BE SOFTWARE ARCHITECTURE WISE · CHAPITRE 8

Sécurité, confidentialité et conformité dès la conception

La sécurité et la protection de la vie privée sont des qualités de la structure d'un système, déterminées par l'emplacement des frontières de confiance, par qui peut faire quoi et par le chemin que suivent les données. Ce chapitre couvre les principes de la conception sécurisée, la modélisation des menaces avec des diagrammes de flux de données, STRIDE et les arbres d'attaque, l'identité et l'accès avec OAuth 2.0 et OpenID Connect, le chiffrement, la gestion des clés et les secrets, le zero trust entre services, la sécurité de la chaîne d'approvisionnement logicielle, la protection de la vie privée dès la conception au titre du RGPD, et la sécurité en exploitation. Il s'adresse aux développeurs et aux architectes qui connaissent les architectures distribuées et les architectures de données des chapitres 6 et 7 et qui doivent maintenant les sécuriser.

bewiseapps.com/fr/software-architecture-wise/security-by-design/ · Version 2026-11-03. Information pédagogique uniquement, pas un conseil professionnel.

EN UN COUP D'ŒIL

Vous apprendrez à

- Appliquer la triade CIA, les principes de conception de Saltzer et Schroeder, la défense en profondeur et la sécurité par défaut aux décisions d'architecture, et expliquer ce dont les architectes sont responsables dans un cycle de développement sécurisé.
- Modéliser les menaces d'une conception à l'aide d'un diagramme de flux de données avec frontières de confiance, de STRIDE et d'arbres d'attaque, puis évaluer et consigner les risques et les mesures d'atténuation qui en résultent.
- Distinguer l'authentification de l'autorisation, choisir entre sessions et tokens, et expliquer OAuth 2.0 avec PKCE, les client credentials, OpenID Connect ainsