la plus importante pour les technologies de l'information et de la communication (TIC).

En 2016, la décision a été prise de développer l'entreprise selon des principes agiles. Le département d'innovation, suivi par les départements de développement, ont effectué les premiers pas. Désormais, l'ensemble de l'entreprise est concerné. Les chapitres *3.3* et *3.4* montrent comment Swisscom utilise concrètement DevOps.

Au cours de la transformation, il est alors apparu clairement que le soutien fourni par le département de sécurité central ne pouvait être appliqué que dans une mesure limitée dans un environnement agile. Les initiatives de sécurité décrites dans ce livret ont été lancées ou développées sur la base de cette conclusion.

La présente publication propose un regroupement de Best Practices de l'environnement Dev(Sec)Ops. Elle a été élaborée du point de vue de l'environnement entrepreneurial et documente également les expériences ayant vu le jour jusqu'à présent au sein de l'entreprise. Nous espérons que d'autres entreprises et spécialistes confrontés à des défis similaires puissent ainsi en profiter.

> En 2016, le premier département entièrement conçu selon les principes DevOps a été créé dans le secteur du développement chez Swisscom. L'entreprise a ainsi pu découvrir comment l'agilité pouvait être mise en œuvre au sein d'un service regroupant plus de 100 personnes.
>
> De plus, il a été possible de constater que «l'agilisation» du développement ne réduisait pas les problèmes opérationnels. Certes, de plus en plus de collaborateurs de l'exploitation ont été intégrés au développement, permettant ainsi de développer des solutions véritablement exploitables en pratique. En revanche, pour l'exploitation effective 7×24 de toutes les applications existantes, il y avait de moins en moins de personnel disponible.

C'est pourquoi l'organisation a de nouveau été transformée en 2017 selon la devise «you build it, you run it». A la place de la subdivision classique entre le développement et l'exploitation, le service informatique a été structuré en un secteur chargé des logiciels et un autre chargé des infrastructures. Cependant, au sein de ces secteurs, toutes les tâches (développement, tests, sécurité, exploitation) sont sous la responsabilité de l'équipe correspondante. Cela a permis d'améliorer considérablement la conscience de la responsabilité pour le code développé.

Avec ces évolutions, de plus en plus de secteurs ont été introduits au mode de travail de DevOps. Mais il est vite devenu clair qu'une structure était nécessaire pour coordonner ce mode travail au-delà d'une seule équipe. Swisscom a décidé d'introduire le Scaled Agile Framework (SAFe) et de l'appliquer partout où cela était nécessaire.
Grâce aux adaptations, l'agilité opératoire a considérablement augmenté. Il est également devenu clair que les avantages de DevOps ne concernaient pas seulement le développement et l'exploitation, bien au contraire. La volonté d'intégrer l'ensemble de la création de valeur a toujours été au centre des préoccupations de DevOps.

Ainsi, en 2018, l'objectif principal était de déplacer progressivement l'agilité du niveau opératoire vers le niveau stratégique. Le Business devait être encore plus impliqué et les processus d'entreprise adaptés lorsqu'ils étaient contraires aux principes d'agilité.
Un «voyage DevOps» nécessite du temps et de l'endurance. C'est pourquoi ces prochaines années, Swisscom continuera de travailler pour se développer continuellement. Cela concerne d'une part les compétences techniques comme le Decoupling, le Continuous Delivery ou l'automatisation des tests. Mais d'autre part, il s'agit aussi du niveau culturel, c'est-à-dire une meilleure organisation autonome des équipes et une plus grande optimisation des flux.

2.2 Public cible

Cette publication s'adresse aux spécialistes qui conçoivent eux-mêmes un environnement DevOps, ou bien qui peuvent ou souhaitent participer à sa conception. Il s'agit en particulier de s'adresser à des personnes qui doivent également garantir la sécurité dans un environnement hautement agile. Les concepts fondamentaux leur sont expliqués avec de courts rapports d'expérience issus de l'environnement de Swisscom. Il est attendu que les lecteurs aient une connaissance de base de DevOps. Des ressources, comme par exemple des sites web ou des livres permettant d'approfondir ces thématiques sont mis en lien ou énumérés au cours et à la fin du document.

2.3 Organisation de la publication

La première partie de ce livret aborde les idées et principes fondamentaux de DevOps. Ce faisant, les aspects suivants sont particulièrement approfondis:

- Why – pourquoi une organisation choisi-t-elle DevOps?
- What – qu'est-ce qui définit et qualifie DevOps?
- How – comment DevOps est-il introduit?

La deuxième partie se base sur ces conclusions et aborde en particulier les aspects relatifs à la sécurité dans DevOps. Les sous-chapitres commencent d'abord par les points à prendre en compte en matière de People, Process et Technology.

La deuxième partie est structurée selon un Software Development Lifecycle (SDLC, voir image 1) simplifié. Elle traite d'abord les thèmes globaux comme Training & Awareness (T&A) ainsi que la manière dont la sécurité doit être organisée afin de pouvoir s'intégrer à la dynamique d'une organisation DevOps. Ces thèmes sont décisifs si l'organisation de sécurité doit participer avec succès à la conception de DevOps. Ensuite, le SDLC est analysé étape par étape afin de savoir comment gérer la sécurité pour qu'elle puisse être efficace au sein de l'environnement DevOps.

Le chapitre suivant aborde la «Deployment Pipeline», c'est-à-dire la fabrique elle-même. De manière peu surprenante, l'automatisation y joue un rôle important. Pour conclure, il est expliqué comment les nouvelles approches peuvent être utilisées de manière bénéfique pour la détection et la défense rapides d'attaques.

Des rapports d'expérience et approches de solutions tirées de l'environnement de Swisscom sont documentés dans certaines sections. Ils doivent permettre de mieux comprendre les principes décrits et d'apprendre des erreurs déjà commises. Ces parties sont respectivement signalées en bleu.

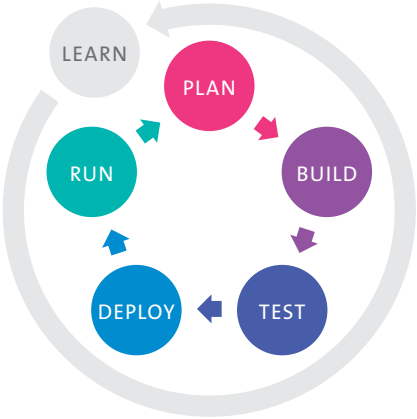

Image 1 | Un Software Development Lifecycle simplifié

3 DevOps – Introduction

3.1 Why – pourquoi DevOps?

Faster, better, cheaper, happier: avec DevOps, il s'agit de créer plus rapidement une valeur ajoutée pour le business et d'élaborer des systèmes plus solides à l'origine de produits davantage axés vers les clients. Pour cela, des formes adaptées de collaboration et d'organisation d'équipe garantissent que ces efforts ne soient pas réalisés au détriment de la satisfaction des collaborateurs. Ceux qui maîtrisent cela seront parés pour l'univers informatique en évolution de plus en plus rapide qui se caractérise aussi par une hausse de la volatilité, de l'incertitude, de la complexité et de l'ambiguïté (VUCA: volatility, uncertainty, complexity and ambiguity).

3.2 What – qu'est-ce que DevOps?

DevOps s'est développé à partir de plusieurs mouvements. L'Agile Manifesto, le Lean Movement, le Continuous Delivery Movement et le Toyota Kata en font notamment partie[1]. DevOps n'est donc pas un cadre, ni un outil et pas non plus une forme d'organisation. DevOps doit plutôt être considéré comme une intégration et un développement systématique des concepts évoqués.

L'expert informatique John Willis est l'un des pionniers DevOps qui influence fortement le secteur. Il a introduit l'acronyme CALMS (d'abord simplement CAMS avant que Jezz Humble ajoute le L) à la description de DevOps.[2] CALMS se compose de:

- **Culture** – une collaboration intersectorielle avec une culture de responsabilité partagée
- **Automation** – automatisation du plus grand nombre de tâches possibles
- **Lean** – représentation visuelle du flux de valeurs et des paquets de travail à chaque instant
- **Measurement** – élaborer des hypothèses, valider et apprendre[3]
- **Sharing** – partager les responsabilités et les succès, aussi bien entre l'exploitation et le développement qu'au-delà des limites de l'équipe

[1] The DevOps Handbook de Patrick Debois, Jez Humble, Gene Kim, John Willis

[2] DevOps Culture (Part 1) de John Willis, URL: *itrevolution.com*

[3] Hypothesis-Driven Development de Jeffrey L. Taylor, URL: *drdobbs.com*

3.3 How – qu'est-ce qui est nécessaire pour DevOps?

3.3.1 Aptitudes

Le modèle de référence IBM «DevOps: The IBM approach – Continuous delivery of software-driven innovation» confère un bon aperçu des aptitudes nécessaires.[4]

Swisscom mise sur une version adaptée du modèle de référence IBM. L'entreprise l'utilise en tant que système global couvrant non seulement le développement, mais également toutes les aptitudes pertinentes le long de la chaîne de valeur. Le modèle Swisscom décrit des aptitudes pertinentes pour une exploitation DevOps, mais il ne s'agit pas d'un schéma de procédure.

Image 2 | Modèle Swisscom des aptitudes nécessaires à DevOps

[4] DevOps: The IBM approach – Continuous delivery of software-driven innovation (surtout page 6), URL: *developer.ibm.com/community*

Afin que DevOps fonctionne, il faut pour l'essentiel que les conditions suivantes soient remplies:

1. **L'équipe se trouve au centre:** Une équipe interdisciplinaire, entièrement dédiée, autonome se trouve au centre et communique au même niveau *(voir également 3.2, partie «Culture» dans CALMS)*.

2. **Continuous Business Planning:** Il s'agit de la capacité à réagir rapidement aux besoins des clients et d'intégrer en continu des retours correspondants. Les besoins des clients sont toujours au centre des préoccupations et définissent le produit.

3. **Collaborative Development:** Ce terme désigne la capacité de générer continuellement de la valeur avec de courtes itérations et des équipes interdisciplinaires constituées de représentants du business, du développement, de la sécurité, des tests et de l'exploitation. Ceci suppose que le code soit intégré en continu au dépôt de code (Code Repository), si possible plusieurs fois par jour. Ainsi, tout est considéré sous forme de code. En plus du code source proprement dit, cela concerne également l'infrastructure, les Deployment Scripts, le monitoring, les données de test, etc.

4. **Continuous Testing:** Ceci regroupe la capacité à tester le code de manière reproductible et récurrente. Des tests automatiques permettent de l'analyser sous plusieurs points de vue juste après la saisie dans le Repository. Ainsi, il est possible d'effectuer les tests de façon précoce et continue plutôt qu'à la fin d'une chaîne de processus, peu avant l'introduction dans la production. Les tests font partie du processus global. Cela permet d'améliorer l'efficacité des coûts puisque les erreurs peuvent être résolues de manière plus précoce et donc plus économique[5] dans le cycle.

5. **Continuous Release & Deployment:** Cela désigne la capacité de réaliser automatiquement un Build et de le fournir à un environnement donné. Ceci est nécessaire afin d'exploiter l'ensemble du potentiel des tests automatisés. Le Code Check-in dans le Repository permet d'exécuter une série de processus automatiques, du Build aux tests, en passant par le Deployment. Cela réduit considérablement les activités manuelles.

6. **Continuous Monitoring:** Il s'agit de surveiller à tout moment l'ensemble du système sur toutes les plateformes. Juste après l'intégration d'une fonctionnalité, il est ainsi possible de déterminer si celle-ci influence les performances du système global. Ce feedback précoce permet d'améliorer encore la qualité.

[5] Figure: Relative Cost of Fixing Defects de Maurice Dawson, URL: *researchgate.net*

7. **Continuous Customer Feedback and Optimization:** C'est comme cela que l'on désigne la capacité d'améliorer constamment un logiciel en se basant sur des mesures et le retour des clients et de l'entreprise. Pour cela, le comportement des clients est analysé de manière très précise et différents canaux sont utilisés afin de bénéficier de façon précoce d'un retour complet.

3.3.2 Flexibilité – DevOps dans les grandes organisations

Toutes les approches agiles placent l'équipe au centre. C'est essentiel, car la valeur d'une petite équipe (comprenant dans l'idéal 5-7 personnes) motivée n'est pas assez reconnue. Dans le meilleur des cas, de telles équipes peuvent développer leurs services et fournir le code de manière entièrement autonome.

Cependant dans les grandes organisations possédant une architecture informatique fortement connectée, des équipes autonomes sont difficilement envisageables. Il se pose donc la question de savoir comment la flexibilité peut être atteinte. Ici, le Scaled Agile Framework (SAFe) peut par exemple être utile. Il s'agit d'un regroupement de bonnes pratiques qui décrit un possible dimensionnement au-delà d'une seule équipe.[6]

> Swisscom a pris la décision de faire appel au SAFe afin de pouvoir pratiquer «Agile at Scale». Dans la pratique, cela signifie que si un produit est trop complexe et trop grand pour être géré uniquement par une équipe autonome, Swisscom utilise des structures SAFe. Pour cela, le cadre aide à créer une compréhension commune de la façon dont fonctionne «Agile at Scale». Mais seuls les éléments et rôles de SAFe offrant une valeur ajoutée effective sont utilisés. Swisscom reprend certains éléments tels quels tandis que d'autres sont adaptés aux besoins, comme par exemple le rôle de la sécurité.

3.4 Impact – quelles sont les conséquences de DevOps?

Ceux qui souhaitent introduire DevOps dans leur organisation doivent être conscients qu'ils influenceront ainsi l'ensemble de la chaîne de création de valeur. Nous allons aborder les principaux aspects dans ce qui suit.

Chez le client

La manière dont les services sont produits n'est pas importante pour les clients. Ils n'ont qu'une exigence: que leurs besoins soient couverts de manière rapide et fiable. Vos systèmes doivent être disponibles et fonctionner de manière sûre, sans erreur. Dans notre monde évoluant rapidement, ceci est de plus en plus difficile en utilisant des modèles

[6] SAFe. Scaled Agile Inc., URL: *scaledagileframework.com*

classiques. Au contraire, avec DevOps, le client bénéficie de solutions mieux adaptées à ses besoins et pouvant être implémentées plus rapidement.

Au sein de sa propre entreprise
Le State of DevOps Report[7] analyse depuis quelques années l'influence des pratiques DevOps sur les entreprises. Il a pu être clairement prouvé que les entreprises avec une maturité DevOps élevée étaient plus performantes économiquement. Mais dans ce contexte, il est important de noter que DevOps ne constitue pas un programme d'économie de coûts. Une efficacité plus élevée est le résultat de pratiques DevOps correctement introduites, mais ne doit pas constituer les intentions initiales.

Pour la mise en place organisationnelle
L'agilité et DevOps misent surtout sur des équipes autonomes. Si possible, les décisions sont prises de manière décentralisée. Naturellement, un tel environnement nécessite des qualités de direction spéciales. Il est important que les équipes soient diverses et Interdisciplinaires. D'après les expériences, cela entraîne de meilleures solutions.

Dans la mesure du possible, la composition des équipes doit être constante. Cela favorise un apprentissage commun et mutuel. Il en résulte aussi un important changement de paradigme pour les processus: le travail doit être amené vers l'équipe et non l'équipe vers le travail. Et cela a ensuite pour conséquence que l'équipe travaille bien plus efficacement au développement continu d'un produit qu'au sein de projets limités. L'important, c'est que dans la mesure du possible, les décisions soient prises où celles-ci ont un impact. C'est également pour cela que les équipes interdisciplinaires sont tellement importantes. DevOps doit aussi réduire les structures organisationnelles. Ainsi, l'importance accordée au flux de valeur est bien plus élevée que l'appartenance organisationnelle. Par conséquent, l'organisation fonctionnelle devient beaucoup plus importante que l'organisation structurelle.

> Chez Swisscom, la mise en œuvre de DevOps a en particulier entraîné des adaptations organisationnelles au sein du service informatique. Auparavant, le développement et l'exploitation des applications se déroulaient dans des services séparés ayant différents objectifs. «Dev» était mesuré au nombre de fonctionnalités fournies. «Ops» au contraire était mesuré selon la stabilité des systèmes et tentait donc de minimiser le nombre de changements. Afin de résoudre ce conflit d'intérêts et de renforcer la responsabilité commune, ces services ont été regroupés. Désormais, chaque équipe est non seulement responsable du développement, mais également de l'exploitation de ses applications.

[7] State Of DevOps Report de Puppet + Splunk, URL: *puppet.com*

Au sein de l'équipe et chez les individus
DevOps se focalise fortement sur l'équipe et la responsabilité de celle-ci. Les objectifs sont définis au sein de l'équipe et les membres se soutiennent afin de les atteindre. Ce sont les objectifs de l'équipe et non ceux des différents membres qui prévalent.

DevOps raccourcit les cycles et intensifie fortement la collaboration au sein de l'équipe. C'est pourquoi la communication directe entre tous les participants doit être consciemment encouragée. L'Agile Manifesto fournit par exemple des concepts à ce sujet[8].

Dans son livre «Drive», Daniel Pink[9] écrit que la motivation dépend surtout des facteurs «Autonomy», «Mastery» et «Purpose». DevOps satisfait très bien à ceux-ci:
- Autonomy devient possible grâce à des décisions décentralisées et l'autonomie de l'équipe.
- Mastery est favorisé grâce à l'apprentissage continu nécessaire.
- Purpose, c'est-à-dire le sens et le but, parce que les responsabilités et les effets du travail peuvent être vécus de manière très directe. On n'examine pas seulement la petite roue dans une grande machine, mais l'on est responsable d'un produit concret jusqu'à sa production. La plupart des collaborateurs apprécient cela.

Généralement, avec DevOps, le processus de développement devient plus court et plus rapide. De plus, les circuits de feedback sont raccourcis. Par conséquent, les informations et les connaissances, ainsi que la responsabilité sont transférées plus rapidement au début du processus, c'est-à-dire au développement. Cet effet est également appelé «Shift Left».[10]
Avec DevOps, les collaborateurs travaillent toujours de la manière la plus dédiée pour leur équipe. Cela permet d'éviter les conflits relatifs aux priorités, de développer une culture d'équipe saine et de favoriser l'apprentissage commun. De plus, ce qui a déjà été décrit concernant l'influence de DevOps sur la structure organisationnelle s'applique: le travail de l'équipe s'oriente plutôt vers le développement continu de produits plutôt que vers le traitement de différents projets.

[8] Manifesto for Agile Software Development de Kent Beck et al., URL: *agilemanifesto.org*

[9] Drive de Daniel Pink, URL: *danpink.com*

[10] Shifting Left – Approach and Practices de Paul Bahrs, URL: *slideshare.net*

4 DevSecOps – la sécurité dans le contexte DevOps

Par définition, la sécurité est déjà[11] fortement représentée dans DevOps. Par conséquent, à première vue, il ne semble pas nécessaire de lui accorder plus de poids par un terme artificiel comme DevSecOps. Cependant, l'expérience montre que dans le cadre de «Shift Left», des thèmes sont souvent négligés. C'est en particulier la sécurité qui est menacée. Pour l'essentiel, trois principaux défis résultent de DevOps:

- **Suppression de points de contrôle de processus fixes respectivement des échéances**
  Ce qui s'applique déjà aux processus itératifs de conduite de projet est encore plus valable pour DevOps: différentes étapes nécessitant une validation manuelle sont contreproductives pour DevOps, car elles prolongent la durée nécessaire entre l'idée et la production. Par conséquent, avec DevOps, il n'est pas possible d'ancrer des rapports de sécurité avant le lancement de la production, comme c'est souvent le cas pour les projets conduits de manière classique.

- **Suppression des pouvoirs entre le système et l'application (Segregation of Duties)**
  En raison de la diversification des capacités et de la responsabilité de bout en bout qui en découle au sein des équipes DevOps, tous les membres de l'équipe bénéficient des mêmes droits. La conséquence est que beaucoup plus de personnes ont accès à des données précieuses.

- **Incertitude relative aux responsabilités pour les risques**
  Ce qui a été ancré via la fonction hiérarchique au sein d'organisations à la structure strictement hiérarchique est lié au rôle dans une organisation DevOps. Les risques sont endossés ou réduits par un responsable.

C'est pour cela que les spécialistes du domaine de la sécurité ont créé le terme DevSecOps. Il doit aider à mieux comprendre la sécurité dans l'environnement DevOps. Mais il illustre aussi que toutes les activités autour du cycle de vie des produits doivent être étroitement liées entre elles. Pour la sécurité, cela signifie dans un premier temps que les activités sont uniquement adaptées à DevOps lorsqu'elles sont applicables de manière itérative avec la sécurité.

[11] The DevOps Handbook de Patrick Debois, Jez Humble, Gene Kim, John Willis

Les conditions cadres pour la sécurité peuvent être représentées via les trois piliers People, Process et Technology:

People	Process	Technology
Les personnes qui vivent DevOps sont amenées à prendre intrinsèquement en compte la sécurité d'un produit	Les processus favorisent le développement sûr d'un produit Au sein d'organisations en mutation, les interlocuteurs pertinents en matière de sécurité doivent être faciles à trouver	«Security as Code» est mis en œuvre de manière systématique afin de devenir compréhensible, reproductible et modifiable

Ces trois piliers sont de nouveau abordés au début de chaque sous-chapitre. Les points les plus importants du domaine d'activité respectif doivent être décrits. Les trois piliers peuvent être définis comme suit:

- **People:** Activités et mesures ayant un effet sur la culture DevOps vécue
- **Process:** Tout ce qui est nécessaire en matière de processus et de structure organisationnelle afin que la sécurité occupe sa place due dans DevOps
- **Technology:** Le cœur de DevOps est l'automatisation de processus répétés. Ici, il s'agit de leur traitement et des opportunités qui en résultent.

Dans une mise à disposition du produit selon DevOps, il est difficile d'attribuer précisément différentes activités à l'un des trois piliers. Par conséquent, par la suite, les piliers sont toujours représentés ensemble.

Les sous-chapitres suivants décrivent les compétences essentielles en matière de sécurité (Capabilities) devant être atteintes au sein d'une organisation DevOps aux dimensions d'un groupe. De même, ils montrent où existent des potentiels de synergie entre les différentes activités de sécurité. Cela permet de minimiser l'investissement et les frictions supplémentaires au minimum. Pour cela, la chronologie des sous-chapitres s'oriente selon les différentes activités du Software Development Lifecycle simplifié *(voir chapitre 2.3)*, c'est-à-dire:

- Training et organisation
- Planification et implémentation d'un produit/d'une fonctionnalité
- Deployment et exploitation

4.1 Training & Awareness

Eléments essentiels dans ce chapitre		
People	**Process**	**Technology**
Savoir où la sécurité doit être prise en compte	Training & Awareness constitue une obligation pour les parties prenantes concernées	Evolutivité et durabilité des méthodes Awareness et Training

Lorsqu'il s'agit de gérer de manière technique les décisions, les priorisations ou autres activités de direction, il est important que tous les participants participent au Security Training & Awareness (T&A). Dans un environnement DevOps, tous les membres de l'équipe doivent être conscients qu'un comportement inadapté ou une erreur systémique peuvent avoir d'importantes conséquences. De même, tous doivent avoir conscience des actifs fondamentaux à protéger (par exemple les données).

Une bonne sensibilisation aide à développer la conscience vis-à-vis des incertitudes. Ceci est très utile, car par rapport à d'autres exigences, la sécurité est souvent moins visible et palpable. Pour les parties prenantes, une intuition saine facilite la réflexion relative aux conséquences de leurs actes.

4.1.1 Pour qui?

D'une manière générale, au sein d'une organisation, tous les membres sont co-responsables de la sécurité. Par conséquent, un programme Training & Awareness adapté à tous les collaborateurs et transmettant de larges connaissances de base est nécessaire (par exemple pour le traitement sûr des données). De plus, un tel programme doit également transmettre des connaissances spécifiques aux rôles. Cela permet aux collaborateurs de se former dans leur domaine de compétences, si nécessaire jusqu'à un niveau d'expert.

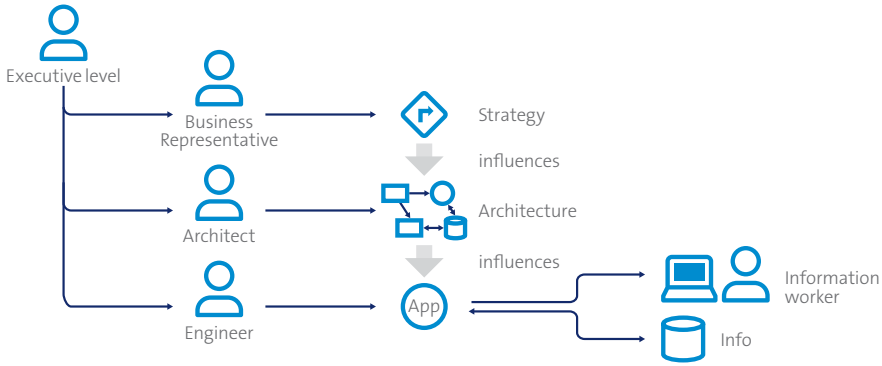

Image 3 | Rôles impliqués dans les connaissances de base DevOps et sensibilisation générale

Chaque membre de l'organisation doit posséder des connaissances de base minimales en matière de sécurité et de protection des données. Des formations de base doivent donc être proposées, par exemple sous forme d'e-learning. Tous les rôles impliqués dans DevOps font partie du groupe cible de telles formations, c'est-à-dire:

- **Engineers** – s'occupent de la mise à disposition technique des produits
- **Architects** – sont responsables des interactions et processus corrects concernant les produits et leurs interfaces vers l'extérieur
- **Business Representatives** – priorisent les aspects de contenu du produit et influencent ainsi l'orientation du développement
- **Executives** – sont responsables des aspects globaux dans l'entreprise et pilotent ceux-ci
- **Information Workers** – les bénéficiaires (internes) de produits mis à disposition

Unités Training & Awareness spécifiques aux rôles
Les unités de formation spécifiques aux rôles pour Training & Awareness permettent aux membres des équipes d'identifier les thèmes pertinents relatifs à la sécurité et à la protection des données, et d'appliquer leurs connaissances dans la pratique. Ici, il s'agit donc de transmettre une compréhension de base permettant par exemple à un architecte de prévoir les composants de sécurité nécessaires. Les unités doivent par exemple également aider le Business Representative à prévoir la marge de manœuvre et les ressources nécessaires pour la sécurité et les corrections. Afin que cela fonctionne, les contenus des formations doivent être adaptés au rôle correspondant.

Training & Awareness pour les domaines spécialisés
Des formations très approfondies et spécifiques sont nécessaires pour Engineering et Development. L'objectif doit être de disposer au sein des équipes DevOps d'au moins une personne spécialisée *(voir 4.2.2 → Security Champion)* possédant des connaissances élargies en matière de sécurité et de protection des données. Cette personne doit faire office de lien avec l'organisation de sécurité centrale. Des formations spécifiques sont particulièrement utiles pour les ingénieurs.

4.1.2 Concept Training & Awareness

Le concept pour Training & Awareness se base sur les trois principaux piliers suivants:

1. **Formation continue** avec de courtes unités d'apprentissage (Micro Learnings) qui proposent partiellement aussi des contenus répétés. Lorsque c'est possible, les longues unités d'apprentissage sont scindées en unités plus courtes. Cela facilite la compréhension des contenus et augmente l'acceptation des collaborateurs.
2. **Les actions Awareness** permettent d'informer les collaborateurs en ce qui concerne les possibles dangers (par exemple les campagnes fake-phishing). Cela doit créer un «bruit de fond» qui ancre durablement la Security Awareness dans les esprits.
3. **Des canaux spéciaux de communication** aident à diffuser les thèmes relatifs à la sécurité et en particulier les offres Training & Awareness.

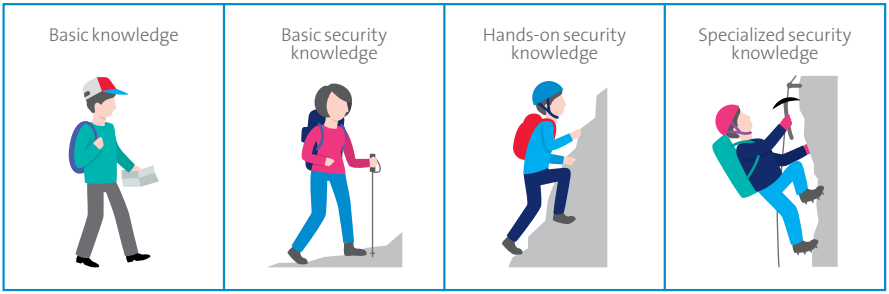

Image 4 | Les quatre mascottes de la certification Swisscom dans le domaine DevSecOps: Trekker, Hiker, Mountaineer, Alpinist

Au niveau du contenu, le concept Training & Awareness prévoit quatre niveaux chez Swisscom. A chaque niveau, les contenus deviennent plus techniques, ce qui restreint également le public cible. Les participants peuvent obtenir un certificat Swisscom interne pour chaque niveau. Les modules **Trekker** constituent le premier niveau. Ils servent surtout à faire connaître les services relatifs à DevOps disponibles en interne et gérés de façon centralisée. Des approches DevOps fondamentales comme le Source Code Management, Continuous Integration/Continuous Deployment sont également présentées. De plus, les participants découvrent le vocabulaire commun que l'on retrouve ensuite aux niveaux suivants. A la fin, même les débutants doivent disposer des informations nécessaires et connaître des services DevOps sûrs avec lesquels ils peuvent rapidement travailler de manière productive. Des contenus spécifiques relatifs à la sécurité ne sont pas présentés.

Dans les modules **Hiker**, les Security Services sont présentés. D'autre part, les participants sont introduits aux principales vulnérabilités comme le Cross-Site-Scripting ou SQL Injection. Pour la priorisation des catégories de vulnérabilité, des données du programme Bug-Bounty de Swisscom *(voir chapitre 4.6)* ont été utilisées. En guise d'alternative, l'Open Web Application Security Project (OWASP) propose aussi le projet Top-10. Les deux niveaux peuvent être contrôlés facilement sous forme automatisée via Multiple-Choice-Quiz.

Les modules **Mountaineer** sont destinés aux personnes qui souhaitent découvrir comment l'on réagit dans la pratique aux vulnérabilités présentées dans le module Hiker. Pour cela, les participants qui élaborent des systèmes ou des produits changent de rôle pendant un jour ou deux afin de découvrir le point de vue de l'attaquant. Les participants qui souhaitent obtenir ce

certificat doivent savoir utiliser les résultats dans leurs tâches quotidiennes. Les expériences doivent également être partagées avec la communauté DevOps.

Au niveau **Alpinist**, un haut degré de spécialisation est nécessaire. Ici, l'objectif est un certificat externe reconnu sur le marché, par exemple le Certified Secure Software Lifecycle Professional (CSSLP). A la fin, les participants doivent pouvoir influencer la culture de la sécurité interne de l'organisation.

4.2 Flexibilité organisationnelle

Eléments essentiels dans ce chapitre

People	Process	Technology
Garantir des canaux de communication courts Favoriser la communication matricielle	Documenter et communiquer clairement les responsabilités	Permettre à tout moment un aperçu de l'organisation et des différents rôles

Même à l'ère de DevOps, ancrer la sécurité en tant que comportement dans l'organisation reste un défi. Contrairement aux autres attributs non-fonctionnels comme la performance ou la stabilité d'exploitation, un manque relatif à la sécurité n'est pas immédiatement perceptible pour un produit. Nous remarquons au quotidien que lorsqu'une application réagit lentement, les clients deviennent assez rapidement actifs. Mais s'il existe un manque au niveau de la sécurité, ils le remarquent, lorsque c'est le cas, beaucoup plus tard, lorsque quelque chose n'a pas bien fonctionné. Il s'agit donc de conférer à la sécurité la visibilité qu'elle mérite. Si des données sont perdues ou qu'elles sont manipulées, les conséquences sont souvent fatales.

Généralement, au sein du service informatique, environ 10 exploitants, mais seulement 1 spécialiste de la sécurité sont disponibles pour 100 développeurs. L'accompagnement en face à face des collaborateurs concernant les questions de sécurité n'est donc pas réaliste. Cela sera également le cas dans une organisation DevOps. Il est donc important que tous les parties prenantes impliquées dans DevOps (Business, développement, test, exploitation: *voir aussi «Collaborative Development» au chapitre 3.3*) développent de meilleures compétences en matière de sécurité tout en étant assistées par l'organisation de sécurité centrale. Ici, la communication matricielle est également utile. Celle-ci désigne la mise en réseau de personnes endossant le même rôle au sein des différentes équipes. Ainsi, elles peuvent

constamment échanger avec leurs collègues spécialisés dans toute l'entreprise et faire profiter leur propre équipe de leurs connaissances.

> Comme déjà décrit dans le *chapitre 2.1*, Swisscom utilise le Scaled Agile Framework (SAFe) sous une forme appropriée. L'un des domaines adaptés est l'intégration de la sécurité dans l'organisation. Celle-ci n'est pas assez efficace pour les usages de Swisscom dans le framework de base. Ce chapitre analyse de manière plus précise où et pourquoi la mise en œuvre de la sécurité chez Swisscom diverge de SAFe. Il explique aussi comment la sécurité est intégrée d'un point de vue organisationnel.

4.2.1 Diversité des capacités dans l'organisation DevOps

Avec l'approche «Shift Left», de nombreuses responsabilités sont reportées vers l'équipe DevOps. Cela signifie qu'en plus du comportement fonctionnel du produit, l'équipe doit aussi garder en vue d'autres aspects qualitatifs. L'éventail des thèmes à traiter est donc considérablement élargi. Une vue d'ensemble du produit est nécessaire et cela est uniquement possible si l'équipe est hétérogène et diversifiée.

Lorsque l'équipe regroupe des spécialistes de différents domaines, il est plus facile pour chacun des membres de formuler des scénarios de menace pertinents. Il est également plus facile de comprendre pourquoi certaines activités qui ne sont pas directement rentables peuvent être très importantes. D'une manière générale, cela augmente donc la conscience de l'équipe en matière de sécurité grâce à la diversité des compétences.

4.2.2 Rôles de sécurité au sein de l'organisation

Afin que la conscience relative à la sécurité puisse se déployer au mieux, il faut qu'elle soit appuyée par un système évolutif de rôles pertinents en matière de sécurité. Au sein de la Security Community, l'utilité de cette évolutivité est incontestée. Elle peut par exemple être mise en œuvre à l'aide d'un modèle Security Champion[12].

Généralement, les Security Coaches et Security Officers sont rattachés à l'organisation de sécurité centrale. Le Security Officer dispose d'une vue d'ensemble et identifie les risques techniques ainsi que les risques commerciaux qui en résultent. Les décideurs commerciaux constituent ses interlocuteurs. Il doit donc discuter des risques avec eux. Pour cela, les représentants Business font office de responsables du risque. Le Security Officer les aide à classifier correctement les risques afin de pouvoir les prioriser de manière adaptée lors du développement continu d'un produit ou d'un service.

[12] Security Champions. Open Web Application Security Project, URL: *owasp.org*
Security Champions Playbook. Open Web Application Security Project, URL: *owasp.org*

Selon son interlocuteur, le Security Coach s'intéresse principalement aux menaces[13] ainsi qu'aux risques ou vulnérabilités qui en découlent. Il assiste les Security Champions au sein des équipes. Ceux-ci sont des membres permanents des équipes DevOps où ils sont en charge de la sécurité. En collaboration avec les ingénieurs au sein de l'équipe, le Security Champion traite les vulnérabilités et les menaces techniques.

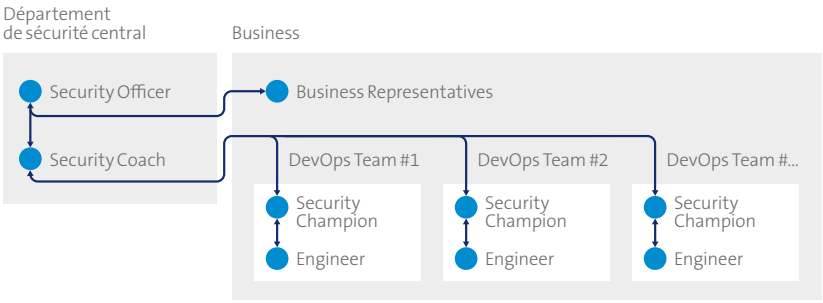

Image 5 | Structure organisationnelle des rôles de sécurité

Conformément à l'approche «Shift Left», la responsabilité pour les vulnérabilités et menaces se concentre directement au sein de l'équipe DevOps. L'organisation de sécurité centrale a un rôle d'assistance et d'habilitation en pilotant le traitement des risques de sécurité.

Comme les organisations DevOps sont très dynamiques, il est souhaitable que les rôles soient gérés dans un registre central. Ainsi, il est toujours possible de savoir clairement qui endosse quel rôle et comment il doit être informé. Dans tous les cas, il faut gérer avec parcimonie la définition de nouveaux rôles, car chaque rôle supplémentaire engendre de nouvelles interfaces et donc un travail de coordination supplémentaire. Afin que l'organisation reste la plus légère possible et puisse réagir rapidement à l'évolution des exigences, l'introduction par étape des rôles de sécurité est pertinente. Par exemple, dans une première étape, les rôles du Security Officer et du Security Coach décrits ci-dessus peuvent être combinés.

> Lorsqu'une transformation DevOps a lieu au sein d'une grande entreprise, cela produit généralement beaucoup d'incertitude (au sens entrepreneurial). Dans un tel environnement, le modèle Security Champions offre à l'organisation de sécurité centrale la possibilité d'intervenir rapidement (en tant que sous-organisation) et de concevoir les structures nécessaires.

[13] Dans cette publication, nous utilisons le mot français «menace», synonyme du terme anglais «threat».

Security Champions
Plusieurs définitions, qui diffèrent quelque peu, existent dans l'industrie en ce qui concerne le rôle du Security Champion. Ce qui est sûr, c'est qu'il doit avant tout être l'interlocuteur au sein de l'équipe pour tout ce qui touche à la sécurité. Dans cette fonction, il favorise la sensibilisation. Il aide à identifier les menaces et les vulnérabilités, les clarifie et s'assure qu'elles soient prises en compte.

Cependant, la responsabilité relative à la sécurité d'un produit ne doit pas être endossée exclusivement par le Security Champion, mais par l'équipe et l'organisation. Il n'est pas prévu que le Champion aborde seul tous les thèmes de sécurité. Il le fait plutôt en échangeant constamment avec son équipe, son Security Coach, mais également d'autres Security Champions (*voir plus bas: Security Community*).
Dans le cadre des échanges avec son coach, le Champion profite surtout de son savoir-faire. En retour, le coach peut se faire une idée de la situation (relative à la sécurité) au sein de l'équipe ou du projet. Ainsi, le Security Champion permet un point de vue consolidé axé sur l'équipe et représentant les besoins concrets.

Dans la pratique, l'expérience chez Swisscom a révélé ce qui suit: pour l'acceptation et donc la réussite d'un modèle Security Champion dans un contexte entrepreneurial, il est important que le Security Champion ne soit pas seulement intégré à l'équipe DevOps, mais utilise également son langage technique. Lorsque différentes équipes travaillent sur un produit commun, la vue d'ensemble relative à la sécurité ne doit pas être délaissée. Autrement, comme chaque équipe développe généralement une partie du produit final, il est facile de perdre la vue globale. Il est donc également important de nominer un Champion au niveau Business. Ce «Champion of Champions» garde surtout un œil sur l'architecture et les interactions des différentes équipes en matière de sécurité.

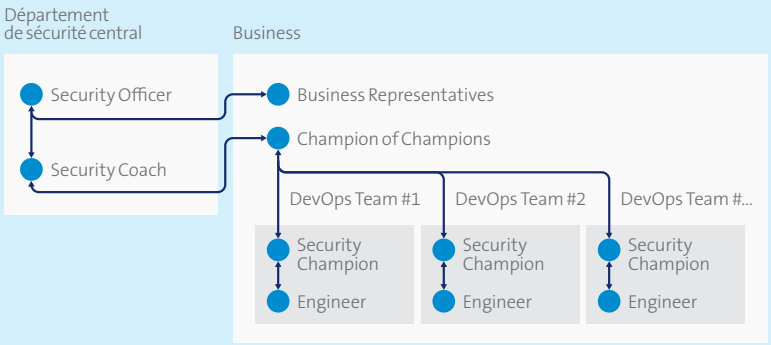

Image 6 | Le Security Champion of Champions garde une vue d'ensemble des produits de plus grande envergure.

Plus l'organisation est grande, moins il est utile que le Security Champion soit un véritable spécialiste de la sécurité. Afin qu'au sein d'une grande entreprise, le savoir-faire soit malgré tout intégré au produit final, le rôle du Champion peut être réparti sur plusieurs personnes au sein d'une équipe de produit.

La nomination de Security Champions peut entraîner des tensions. Cela survient surtout lorsqu'au sein de l'équipe DevOps, personne ne souhaite endosser volontairement ce rôle. Souvent, un membre doit sortir de sa zone de confort ou est poussé vers ce rôle. Dans de tels cas, il faut beaucoup de prudence et de pragmatisme de la part de tous afin que le «Champion malgré lui» se sente bien dans son rôle.

Ce qui est primordial, c'est que la nomination des Security Champions ne suffit pas pour que le travail soit fait, aussi bien pour le Business que pour l'organisation de sécurité centrale. Afin que les Champions puissent remplir leur rôle, il faut s'assurer qu'ils disposent également de toutes les ressources nécessaires (temps, argent, compétences de décision). De plus, il s'agit de garantir que les connaissances nécessaires leur soient transmises afin de leur permettre d'endosser la responsabilité de la sécurité au sein de l'équipe.

Mais il faut être conscient qu'au niveau de l'entreprise, la responsabilité des risques de sécurité et les mesures en découlant est toujours endossée par l'organe qui prend la décision finale pour un produit ou un service. Certes, avec DevOps, une grande partie de la responsabilité et donc également du pouvoir de décision sont délégués aux équipes. Mais l'organisation de sécurité doit accompagner les équipes via le modèle Coaching. Et elle doit disposer des compétences pour intervenir et tirer le frein à main lorsque la sécurité est négligée.

Security Coaches
La mission des Security Coaches est en première ligne de permettre aux Security Champions de faire leur travail.

Ce point ne doit pas être sous-estimé. En effet, si un Security Champion manque de savoir-faire, cela entraîne une charge de travail supplémentaire de la part des Security Coaches. Si les failles ne sont pas identifiées, des zones d'ombre apparaissent concernant la sécurité. Dans le pire des cas, celles-ci peuvent entraîner des failles non identifiées (et donc imprévisibles) en matière de sécurité.

Le Security Coach doit assister par ses conseils le Security Champion durant son travail quotidien. Il doit identifier, regrouper et piloter les synergies entre les besoins des différentes équipes. Pour les thèmes de sécurité critiques, le Coach peut tout à fait approfondir, et si nécessaire également assister au niveau du contenu le Champion et le projet. Il peut par exemple aider le Champion pour les audits de sécurité.
Généralement, le Coach est l'interlocuteur de plusieurs équipes. Ce faisant, il ne se concentre pas sur des spécificités techniques d'implémentation. Il apporte plutôt son aide concernant le regroupement des conditions cadres pertinentes, pour les rapports relatifs aux adaptations de l'architecture et lors de la mise en œuvre d'un processus Threat Modeling utile.

Un Security Coach est particulièrement utile lorsqu'il peut directement renvoyer à des modules de solutions, procédures ou implémentations de modèles précis pour la mise en œuvre des directives. Dans l'idéal, il peut se servir d'un catalogue consultable associant les possibles solutions aux directives. Par ailleurs, le Security Champion peut aussi profiter d'un tel catalogue. S'il est librement accessible aux équipes DevOps, celles-ci peuvent trouver elles-mêmes la meilleure solution qui dans le même temps répond à toutes les exigences de sécurité.
D'autre part, la mission du Security Coach est de rendre visible de manière transparente les risques techniques relatifs à la sécurité qui sont identifiés en concertation avec les Champions. Le cas échéant, il doit s'assurer que toutes les parties prenantes comprennent aussi ces risques.

Security Officer
Selon la taille de l'organisation, plusieurs rôles de sécurité peuvent être introduits. En particulier pour les organisations complexes et imbriquées, il est pertinent d'introduire des niveaux d'abstraction supplémentaires.
Le Security Officer est associé aux parties prenantes Business et bénéficie d'un aperçu global des différents produits et de leurs interactions. Cette vue élargie lui permet de mieux évaluer les risques importants et les interactions entre les différents éléments, équipes et services. Le Security Officer s'intéresse donc principalement les risques. Grâce à sa connaissance du Business, il peut à la fois identifier les risques techniques et commerciaux tout en communiquant avec les organes responsables. Lors de la priorisation de mesures de mitigation, le Security Officer apporte son aide.
Le Security Officer échange régulièrement avec les Coaches dans son champ d'action. Des informations importantes sont partagées et l'orientation stratégique est définie.

Security Community
Lorsque de nouveaux rôles de sécurité décentralisés sont créés, il est recommandé d'associer le savoir-faire des détenteurs du rôle afin de permettre ainsi une «communication matricielle». Cela signifie qu'avec son rôle au sein de l'équipe, le Security Champion est certes unique, mais qu'il a la possibilité d'échanger avec des pairs spécialisés dans le cadre de la Security Community.

Cet échange d'expérience est particulièrement important pour la sécurité. Afin de l'encourager et de l'entretenir, il peut être utile de créer une «guilde de sécurité». Celle-ci permet que les membres de la communauté s'entraident, discutent et élaborent également ensemble des solutions relatives aux exigences en matière de sécurité. L'expérience montre que de nombreux aspects traités par un Security Champion sont aussi pertinents pour d'autres équipes. C'est pourquoi les solutions élaborées peuvent également être réutilisées par d'autres équipes. Lorsque les expériences et solutions sont partagées et développées au sein d'une guilde, cela améliore la maturité de la sécurité de l'ensemble de l'entreprise. Et en retour, cela soulage l'organisation de sécurité centrale.

> Chez Swisscom, la guilde de sécurité a été mise en œuvre de la manière suivante: tous les Security Champions et Coaches se réunissent plusieurs fois par an pour aborder ensemble les problèmes relatifs à la sécurité. Afin d'élargir le savoir-faire, de favoriser les échanges et la motivation, des ateliers, formations et évènements sont régulièrement organisés. Le hackaton annuel «Capture The Flag» [14] en fait par exemple partie. Durant cet évènement, les participants peuvent résoudre des problèmes de sécurité sous forme ludique et à l'aide d'exemples pratiques. La guilde gère un chat qui permet et favorise les échanges.

4.3 La sécurité des activités de planification

«Everything as Code» est l'un des principes centraux de DevOps. Mais il ne faut pas oublier

Eléments essentiels dans ce chapitre		
People	**Process**	**Technology**
Prise en compte des exigences de sécurité Expliquer l'impact d'une planification itérative typique d'un environnement agile sur la sécurité	Mettre à disposition des processus efficaces et applicables de façon itérative pour l'identification de menaces et de risques	Assistance d'outils, lorsque c'est possible, afin de créer des versions des informations, de les partager et de les annoter

que le code ne peut être écrit sans préparation et réflexion approfondie. D'une manière générale, les activités de sécurité doivent être prises en compte dès les phases abstraites «Requirements gathering» et «Architecture & Design» du Software Development Lifecycle.

[14] CTF? WTF?, CTFtime team, URL: *ctftime.org*

4.3.1 Définir les exigences de sécurité

Dans tous les cas, les sources suivantes ont une influence sur le catalogue d'exigences d'un développement de produit:

- **Dispositions légales** relatives aux informations traitées (p. ex. la législation suisse sur les télécommunications ou sur la protection des données)
- La **conformité** qu'il faut respecter (par exemple concernant la circulaire de l'Autorité fédérale de surveillance des marchés financiers (FINMA) ou ISO/IEC 27001)
- La **gestion** des données et informations conformément à la classification des données interne à l'entreprise et aux directives correspondantes ainsi que, selon les cas, les exigences des clients

Dans un contexte d'entreprise, ces sources d'exigences sont critiques pour l'activité. Une infraction peut avoir de lourdes conséquences: amendes pécuniaires et atteinte à la réputation, perte de secrets d'affaires et donc de la base de l'activité.

Une fois que le catalogue d'exigences a été élaboré, il doit être communiqué pour servir de base au produit complet et à son architecture, à l'implémentation et à la mise à disposition de toutes les personnes concernées. Naturellement, il faut aussi s'assurer qu'il soit adapté lorsque l'utilisation du produit est élargie à de nouveaux domaines.

4.3.2 Threat Modeling

Afin de pouvoir protéger de façon adéquate les informations traitées, la solution technique doit être armée contre les attaques. Une modélisation des menaces (Threat Modeling) permet d'identifier des formes d'attaques possibles auxquelles le produit est confronté.

Si la modélisation des menaces se base sur une approche systématique (p. ex. STRIDE[15]), des aspects supplémentaires comme la traçabilité et reproductibilité peuvent être pris en compte. Souvent, il est utile de compléter et d'affiner l'approche fondamentale avec des valeurs empiriques.

> Les auteurs de ce livret pensent que le Threat Modeling est d'une grande utilité pour la sécurité d'un produit. Cependant, sa mise en œuvre systématique renferme de nombreux défis et suppose un degré élevé de sensibilisation à la sécurité de la part de toutes les parties prenantes. En particulier au vu de la diversité des activités dans une grande entreprise, la garantie

[15] STRIDE est un acronyme qui désigne les 6 catégories de menaces Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service et Elevation of Privilege; à l'origine, STRIDE a été créé par des collaborateurs Microsoft.

de la qualité continue de modèles de menaces devient un travail de titan. Par conséquent, un Threat Modeling systématique peut tout à fait être considéré comme une caractéristique de qualité et de maturité d'une organisation.

Résultats d'un Threat Model

Les menaces déterminées ne constituent pas les seuls résultats tangibles d'un Threat Model. Même si elles sont importantes pour la traçabilité, les mesures en découlant ainsi que les scénarios de test et de détection sont plus précieux.

Lors de la sélection de la mitigation à appliquer, les principes suivants s'appliquent par ordre décroissant d'importance:

- Dans la mesure du possible, les menaces doivent être évitées par une **adaptation de l'architecture**.
- Si l'architecture ne peut pas être adaptée, la mitigation technique doit, dans la mesure du possible, être mise en œuvre avec une **solution standard**.
- Lorsqu'aucune solution standard n'est disponible, la **solution autonome** doit être contrôlée au moins par un spécialiste.
- Si une solution technique est également impossible, la vulnérabilité (restante) doit être **documentée** et communiquée en tant que **risque**.

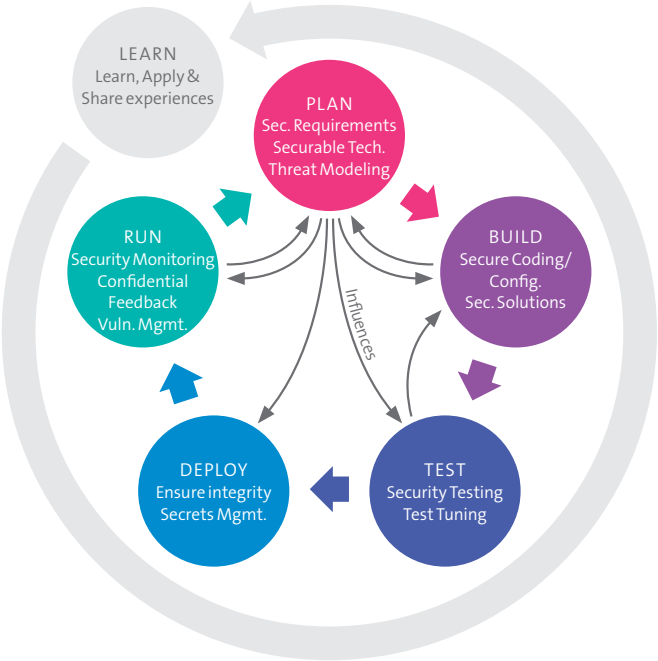

Image 7 | Un Threat Model a un impact sur toutes les autres activités dans un SDLC

Selon la mitigation choisie, des scénarios de test et de détection peuvent être définis. Tous les résultats élaborés dans le cadre du Threat Model ont un impact sur les activités de sécurité durant les phases suivantes du SDLC:

- **Implémentation:** Les mitigations techniques sont mises en œuvre directement ou via l'intégration d'une solution.
- **Test/Intégration:** Les mitigations sont testées automatiquement. Pour cela, des outils et services d'analyse de la sécurité peuvent être utilisés. Lorsque cela est nécessaire, ils s'orientent plus spécifiquement vers le produit grâce aux conclusions du Threat Model.
- **Deployment:** Les menaces qui apparaissent lors de la distribution de logiciels, de la configuration ou bien le traitement d'informations confidentielles comme les mots de passe, les clés privées ou les jetons, peuvent être abordées et surveillées de manière précoce avec une mitigation technique adaptée.
- **Exploitation:** Afin que les attaques soient identifiées pendant la durée de fonctionnement, deux conditions importantes doivent être remplies. Il faut d'une part que les scénarios de détection identifiés soient planifiés et mis en œuvre. D'autre part, il faut s'assurer que le produit regroupe sur une plateforme centrale les informations nécessaires des journaux.

Gestion de la documentation Threat Modeling
En pratique, il existe différentes approches pour la gestion d'une documentation Threat Modeling. Elles se différencient pour l'essentiel par leur mode de gestion:

- Pour un Threat Model **central**, l'ensemble de l'équipe travaille sur la même documentation archivée de manière centralisée. Ici, lors de la mise en œuvre de différentes fonctionnalités, il faut respectivement se référer au Threat Model central.
- Pour un Threat Model **distribué**, les nouvelles menaces et les mitigations correspondantes font directement partie du paquet de travail concerné (ticket, user story, etc.). Par conséquent, elles sont directement spécifiées et mises en œuvre dans ce contexte.

Selon les individus et le niveau, les deux approches ont des avantages et des inconvénients spécifiques. De plus, elles fournissent un aperçu différent.

Un Threat Model central permet une orientation rapide dans le contexte d'un produit. Ainsi, les potentiels de synergies peuvent facilement être identifiés. Cependant, lors de la mise en œuvre de différentes fonctionnalités, «User Stories» ou tâches, un investissement cognitif de la part de l'ingénieur DevOps devient nécessaire. Il doit alors d'abord rechercher les points pertinents à partir de l'ensemble. De plus, un Threat Model isolé et détaché du véritable produit peut rapidement devenir asynchrone par rapport à celui-ci. Par conséquent, son utilisation et son actualisation demandent souvent beaucoup de travail.

Le Threat Model distribué est plus proche de l'ingénieur DevOps et isole la partie pertinente pour la fonctionnalité correspondante. Comme il existe déjà un ticket faisant office de base de travail, la gestion du modèle distribué est plus simple. Ici, à l'opposé, une charge de travail supplémentaire est nécessaire lorsqu'il s'agit de bénéficier d'un aperçu complet des menaces et des mitigations mises en œuvre. Ensuite, il faut tout d'abord regrouper toutes les pièces uniques.

Selon l'aperçu dont l'on souhaite bénéficier, les deux approches peuvent donc entraîner une charge de travail supplémentaire. Afin que l'utilisation, la maintenance du modèle ainsi que les activités ultérieures se déroulent de manière ordonnée, elles doivent être représentées par un processus Threat Modeling simple et approprié.

DevOps et Threat Modeling

Dans le cycle de vie d'un produit, le Threat Modeling est déterminant pour sa sécurité. Cependant, l'intégration correcte au processus de développement est complexe. Ici aussi, il existe différentes approches.

D'une manière ou d'une autre, les menaces doivent toujours être identifiées lorsqu'un élargissement de l'architecture est prévu. Afin de pouvoir effectuer cela, il est nécessaire qu'une personne puisse se mettre à la place d'un attaquant. Cela est compliqué, car il faut passer d'un comportement systématique-constructif (en tant que développeur ou opérateur) au rôle créatif-destructif d'un attaquant. Souvent, lorsque l'équipe de développeurs doit développer elle-même un Threat Model, il s'agit du plus grand défi.
Il est aussi possible de générer les menaces via un outil. Mais souvent, les résultats ne possèdent pas la spécificité nécessaire au contexte du produit. Par conséquent, de nombreuses menaces générées ne sont pas considérées comme applicables par les utilisateurs des outils. Ainsi, les résultats de l'outil sont analysés de manière moins précise et leur effet s'estompe.

Au final, la qualité d'un modèle de menace peut sans doute être améliorée le plus efficacement possible lorsqu'un modélisateur de menaces expérimenté assiste l'équipe de développeurs. Cependant, cette approche engendre souvent des problèmes de flexibilité. En particulier dans le contexte entrepreneurial, les experts de sécurité manquent souvent de compréhension technique ou thématique du produit. Cela entraîne des difficultés qu'il faut gérer.

Pour combler cette faille, un processus Threat Modeling continu et simple est nécessaire. Il doit créer les interfaces nécessaires, notamment pour les étapes suivantes du Product Lifecycle, et garantir que les participants puissent collaborer sans dysfonctionnements. Ainsi, un Threat Modeler expérimenté connaissant mal le produit doit pouvoir assister une équipe qui est peu au fait des menaces de sécurité.

Comme DevOps est un concept itératif, l'architecture d'un produit peut être modifiée avec chaque nouvelle User Story. Il n'est donc pas possible d'élaborer un modèle de menace initial et de le référencer sur une longue période sans qu'il soit entretenu.

Il faut plutôt prêter attention à ce que le modèle puisse être traité en tant qu'ensemble, également sous forme de différentes parties partielles spécifiques. Ce premier point est important pour bénéficier d'un aperçu global de la sécurité et le dernier afin de pouvoir aborder précisément les menaces. En tout cas, l'évolution constante du modèle de menace dans le processus de développement ne rend pas sa gestion plus simple.

4.4 Activités de sécurité techniques

Eléments essentiels dans ce chapitre		
People	**Process**	**Technology**
L'équipe doit connaître et comprendre les domaines d'application des solutions techniques	Les vulnérabilités identifiées doivent être suivies et il faut clarifier qui en prend la responsabilité Un soutien technique est nécessaire pour l'intégration, l'élaboration et la mise à disposition des solutions de sécurité	Les solutions de sécurité doivent être faciles à intégrer

Sous «activités de sécurité», on regroupe les étapes pouvant être mises en œuvre immédiatement d'un point de vue technique ou qui n'influencent pas directement l'implémentation d'un produit.

4.4.1 Security Solutions

Dans le cadre de «Shift Left», ce ne sont pas seulement les thèmes de sécurité (constituant à eux seuls déjà un travail de titan) qui sont transférés à l'équipe DevOps. La stabilité, le Change Management, le monitoring opérationnel, etc. s'y ajoutent aussi. Pour cela, il faut toujours prendre en compte le fait que tout cela s'étire sur l'ensemble du cycle de vie du produit.

Par conséquent, la responsabilité transférée à l'équipe est grande. Si le soutien n'est pas adapté, des erreurs aux lourdes conséquences peuvent facilement survenir. C'est ici que l'organisation de sécurité centrale entre en jeu. En mettant à disposition des solutions de sécurité intégrables, modulables et configurables, elle peut apporter les connaissances spécialisées nécessaires.

Solutions clés en main

Afin que le niveau de sécurité puisse être maintenu ou amélioré dans l'entreprise, un catalogue avec des solutions clés en main doit être mis à disposition des équipes DevOps. Si elles sont adaptées au modèle de menace correspondant, elles peuvent simplement être intégrées au produit. Les intégrations correspondantes peuvent être proposées et développées via Inner Source (une approche Open-Source interne à l'organisation).

Lorsque le catalogue est standardisé, le nombre de solutions de sécurité reste limité. L'avantage, c'est que différentes solutions peuvent être gérées par les spécialistes de sécurité du département de sécurité central. Ils gèrent le portefeuille en fonction des risques de sécurité, mais se basent avant tout sur le feedback des utilisateurs. L'objectif doit être de concevoir des solutions le plus simple et efficace possible.

> Il s'est avéré utile – à partir d'une perspective d'organisation – de développer et d'exploiter des solutions de sécurité applicables de façon générique de manière centrale. Au sein de Swisscom, c'est le DevSecOps Tooling Competence Center qui prend ceci en charge en tant que partie de l'organisation de sécurité centrale. L'objectif d'une telle configuration doit être d'intégrer de la manière la plus automatisée possible les connaissances spécifiques des experts d'une organisation de sécurité dans les Deployment Pipelines du produit.
>
> Pour cela, il s'agit de vérifier que cette équipe prend en charge les tâches DevOps usuelles comme l'assistance et la documentation. En particulier concernant l'offre de solutions de sécurité, l'intégration correcte aux systèmes cibles est déterminante.

Afin que le catalogue de solutions reste clair et accessible, une fois de plus, un modèle de menace peut être utilisé. Il est ainsi possible de référencer directement des solutions de sécurité plutôt que des directives relatives à la mitigation de certaines menaces. Cela constitue une aide pour l'équipe DevOps qui n'est pas obligée de développer ses propres solutions, mais également pour les Security Coaches et les Security Officers. Ils peuvent alors s'assurer que le service de sécurité particulier proposé de manière centralisée est imperméable.

Quelques exemples de solutions de sécurité
La liste des solutions de sécurité pouvant être proposées via un catalogue est longue. Voici quelques exemples qui se basent la méthode STRIDE connue pour l'identification de menaces:

- **Authentification:** authentification (facteurs multiples) en tant que service pouvant être intégré à une application
- **Intégrité:** Public Key Infrastructure (PKI) en tant que service automatique afin de permettre la signature et le cryptage d'artefacts et de données d'utilisation
- **Traçabilité:** solutions de logging permettant un Security Monitoring
- **Confidentialité :** bibliothèques et services pour le cryptage et l'accès aux informations sensibles, comme les clés nécessaires à l'exploitation (mots de passe, SSH Keys, certificats, etc.)
- **Disponibilité :** bibliothèques et services par lesquels les attaques peuvent être identifiées et repoussées au niveau des applications et du réseau
- **Autorisation:** ssolutions de gestion des identités dont l'intégration permet une autorisation simple au niveau des applications

4.4.2 Security Testing automatisé

La vérification automatisée de produits constitue la base de DevOps. Les tests automatisés permettent d'identifier et de résoudre rapidement des erreurs. Les règles sont les suivantes : Plus une faille est identifiée rapidement, plus elle peut être résolue facilement et moins la charge de travail nécessaire est élevée.

La sécurité ne constitue pas une exception. Au contraire, les Deployment Pipelines entièrement automatisées (avec validation de sécurité implicite) rendent visibles les aspects relatifs à la sécurité dès l'implémentation. L'automatisation du Security Testing constitue donc non seulement un contrôle de la qualité, mais également un outil pour améliorer la sensibilisation en matière de sécurité.

Afin que tous les avantages puissent être exploités, quelques tâches préliminaires sont nécessaires. Ainsi, des services Security Testing adaptés aux besoins des équipes DevOps dans le contexte de l'automatisation doivent être mis à disposition. En font partie, notamment :

- l'intégration technique sans accroc de la solution Security Testing dans la Deployment Pipeline
- aucun ralentissement du Build (Test Performance)
- Investissement Onboarding minimal

Ceux qui suivent ces principes peuvent améliorer l'acceptation des services Security Testing. En retour, cela facilite la mise en œuvre des services nécessaires.

> Souvent, des membres d'équipes DevOps accordent moins d'importance à la sécurité. Ils abordent alors souvent les outils de sécurité de manière critique et communiquent de manière correspondante.
>
> Afin d'éviter une telle frustration relative à la sécurité, un service d'assistance adapté est nécessaire. Celui-ci apporte son aide lors de problèmes relatifs à l'intégration respectivement à l'application Security Tooling, et répond aux questions concernant le contenu. Chez Swisscom, c'est de nouveau le DevSecOps Tooling Competence Center qui s'en occupe. Cette équipe multidisciplinaire développe les outils de sécurité et les exploite. Dans le même temps, elle fait office d'interlocuteur pour les utilisateurs et reçoit les feedbacks qui sont ensuite utilisés pour des solutions existantes ou nouvelles.

Exigences relatives aux Security Testing Tools
Les outils réalisant les tests constituent la base de l'automatisation. Lors de leur sélection, il convient d'observer certains aspects fondamentaux :

- **Adéquation avec l'avenir:** En particulier au sein d'une organisation DevOps, les besoins évoluent très vite. Les outils doivent donc être intégrables de manière flexible, et accessibles de plusieurs façons. Ceci est possible lorsque toutes les étapes de travail peuvent être automatisées via API. Dans l'idéal, le Security Tool est développé selon une approche «API-first».
- **Orientation utilisateur:** L'outil doit satisfaire les principaux groupes d'utilisateurs, c'est-à-dire:
  - Les équipes DevOps: leurs technologies de base doivent être soutenues pour répondre à leurs besoins techniques. L'outil doit également permettre le développement technique continu.
  - Les Security Champions et Coaches: ils assistent les équipes afin d'intégrer la sécurité au produit. Par conséquent, l'outil doit offrir la possibilité d'agréger les défauts (c'est-à-dire les failles de sécurité identifiées) de manière à identifier les domaines pour lesquels l'équipe nécessite encore une assistance.
- **Capacité multi-mandants:** En particulier dans les grandes entreprises, il est intéressant d'évaluer les résultats des tests selon les domaines d'activité. Ainsi, cela peut être utile lorsque les domaines sont plus ou moins critiques pour l'organisation. Ou bien cela permet de bénéficier d'un aperçu des domaines pour lesquels il faut encore investir dans la sécurité. Un outil doit assister de telles observations spécifiques.

En particulier dans les environnements axés sur le service, de nombreuses technologies variées sont souvent utilisées au quotidien. Celles-ci sont plus ou moins soutenues sur le marché. Ainsi, la maintenance de langages de programmation obsolètes est souvent

maintenue tandis que dans le même temps, de nouveaux services sont développés avec des technologies tellement actuelles que quasiment personne ne les connaît encore.

Il convient d'aborder cette diversité de différentes manières. Il est possible de mettre à disposition un outil respectivement un service pour chaque technologie ou langage de programmation, ou bien de trouver sur le marché un produit couvrant l'ensemble du spectre. Quoi qu'il en soit, dans l'environnement de l'entreprise, il faut ici procéder selon la règle 80-20 en s'orientant selon la criticité des produits à analyser.

Test Scope
Les Security Tests automatisés permettent d'analyser différents aspects. En pratique, on peut différencier au minimum les cas suivants :

- **Code source** développé soi-même: Lorsqu'un produit est développé de manière autonome, le code source peut être vérifié. Il est ainsi possible d'identifier, de mitiger ou de supprimer directement les vulnérabilités.
- **Third-party Libraries:** Quelque part, quasiment chaque logiciel est basé sur des artefacts logiciels sous la responsabilité de tiers. Ainsi, pour les librairies Open Source, toutes les responsabilités incombent souvent à l'utilisateur. Il doit d'autant plus être informé de toutes les vulnérabilités connues.
- **Infrastructure:** D'une manière ou d'une autre, chaque logiciel est basé sur une infrastructure. La base doit être configurée de manière sûre et tenue à jour. Les vulnérabilités comme les ports de communication trop exposés, les algorithmes de cryptographie peu sûrs ou les Security Headers manquants doivent être évitées.

Types de tests
Certains Security Tests peuvent et doivent fréquemment être réalisés de manière complémentaire. Même si les portées des tests de certains types peuvent se superposer, d'autres vulnérabilités peuvent malgré tout être identifiées. Les types de test les plus répandus sont décrits ci-dessous. En adaptant les paramétrages, ils peuvent être rendu plus spécifiques en soi ou en combinant avec d'autres types de tests selon le produit à tester, en utilisant des informations du Threat Model.

- **Static Application Security Testing** (SAST): Dans le cadre de l'analyse statique, le code source est analysé. Selon la technique, on recherche des mots-clés pouvant engendrer un comportement peu sûr. Les outils d'analyse avancés élaborent, à partir du code source, des modèles graphiques via lesquels il est ensuite possible d'identifier la présence de certains motifs.
  D'une manière générale, les solutions SAST sont efficaces, car elles permettent de cerner en profondeur le corde source. Cependant, leur principal défaut est leur vulnérabilité par rapport aux faux positifs qui doivent souvent être examinés à part. De plus, les outils SAST ne suivent pas toujours le développement continu des langages

de programmation et des frameworks. Comme SAST ne prend pas en compte le contexte d'application, des erreurs logiques ne sont souvent pas identifiées.

- **Dynamic Application Security Testing** (DAST): Durant l'analyse dynamique, le logiciel est exécuté dans un environnement protégé et accessible via des saisies relevant de la sécurité. Les saisies peuvent soit être formatées de manière pertinente soit générées aléatoirement («fuzzing»). Selon le comportement ou les résultats de l'application, des vulnérabilités peuvent être déduites.
  Les outils DAST génèrent moins de faux positifs. Comme le contexte de l'application est pris en compte, ces outils sont mieux adaptés pour identifier les erreurs logiques. Cependant, en raison de l'aperçu limité dans le logiciel, seules des vulnérabilités directement exposées sont identifiées. Naturellement, dans une certaine mesure, les outils peuvent être adaptés selon les applications cibles. Mais il reste difficile de simuler de manière réaliste un attaquant agissant de manière créative.
- **Interactive Application Security Testing** (IAST): Dans IAST, «Interactive» désigne les interactions entre les deux approches déjà présentées. L'objectif est de regrouper les aspects positifs de SAST et DAST. Ainsi, IAST permet par exemple d'évaluer si un certain input est traité dans un flux de données. Cela diminue la part de faux positifs. Cependant, les solutions IAST sont souvent très spécifiques à une technologie.

Les trois concepts font partie des solutions Application-Security dédiées.

> Pendant l'introduction d'une solution SAST en self-service pour les équipes, Swisscom a pu tirer des enseignements précieux: les ingénieurs DevOps, possédant souvent un savoir-faire insuffisant par rapport à SAST et la sécurité, ont quelquefois fait scanner l'ensemble de leur code source (dossiers test, données Fixture incl.). Par conséquent, le scan a nécessité beaucoup de temps (plusieurs heures) et n'a donc pas pu être intégré aux processus de builds automatiques. Ainsi, ces scans ont uniquement été réalisés manuellement et de manière sporadique.
>
> En raison de «l'ajout» illimité de code, une quantité énorme de défauts a été présentée aux utilisateurs. Il s'agissait dans la plupart des cas de faux positifs. L'interprétation des résultats du scan revenait donc à chercher une aiguille dans une botte de foin.
>
> L'acceptation du nouveau service SAST a donc considérablement diminué. En raison des expériences négatives, de nombreux utilisateurs ont développé une véritable aversion contre le service et ont refusé de continuer à l'utiliser.

La principale conclusion, c'est qu'un outil SAST ne peut pas fonctionner sans informations spécifiques relatives à l'objet cible (contexte). Afin de résoudre ce problème, deux points ont été améliorés:

- L'ensemble du processus Onboarding a été reconfiguré.
- Le public cible a été réajusté afin de pouvoir relancer le service sans réticences.

De plus, le service a été lancé sous un autre nom. Cela a permis d'éliminer l'image négative et de reprendre à zéro avec l'outil, ce qui en soi, était bon.

Aujourd'hui, le service est exploité par une équipe SAST spécialisée disposant d'un savoir-faire adapté. Les équipes DevOps qui enregistrent pour la première fois leur produit pour le service SAST doivent parcourir trois étapes lors de l'Onboarding:

- Durant un Threat Model initial, les flux de données sont analysés et les parties de codes non pertinentes sont identifiées.
- Ensuite, l'équipe SAST ajuste spécifiquement le scan au produit afin d'éliminer les faux positifs, d'identifier les faux négatifs et d'optimiser la durée d'exécution.
- Après le premier scan, les défauts trouvés sont discutés dans le cadre d'un Fact Finding Meeting. Les équipes peuvent également y bénéficier d'une assistance facilitant la résolution de défauts et les aidant à les éviter à l'avenir.

Quasiment 95 % des faux positifs ont pu être empêchés avec ces modifications du processus. La durée d'exécution du scan a été réduite de plusieurs heures à seulement quelques minutes.
Avec cette durée courte et la pertinence élevée des résultats, les scans peuvent désormais être automatisés. D'une manière générale, l'acceptation auprès des utilisateurs s'est également significativement améliorée.

Fondamentalement, les aspects relatifs à la sécurité peuvent également être testés sans environnement de test dédié à la sécurité. Ainsi, il est par exemple possible d'effectuer la validation d'input via des **Unit-Tests** automatisés.

Bien entendu, le nombre de vulnérabilités introduites par les Third Party Libraries doit rester aussi faible que possible. Afin d'atteindre ceci, le code peut tout d'abord être examiné avec SAST afin de déterminer les dépendances externes intégrées. Grâce aux résultats obtenus, il est ensuite possible d'identifier les vulnérabilités pertinentes via la corrélation de CPE (Common Platform Enumeration) et CVE (Common Vulnerability Enumeration).

Une autre approche pouvant être utilisée en complément consiste à exécuter le scan seulement à la fin de la Deployment Pipeline sur l'artefact assemblé. Afin d'obtenir des informations au sujet des Third Party Libraries incluses, l'artefact peut être décompressé. Cela fournit un inventaire des composants et versions logicielles qu'il contient. Cet inventaire permet alors de corréler les vulnérabilités identifiées. Afin que des corrélations de vulnérabilités soient également possibles ultérieurement, l'inventaire peut être persisté. Ce processus est traité *au point 4.6.1 sous Security Monitoring*. Il est désigné comme Artefact Vulnerability Management (AVM) ou Software Component Analysis (SCA).

Afin de tester la sécurité de la configuration de l'infrastructure (également appelée «renforcement»), différents outils sont utilisés. nmap [16], nessus [17], sslscan [18] et d'autres en font partie. Les outils peuvent être automatisés un par un, via l'intégration dans les lignes de commande, ou en tant que composants de frameworks de test établis. De tels frameworks peuvent par exemple être orchestrés avec le langage d'abstraction «Gherkin» [19] qui s'est établi dans le testing de bout en bout. Gherkin est une syntaxe qui est utilisée dans le Behaviour Driven Development (BDD). Elle doit permettre également aux non-techniciens d'écrire des cas de test en formulant des scénarios via du texte en prose *(voir image 8: Exemple d'un cas de test dans la syntaxe Gherkin).*

D'une manière générale, lors de cette approche, les cas de test d'infrastructure doivent également pouvoir être déduits directement du modèle de menace ou même générés de manière automatisée. Au sein de la communauté, on parle alors généralement de Security as Code. Cela peut naturellement parfaitement intégrer l'exigence DevOps «Everything as Code».

Il existe plusieurs méthodes automatisées de Security-Testing. Il est important de comprendre que chacune d'entre elles considère l'application sous un autre angle de vue. Une unique méthode indépendante fournit donc un aperçu insuffisant de l'application. Une sécurité optimale est atteinte en combinant différentes méthodes de test automatiques en terminant par un Penetration Testing. On peut alors être certain que la créativité d'un attaquant est également intégrée au résultat.

[16] Nmap de Gordon Fyodor Lyon, *nmap.org*

[17] Nessus (Software), Wikipedia: *de.wikipedia.org/wiki/Nessus_(Software)*

[18] sslscan, rbsec, URL: *github.com/rbsec/sslscan*

[19] Gherkin Syntax, Cucumber Community, URL: *docs.cucumber.io/gherkin*

```
Feature: Google Searching
      As a web surfer, I want to search Google, so that I can
      learn new things.

Scenario: Simple Google search
      Given a web browser is on the Google page
      When the search phrase "panda" is entered
      Then results for "panda" are shown
```

Image 8 | Exemple d'un cas de test dans la syntaxe Gherkin *(source: automationpanda.com)*

4.4.3 Security Testing manuel

De nombreux Security Tests peuvent être automatisés. Cependant, leur qualité dépend de celle de leur maintenance. Les résultats de tests automatisés représentent certes un bon aperçu de la sécurité de base. Cependant, ils ne peuvent pas représenter la créativité d'un attaquant pouvant placer des informations dans un contexte.

Penetration Testing

Par conséquent, les tests automatisés ne peuvent pas remplacer les tests de pénétration techniques périodiques. Peu importe qu'il s'agisse de tests White-Box, Grey-Box ou Black-Box[20] par un personnel spécialisé formé. Dans l'idéal, il faut que le logiciel ait déjà été testé de manière automatisée avant le test de pénétration et que les vulnérabilités identifiées aient été résolues. Cela permet aux testeurs de mieux se concentrer sur les vulnérabilités complexes et d'être moins distraits par les vulnérabilités «simples» qui apparaissent.

Audits coordonnés

Les audits périodiques sont surtout intéressants pour les produits critiques et exposés. Conformément à l'approche DevOps, un environnement dédié peut être déployé pour le test de pénétration.

Il faut fournir un maximum d'informations relatives au produit aux testeurs. Cela leur permet de tester ou d'exclure dès le départ des scénarios d'attaque avancés. L'objectif doit être de bénéficier de résultats de la manière la plus efficace possible (correspond à un test White-Box/Grey-Box). Dans cette configuration, les attitudes défensives n'aident pas à contrôler et à évaluer la qualité du produit de façon réaliste.

[20] Lors des tests White-Box, le testeur reçoit le code source et la configuration d'un produit à tester. Lors des tests Grey-Box, seule une partie des informations est divulguée. Lors des tests Black-Box, le testeur ne dispose d'aucune information antérieure.

Red Teaming
Alors que lors du test de pénétration, il s'agit de trouver les failles de sécurité d'une certaine application, pour le Red Teaming [21], il s'agit d'exploiter une faille de sécurité afin de pénétrer le plus loin possible dans le réseau de l'entreprise sans se faire remarquer. Pour cela, les Red Teams agissent avec des conditions réalistes. Cela signifie qu'elles disposent seulement de peu d'informations internes et qu'elles développent celles-ci dans le cadre de leur action.

La Red Team dans le rôle d'attaquant est l'adversaire du Blue Team dans le rôle de défenseur. Le Blue Team décrit toutes les mesures défensives et réactives qu'une entreprise utilise dans le domaine de la sécurité. Au sens large, en plus des équipes de sécurité participants à la réponse aux incidents de sécurité, les équipes DevOps font donc aussi partie du Blue Team.

En soi, le concept du Red Teaming reste inchangé dans une organisation DevOps. Les différences résident surtout dans le temps de réaction potentiel et les conséquences administratives d'une attaque. Comme pour DevOps, un produit est à la fois développé et exploité par la même équipe, des anomalies lors de la circulation de données peuvent par exemple être abordées de manière beaucoup plus rapide et simple *(voir chapitre 4.6)*.

Les publications Red Team finales sont souvent des «War Stories» divertissantes et faciles à lire qui contiennent cependant un fort contenu technique. Elles permettent aux équipes DevOps d'établir immédiatement un rapport et d'élaborer des comparaisons avec le propre produit. En ce qui concerne la Security Awareness, une telle préparation est donc très efficace et précieuse. Naturellement, des mesures systématiques peuvent aussi être déduites des actions Red Team.

> Swisscom utilise le Red Teaming depuis 2015. Chaque année, une à deux simulations d'attaque sont effectuées avec une équipe changeant à chaque fois. L'éventail des utilisations va du Social Engineering au Phishing, en passant par les prises de contrôle de systèmes Swisscom via des malware. Tout ceci se déroule naturellement sans provoquer de vrais dommages. L'objectif des attaques est d'une part d'identifier les failles de sécurité et les risques avant que de véritables hackers les exploitent. D'autre part, le Blue Team (Swisscom SOC, CSIRT et l'exploitation) doit être confrontée à des scénarios réalistes afin d'identifier les faiblesses des compétences et des processus.

[21] Red team, Wikipedia: URL: *en.wikipedia.org/wiki/Red_team*

Il s'est avéré que le principal défi est de sortir les collaborateurs actifs au sein du Red Team de leurs activités quotidiennes afin qu'ils puissent prendre le temps nécessaire pour hacker. Comme le Blue Team développe également ses compétences de détection et apprend de nouvelles choses, avec le temps, la tâche du Red Team devient de plus en plus compliquée. Comme décrit, la préparation et la communication internes des Red Team Cases ont été très précieuses. Les chroniques et rapports préparés touchent concrètement les collaborateurs de tous les domaines d'activité et à tous les niveaux. En retour, cela entraîne des adaptations permettant d'améliorer la sécurité dans leur environnement.

4.5 Deployment Pipeline Security

Eléments essentiels dans ce chapitre		
People	**Process**	**Technology**
L'équipe connaît la situation de menace relative aux systèmes centraux Deployment Pipeline (c'est-à-dire la «fabrique») Il existe une conscience des risques resp. des propres actifs et de ceux des équipes partenaires utilisant la même plateforme	Il existe une séparation des pouvoirs selon le principe du besoin de connaître (need to know)	La Deployment Pipeline assiste l'extension modulaire avec des composants pour contrôler la qualité Les systèmes centraux sont sécurisés de manière adéquate

Etant donné que l'une des compétences clés des organisations DevOps est l'automatisation, l'intégrité et la confidentialité des services d'automatisation doivent toujours être garanties. Les systèmes suivants font partie des services d'automatisation:

- **Workplace** – le système où le code est écrit
- **Version Control System (VCS)** – c'est ici que le code est conservé et géré
- **Continuous Integration/Continuous Deployment (CI/CD) System** – c'est ici que le code est diffusé
- **Scanning Services** – servent à tester le code et les artefacts
- **Archives d'artefacts** – c'est ici que les données pour les packs logiciels ou autres sont stockées
- **Environnements d'exécution** – nécessaires à la production de valeur via le produit logiciel

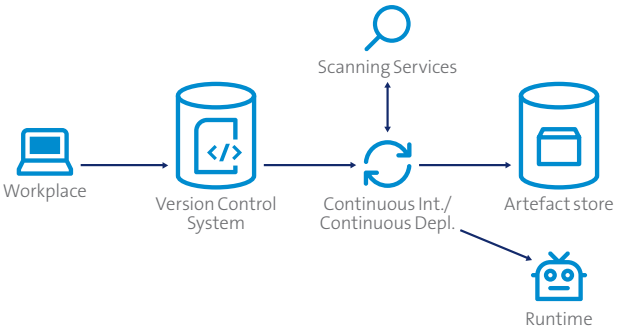

Image 9 | Représentation abstraite d'une Deployment Pipeline avec les différents composants à sécuriser.

Dans ce chapitre, nous aborderons surtout les interactions avec les partenaires externes. Celles-ci sont importantes, car le rôle de plus en plus central de la Deployment Pipeline permet des interfaces avec les systèmes des partenaires et donc de nouveaux vecteurs d'attaque.

Le principe DevOps nécessite que le logiciel et sa configuration soient stockés séparément. Autrement dit : en plus du logiciel, la configuration doit également être intégrée à l'environnement de production. Comme les configurations contiennent souvent aussi des paramètres utilisés pour l'échange de données avec des systèmes tiers, cette communication doit également être sécurisée. Dans la plupart des cas, cela s'effectue grâce à des clés d'authentification. De nouvelles exigences apparaissent ici avec DevOps.

L'influence de «Everything as Code»
Pour les organisations DevOps, la devise «Everything as Code» signifie que tous les tests, Builds, archives et Deployments sont décrits par du code. Cela a des conséquences énormes : ainsi, une adaptation par mégarde au mauvais endroit peut faire échouer les tests. Dans le pire des cas, une modification malveillante volontaire du code peut révéler des clés. Rien que l'accès en lecture à la base du code peut fournir aux attaquant une vue d'ensemble du produit exploité. Cela facilite considérablement leur recherche de failles de sécurité.

Avec la concentration de la logique dans le code, les segmentations sont supprimées et par conséquent aussi les mécanismes de contrôle manuels. Durant une phase de transformation vers DevOps, il s'agit de remplacer en particulier les contrôles de qualité par des mesures automatisées. Celles-ci sont ensuite archivées sous forme de code, de manière adaptée au produit. Cela crée une sorte de redondance logique préservant l'intégrité.

A la suite de la traduction de (presque) tous les aspects en code, la valeur respectivement la criticité de l'archivage du code, c'est-à-dire du Version Control System (VCS) est simplifiée. Si ce système est compromis, cela a des conséquences importantes.

L'archivage des artefacts constitue un autre élément critique. Ici, les défis sont comparables à ceux du VCS. Si des artefacts sont modifiés voire que des artefacts compromis sont introduits, des plateformes de production entières peuvent se retrouver en danger.

Pour les deux systèmes, la préservation de l'intégrité des actifs gérés constitue la priorité. Des modifications non souhaitées peuvent entraîner des problèmes fatals dans les environnements de production. La confidentialité constitue le deuxième point le plus important. La fuite de données d'un VCS ou d'un archivage d'artefacts peut entraîner la perte de secrets et de documents internes.

Naturellement, dans le contexte «Everything as Code», il existe également d'autres menaces. Cependant, celles-ci sont plutôt spécifiques aux technologies ou processus respectivement utilisés.

Intégrité et confidentialité du code et des artefacts
Afin de protéger l'intégrité du code et des artefacts, il faut que la signature des Commits soit obligatoire pour le VCS basé Git [22] le VCS basé Git. D'un point de vue technique, les Commits avec des signatures inconnues doivent être refusés. Il s'agit du seul moyen de garantir que seuls des utilisateurs autorisés peuvent modifier le code.

Pour les artefacts, l'intégrité doit pouvoir être constatée avec au moins une somme de contrôle, c'est-à-dire une somme transversale sur l'artefact complet. Mais dans l'idéal, une signature a également lieu pour les artefacts.

Concernant la préservation de la confidentialité de documents internes et de secrets, les utilisateurs sont donc aussi concernés que la technique. Ainsi, il s'agit par exemple de gérer les accès avec un Identity & Access Management (IAM) adapté. Cela garantit qu'il est uniquement possible d'accéder aux actifs qui sont nécessaires. Les accès excessifs doivent pouvoir être identifiés via le monitoring et freinés via Throttling/Rate Limiting. L'IP-Blocking ou l'Account-Blocking peuvent constituer des contre-mesures afin d'empêcher de possibles attaques *(plus d'informations à ce sujet au chapitre 4.6)*.

Les auteurs du code ou des spécifications systèmes sont également mis à contribution. Une directive Clean-Coding mise en œuvre de manière systématique peut diminuer le risque d'une fuite de codes d'accès (certificats, clés, mots de passe, etc.). Les codes d'accès n'ont rien à faire dans un VCS, car même après leur suppression du code final, ils peuvent encore être visibles dans les anciennes versions. La même chose est valable pour les secrets dans les artefacts.

[22] Git s'est majoritairement imposé en tant que technologie de gestion des versions, Git, Git Community, URL: *git-scm.com*

Les algorithmes doivent aussi être protégés dans la mesure où il s'agit d'innovations exploitables d'un point de vue commercial. Ils doivent être considérés comme des actifs. Ici, des VCS entièrement séparés sont souvent justifiés.

Collaboration avec les partenaires et les fournisseurs
Les données ou systèmes produisant de la valeur sont avant tout précieux pour leur détenteur. Comme les partenaires et les fournisseurs sont moins concernés par ces actifs, ils ne sont souvent pas particulièrement désireux de mettre en œuvre des mesures de sécurité supplémentaires. Des mesures de sécurité convenues contractuellement avec les partenaires sont d'autant plus nécessaires. En se basant dessus, il est par exemple possible d'auditer ou de contrôler des processus pour vérifier si les accords sont respectés.

Comme les changements de personnel surviennent assez souvent et de manière inattendue, le processus IAM utilisé doit permettre un Onboarding rapide. «L'Offboarding» ne doit pas non plus être délaissé en garantissant par exemple que si nécessaire, il soit possible de supprimer rapidement les accès de différents collaborateurs.
Dans un contexte DevOps, il est très avantageux que les Good Practices appliquées soient contrôlées d'un point de vue technique et de manière entièrement automatisée. Il est ainsi possible de refuser les Commits non signés des fournisseurs. La même chose s'applique lorsque les fournisseurs envoient des Commits de collaborateurs qui n'ont pas été inscrits chez le client. Si l'on pousse le concept un peu plus loin, les défauts de sécurité automatiquement identifiés peuvent être renvoyés directement aux partenaires.
D'une manière générale, les exemples montrent que la collaboration avec les partenaires et fournisseurs doit être bien pensée. Les processus doivent permettre de corriger rapidement les erreurs. Les interlocuteurs des deux côtés doivent pouvoir discuter à ce sujet au niveau technique.

Accès aux données et aux systèmes productifs
L'objectif ultime d'une organisation DevOps devrait être de pouvoir représenter des infrastructures complètes sous forme de code. Ainsi, les adaptations sont finalisées durant la durée de fonctionnement. L'accès aux systèmes productifs est alors uniquement possible dans des cas exceptionnels, par exemple afin de pouvoir comprendre des situations et évènements spéciaux. En fait, de tels accès ne seraient pas nécessaires, car les cas problématiques pourraient être tracés et évalués via un logging propre ou auau moins être reproduits dans un environnement non productif.

Pour des raisons de traçabilité, les actions durant l'accès peuvent être enregistrées. Dans l'idéal, un nœud système serait directement remplacé par une nouvelle instance après des accès manuels.

Du point de vue de la sécurité, l'utilité d'une approche « Infrastructure as Code» est grande, comme le démontrent certains aspects:

- Le **renforcement du système** (configuration sûre) peut être contrôlé via des analyses du code
- Il est possible d'effectuer des **adaptations** univoques sur le renforcement du système
- Les **instanciations** d'environnements test dédiés, identiques à l'environnement de production, sont possibles sans charge de travail supplémentaire, par exemple pour les tests de pénétration
- Les systèmes ayant été affaiblis par des actions manuelles peuvent être **remplacés** immédiatement (aussi: «Immutable Infrastructure»)
- Les **données de production** sont mieux protégées, car en règle générale, aucun accès manuel aux systèmes productifs n'a lieu

Un IAM fonctionnant correctement constitue également une condition pour la protection de données et de systèmes. Si des identités ne peuvent pas être assignées de manière fiable, les accès ne peuvent pas non plus être attribués de manière fiable à des personnes. Ainsi, la traçabilité des actions ne peut plus être garantie. Un autre défi apparaît lorsque des utilisateurs qui ne sont pas des humains mais des comptes techniques accèdent à des services.

Gestion de secrets et de certificats
La question de la gestion automatisée de clés qui sont nécessaires à l'accès entre différents systèmes (p. ex. bases de données ou autres services) est inévitablement liée à DevOps. Depuis toujours, le remplacement occasionnel des clés comme les mots de passe correspond à une Best Practice. Cela réduit le risque que l'attaquant puisse avoir accès à un ancien mot de passe afin de visualiser ou même de modifier des données de manière non autorisée et anonyme.

Au sein d'une organisation aux dimensions d'entreprise, il existe de nombreux systèmes dont l'accès a lieu via des processus automatisés. Lorsque les administrateurs système doivent modifier manuellement de manière périodique tous les mots de passe, cela nécessite une importante charge de travail. La modification manuelle des mots de passe peut également entraîner des problèmes d'exploitation lorsqu'un système utilise encore un ancien mot de passe et ne dispose soudain plus de l'accès nécessaire. Cela peut par exemple survenir lorsqu'un mot de passe est ancré dans le code source. D'autres problèmes en découlent aussi. Il faut par exemple savoir comment procéder lorsqu'un administrateur système quitte l'entreprise et qu'il connaît probablement les mots de passe utilisés.

Un autre point en faveur de la gestion automatique des secrets, c'est le cycle de vie technique raccourci des produits au sein de l'environnement DevOps. Au plus tard pour cette raison, les solutions manuelles deviennent définitivement impraticables. Les secrets doivent ensuite faire l'objet d'une rotation, indépendamment de la version du produit, et être intégrés au produit. C'est exactement la même chose pour les certificats (en ligne) et leur contrepartie respective, la clé privée.

Ceux qui abordent ces thèmes avec un degré d'automatisation adapté à DevOps et souhaitent malgré tout garantir un niveau de sécurité approprié trouveront différentes solutions de gestion des secrets sur le marché. Celles-ci sont disponibles aussi bien sous forme de produits commerciaux que Open Source.

Pour les certificats, il faut intégrer une Public Key Infrastructure (PKI) adaptée aux besoins d'automatisation. Pour cela, il faut décider si un service gratuit est utilisé ou si l'organisation doit développer sa propre PKI. Dans ce cas, l'automatisation peut être abordée avec le standard de facto «Automated Certificate Management Environment (ACME)».

4.6 Exploitation productive et Attack Response

Eléments essentiels dans ce chapitre		
People	**Process**	**Technology**
Communication et préparation des scénarios de menace	Gérer le monitoring et les processus Response Permettre une organisation qui garantit la capacité de réaction	Utilisation de solutions pour l'identification d'attaques

Les incidents de sécurité peuvent survenir, avec ou sans DevOps. Cette vérité désagréable permet d'observer d'un autre œil toutes les mesures de sécurité préventives. Elle montre aussi qu'il n'existe pas seulement un côté (préventif) ou l'autre (réactif) en matière de sécurité. Les défis s'étendent plutôt sur l'ensemble du cycle de vie du produit.

Dans les organisations de grande taille, il existe toujours des domaines sur lesquels on se focalise moins en matière d'implémentation de mesures de sécurité. Ceux-ci doivent respectivement agir avec moins de soutien que d'autres services. Vu comme cela, il est important de disposer d'interfaces et de procédures simples pour collecter les logs et feedbacks pertinents pour la sécurité. Le traitement respectivement l'évaluation de don-

nées et de feedbacks doit également être le plus simple possible pour l'équipe DevOps et se dérouler de manière compréhensible.

Dans un cycle de développement logiciel efficace, tous les types de feedbacks doivent être réduits à un message clé pour pouvoir être transférés à la partie préventive du SDLC. Ainsi, les indications relatives aux menaces non prises en compte peuvent être saisies dans un catalogue des menaces accessible durant les activités de planification du Threat Modeling. Le catalogue des menaces et le modèle des menaces sont donc élargis de manière itérative afin de bénéficier d'un aperçu le plus complet possible.

Feedback des utilisateurs
L'un des principes de DevOps est de recueillir des feedbacks implicites (mesurables) et explicites des utilisateurs. Pour cela, le feedback relatif à la sécurité d'un produit doit généralement être traité de façon confidentielle. Il doit uniquement être accessible à un cercle limité de personnes. Autrement, il y a un risque que de potentiels attaquants reçoivent des informations relatives à des failles de sécurité existantes et puissent nuire à l'entreprise.

Afin d'éviter que des évaluations de sécurité puissent être lues sur des forums ou des médias sociaux, il s'agit de créer des canaux légaux et confidentiels via lesquels les failles de sécurité peuvent être signalées. De plus, il faut développer des catalyseurs afin de pousser les chercheurs externes en sécurité à transmettre leur feedback. Des catalyseurs sont habituellement également nécessaires afin de garantir la qualité du feedback.

Vulnerability Management
En pratique, il est possible que durant l'utilisation productive d'une solution, de nouvelles vulnérabilités soient publiées pour certains de ses composants. Les informations correspondantes peuvent être fournies en interne ou en externe de l'organisation *(voir 4.4.2)*, par exemple par la Common Vulnerability Enumeration (CVE). Afin de pouvoir réagir rapidement, il s'agit de surveiller continuellement de telles sources. Le déroulement général lors du traitement de vulnérabilités est décrit de manière formelle par le standard ISO 30111 «Vulnerability Handling Processes».

Ici, DevOps permet de nouvelles approches. Il permet d'une part de remplacer les composants concernés, car l'ensemble de la structure de l'application est décrite par du code source. D'autre part, des tests entièrement automatisés aident à décider rapidement si la fonctionnalité d'un produit est encore garantie. Afin que l'ensemble du processus se déroule sans accroc, un inventaire précis doit être disponible afin d'en déduire les instances du produit concernées par une vulnérabilité.

> La combinaison d'approches centralisées et décentralisées constitue une possibilité de traiter les failles de sécurité: d'une manière générale, les équipes DevOps sont responsables des failles et de leur résolution (traitement décentralisé). Pour cela, elles sont assistées par l'organisation de sécurité centrale. Dans le cadre d'une structure de type force opérationnelle («taskforce»), l'organisation de sécurité centrale offre son assistance pour la résolution de failles critiques et permet ainsi des temps de réaction courts.

Bug Bounty

Les programmes Bug Bounty constituent une mesure de plus en plus appréciée afin de recevoir des feedbacks relatifs à la sécurité. Ils associent le canal de feedback confidentiel au Vulnerability Management en créant une source d'information relative aux vulnérabilités. Pour cela, ils mettent en œuvre le standard ISO 29174 «Vulnerability Disclosure».

Grosso modo, les programmes Bug Bounty fonctionnent ainsi[23]: un utilisateur signale une faille de sécurité du produit à l'entreprise avec une documentation pertinente. Si la faille est confirmée, il reçoit une récompense dans la mesure où il s'engage à la confidentialité pendant une certaine période. L'entreprise utilise ce délai pour résoudre la faille de sécurité. Dans les cercles spécialisés, ce procédé est également connu sous le terme de «Responsible Disclosure Model».

En raison de l'agilité sous-jacente, des défis pertinents en matière de sécurité apparaissent continuellement dans les organisations DevOps de grandes entreprises. C'est pourquoi dans l'idéal, le programme Bug Bounty est géré en un point central. Cependant, la condition est que les exploitants connaissent l'ensemble de la surface d'attaque de l'organisation. Celle-ci se déduit de l'addition des produits exposés.

Afin que les messages Bug Bounty puissent être traités efficacement, un dossier le plus complet et actuel possible avec les indications nécessaires relatives aux produits et personnes est indispensable *(voir aussi: «Alarme – et après?» sous 4.6.1)*. Cependant, garder un tel dossier à jour demande du travail, car les produits, responsabilités et interlocuteurs évoluent constamment. De plus, il est recommandé de définir précisément le domaine de validité technique du programme Bug Bounty et de le documenter pour le grand public. Dans une telle documentation, il est alors indiqué en détail quels services et applications une entreprise expose sur Internet. Les chercheurs en sécurité peuvent uniquement chercher des vulnérabilités ici.

[23] Les conditions des programmes Bug Bounty peuvent varier selon les entreprises. La définition présentée ici est conforme à la compréhension générale.

Afin que l'ensemble de la surface d'attaque soit couverte, il faut également élargir la focalisation sur les acteurs au sein de l'entreprise. Pour cela, les organisations matures mettent à disposition des utilisateurs internes un canal de communication confidentiel pour les feedbacks relatifs à la sécurité.

> En Suisse, Swisscom a été l'une des premières entreprises à proposer un programme [24] Bug Bounty. Depuis son introduction en 2015, le canal a parfaitement fait ses preuves dans de nombreux cas. Les failles de sécurité pertinentes pour le grand public sont publiées sur le portail en tant que «Security Advisories» et se voient attribuer un numéro CVE (Common Vulnerability Enumeration [25]) pour pouvoir être identifiées de manière unique.
>
> Lors de la mise en place du programme Big Bounty, Swisscom a d'abord dû effectuer un apprentissage. Il a par exemple été constaté que les chercheurs en sécurité devaient être encadrés par des représentants spécialisés de l'organisation de sécurité interne. Cela favorise une bonne communication et permet des réactions adéquates aux messages reçus. Depuis, la relation entre les chercheurs en sécurité et Swisscom s'est tellement développée que des apparitions publiques communes sont désormais organisées.

4.6.1 Security Monitoring

On désigne généralement sous Security Monitoring la surveillance d'évènements relatifs à la sécurité. Ce faisant, on oublie souvent que pour une évaluation pertinente et spécifique au produit, des spécifications de Use Case ou Abuse Case sont nécessaires. Celles-ci décrivent les dysfonctionnements en cas d'évènement grave ainsi que les informations nécessaires à la détection d'une attaque. Les données de journal peuvent certes être analysées de manière générique, mais elles fournissent alors souvent trop peu de résultats positifs ou même des faux positifs.

L'écosystème dans lequel évolue une entreprise ou un produit constitue un autre composant à surveiller. Mais pour cela, il faut prendre en compte différents niveaux d'abstraction.

Pour un monitoring efficace au sein d'une organisation DevOps, il faut aussi que tous les interlocuteurs soient définis à l'avance en cas d'incident de sécurité. Si c'est le cas, la configuration DevOps peut faire jouer ses avantages: comme les canaux de communication au sein des équipes sont courts, les mitigations réactives peuvent être mises en œuvre de manière efficace.

[24] Bug Bounty, Swisscom SA,URL: *swisscom.ch*

[25] Common Vulnerabilities and Exposures, The MITRE Corporation, URL: *cve.mitre.org*

Comme l'orientation est difficile, en particulier lors de situations de stress, c'est-à-dire les cas d'urgence ou les crises, tous les scénarios et plans d'urgence nécessitent un entraînement préalable. Ainsi, les failles ou les erreurs au niveau de la chorégraphie des parties prenantes (équipe DevOps, équipes de sécurité, etc.) apparaissent.

Il est dans la nature du Security Monitoring que les incidents deviennent uniquement visibles après leur apparition. Par conséquent, une organisation DevSec-Ops efficace accorde de l'importance à une bonne communication entre les acteurs réactifs et préventifs. Cela permet de matérialiser rapidement les conclusions des incidents en mesures préventives.

Détection des risques connus
L'une des sources pour les Security Monitoring Use Cases est le Threat Model relatif au produit. Il est possible de formuler aussi bien des menaces mitigées que non mitigées sous forme de Monitoring Use Cases. Cela permet d'une part la validation du Threat Model et d'autre part l'identification d'évènements relatifs à la sécurité. Ces derniers doivent être considérés comme plus importants, car sous certaines conditions, un évènement devient une alerte puis un incident.

Afin de pouvoir effectuer des évaluations efficaces, il faut «calculer à l'envers» à partir du résultat souhaité, c'est-à-dire les Monitoring Use Cases. Cela amène un certain nombre de questions:

- Quels composants de l'infrastructure produit doivent fournir les données? En font partie: l'Intrusion Detection System (IDS), Reverse Proxy y compris Load Balancer, Web Application Firewall (WAF), mais aussi l'infrastructure située sous l'application, l'archivage de données, etc.
- Quelles données sont nécessaires? Est-il possible d'y accéder ou est-ce que des composants gérés de manière centralisée sont utilisés?
- Quelles données n'apportent aucune valeur ajoutée? Un «rapport signal-bruit» pertinent peut favoriser la qualité du résultat final.

Du point de vue du Security Monitoring, il n'est généralement pas utile de collecter les journaux les plus détaillés possibles de tous les systèmes. Il vaut mieux sauvegarder un peu plus de données de journal que nécessaire. Un bon début consiste à écrire à l'aide d'un identificateur commun des journaux qui permettent de montrer les transactions via tous les composants d'un système. L'identificateur peut par exemple être un chiffre aléatoire généré par le premier composant qui «voit» l'accès.

Détection d'anomalies
Des potentiels de synergie passionnants apparaissent en raison de l'approche Shift-Left, en particulier en ce qui concerne la sécurité. Ainsi, des solutions de sécurité intrusives comme Runtime Application Self Protection (RASP) Frameworks peuvent tout à coup

devenir intéressantes alors qu'auparavant, elles ne pouvaient pas être utilisées efficacement en raison d'obstacles personnels ou organisationnels.

Même les frameworks d'auto-défense sont fréquents dans l'environnement web. Ceux-ci doivent y détecter et empêcher de façon autonome les attaques en raison d'attributs évoluant de manière atypique. Ainsi, une application peut donner l'alarme lorsqu'elle constate des accès sur un «Honey-Endpoint» [26] Dans un autre cas, cela peut être suspect lorsque l'identification de l'agent change soudainement durant une session en cours.

Naturellement, la configuration détaillée de telles solutions RASP dépend en grande partie du code de l'application. Cela vaut donc le coup d'accorder une grande importance à leur intégration technique complète et aux tests automatisés correspondants. Les solutions RASP peuvent aussi être utilisées en tant que source supplémentaire pour le Security Monitoring.

Alarme. Et après?
Lorsqu'un Security Incident (surveillé et confirmé) survient, dans l'idéal, les contre-mesures implémentées auparavant sont immédiatement et automatiquement déclenchées. Pour les produits critiques pour le bon fonctionnement de l'entreprise, une telle automatisation atteint cependant vite ses limites en raison des faux positifs ne pouvant être exclus. Il est alors utile que des contre-mesures manuelles et/ou techniques puissent être prises grâce à la proximité avec l'équipe DevOps.

Une possible contre-mesure manuelle consiste à adapter le Web Application Firewall (WAF) de manière à ce qu'il autorise un «Hot Patching». Ainsi, le cas échéant, il est possible de sécuriser une vulnérabilité via WAF et l'équipe DevOps bénéficie du temps nécessaire afin de corriger proprement le produit.

Une contre-mesure technique serait par exemple l'intégration d'une attaque dans le Test-Set d'un produit. Il est ainsi possible de vérifier en continu si la vulnérabilité (re)apparaît. Sans de tels tests de régression, la sensibilisation atteinte dans le contexte d'un incident s'estompe généralement vite.

[26] Honey-Endpoint: un Endpoint auquel il n'est pas possible d'accéder durant le fonctionnement normal et dont le but est d'identifier les attaquants qui analysent l'ensemble de la surface d'attaque d'une application Web.

Afin qu'en cas d'évènement grave, toutes les informations pour de telles contre-mesures soient connues, un inventaire systématique est nécessaire au sein de l'organisation DevOps. L'inventaire doit contenir les informations suivantes qui sont en particulier toujours à disposition du Security Operations Center (SOC):

- Liste des membres des équipes DevOps (interlocuteurs techniques)
- Responsabilités des produits (interlocuteurs du Business)
- Technologies (peut contribuer à une atténuation plus rapide)
- Inventaire produit (Endpoints, résultats test, etc.)
- Autres sources d'information (VCS, Threat Model, etc.)

Si une analyse de routine devient un incident, le Computer Security Incident Response Team (CSIRT) intervient. Il profite du fait que d'une manière générale, DevOps soutient les recherches complémentaires criminalistiques relatives aux causes de l'incident et aux dommages causés. Grâce à l'automatisation élevée, il est toujours possible de savoir où, quoi et quand des modifications ont été réalisées sur le produit, l'exposant et permettant l'attaque.

Afin que dans l'agitation, les participants ne détruisent pas de données qui auraient pu fournir des informations sur l'attaque, ils doivent être informés et formés dès le départ. D'une manière générale, SOC et CSIRT sont mis à disposition en tant que services centraux internes à l'organisation. Ils complètent et assistent les équipes DevOps en particulier via des mesures Response professionnels.

> En particulier le programme Security Champions de Swisscom a fait ses preuves lors du traitement de Security Incidents. D'une part, il permet que lors d'un évènement, l'interlocuteur de l'équipe DevOps soit connu dès le départ. D'autre part, grâce aux unités Training & Awareness qu'il a effectuées, le Security Champion possède déjà une compréhension de base concernant la terminologie relative à la sécurité. Cela permet de diminuer les frictions durant les interactions entre CSIRT et l'équipe DevOps et facilite aussi la communication générale.

5 Résumé

Une sécurité efficace au sein d'une organisation DevOps se distingue avant tout par la forte intégration des différentes activités relatives à la sécurité. Plus les différentes étapes sont reliées, plus la sécurité sera efficace. En tant qu'approche, DevOps aide à développer les conditions nécessaires. DevOps supprime les barrières organisationnelles (par exemple entre le développement et l'exploitation) et regroupe ce qui doit l'être. Cela favorise un aperçu global sur l'ensemble du cycle de vie du produit.

Cependant, le défi consiste à prioriser suffisamment la sécurité en tant que domaine thématique. Comme les équipes DevOps n'atteindront jamais la même expertise que les professionnels de la sécurité, l'organisation de sécurité centrale apportant son assistance reste importante pour réussir. Sa devise doit être: «Make it easy to do the right thing». Dans la pratique, elle doit proposer des contenus, processus et services de manière à ce que la variante la plus simple soit aussi la plus sûre.

Ceux qui ne définissent pas clairement les priorités dans le cadre d'une transformation DevOps risquent d'accorder moins d'attention à la sécurité. Afin d'éviter ceci, il faut aborder activement les points suivants:

- **Sensibilisation générale:** avec un programme de formation conférant la visibilité nécessaire à la sécurité.
- **Sécurité intrinsèque:** de manière à ce que la sécurité soit conçue de la manière la plus simple et accessible possible, par exemple avec un service modulaire de solutions pouvant être utilisées telles quelles → La suppression de Tollgates de processus peut être compensée par la création continue de transparence.
- **Identity & Access Management:** représentation de tous les rôles DevOps afin de pouvoir garantir la séparation des autorisations et la pertinence des accès → Il n'y a pas de séparation classique des pouvoirs, mais la valeur d'IAM en tant qu'autorité des autorisations d'accès affinées augmente.
- **Organisation:** la sécurité doit être ancrée au sein de l'organisation avec tous ses aspects et rôles. Afin qu'elle soit flexible, les responsabilités doivent être assignées en largeur, c'est-à-dire aux équipes. → Ce faisant, il faut cependant toujours savoir qui est responsable des risques et vulnérabilités, et les personnes appropriées doivent pouvoir être trouvées et contactées à tous les niveaux.

Si les mesures adaptées sont prises, DevOps et la sécurité ne sont donc nullement incompatibles mais entrouvrent mutuellement de nouvelles possibilités. Il est ainsi possible de rendre le monde, d'un point de vue technique, un peu plus sûr et prévisible.

Swisscom s'efforce depuis mi-2016 à implémenter et appliquer ces Best Practices. Comme c'est le cas dans les environnements agiles, il s'agit d'un processus en constante évolution. Conformément à la devise «Do, Learn and Adapt».

Jusqu'à présent, les défis apparaissent surtout dans les domaines de l'automatisation et de la culture. Pour l'avenir, il est prévu d'améliorer la maturité des trois piliers People, Processes & Technology afin de réduire la différence de niveau entre les équipes précurseurs et les retardataires.

Il s'agit aussi d'améliorer la mesurabilité dans tous les domaines afin de pouvoir ensuite prendre des décisions basées sur les données. Le regroupement de chiffres relatifs aux vulnérabilités détectées et aux niveaux de certification sécurité des membres d'une équipe constitue un exemple. En parallèle, les données doivent être représentées de manière simple et être rapidement disponibles en cas de besoin. De plus, afin d'atteindre la transparence souhaitée, il est nécessaire de regrouper les indicateurs pertinents sur des tableaux de bord en direct.

6 Liste des abréviations

ACME	Automated Certificate Management Environment
AVM	Artefact Vulnerability Management
BDD	Behaviour Driven Development
CALMS	Culture, Automation, Lean, Measurement, Sharing
CI/CD	Continuous Integration and Continuous Delivery ou Deployment
CPE	Common Platform Enumeration
CSIRT	Computer Security Incident Response Team (équipe CSIRT)
CSSLP	Certified Secure Software Lifecycle Professional
CVE	Common Vulnerability Enumeration
DAST	Dynamic Application Security Testing
DevOps	Terme artificiel constitué de Development (Dev) et Operations (Ops); désigne le cycle de vie complet du produit
DevSecOps	Voir DevOps; Security (Sec) a été intégré pour conférer plus d'importance à ce thème
IAM	Identity & Access Management
IAST	Interactive Application Security Testing
IDS	Intrusion Detection System
OWASP	Open Web Application Security Project
PKI	Public Key Infrastructure
RASP	Runtime Application Self-Protection

SAFe	Scaled Agile Framework
SAST	Static Application Security Testing
SDLC	Software Development Lifecycle
SOC	Security Operations Center
STRIDE	Désigne les six catégories de menaces Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service et Elevation of Privilege
VCS	Version Control System
VUCA	Volatility, Uncertainty, Complexity, Ambiguity
WAF	Pare-feu d'application Web

7 Index

8 Autres ressources

Pour l'approfondissement dans les sous-thèmes individuels de DevSecOps, nous recommandons les sources suivantes au lecteur intéressé.

- A Leader's Framework for Decision Making de David J. Snowden and Mary E. Boone, URL: *hbr.org*
- Accelerate: Building and Scaling High Performing Technology Organizations de Nicole Forsgren Phd, Jez Humble, Gene Kim
- Agile Application Security de Laura Bell, Michael Brunton-Spall, Rich Smith, Jim Bird
- Building Microservices de Sam Newman, URL: *samnewman.io/books/building_microservices*
- DevOps Culture (Part 1) de John Willis, URL: *itrevolution.com/devops-culture-part-1*
- The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations de Gene Kim, John Willis, Patrick Debois, Jez Humble, URL: *itrevolution.com/book/the-devops-handbook*
- Drive de Daniel Pink. URL: *danpink.com/drive*
- Manifesto for Agile Software Development de Kent Beck et coll., URL: *agilemanifesto.org*
- The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win de Gene Kim, Kevin Behr, Georg Spafford, URL: *itrevolution.com/book/the-phoenix-project*
- Project to Product: How to Survive and Thrive in the Age of Digital Disruption with the Flow Framework de Mik Kersten
- Securing DevOps – Security in the cloud de Julien Vehent
- Security Champions Playbook/OWASP Wiki, URL: *owasp.org*
- State Of DevOps Report de Puppet + Splunk, URL: *puppet.com*
- Threat Modeling – Designing for Security de Adam Shostack

Notes

Swisscom SA
Alte Tiefenaustrasse 6
3048 Worblaufen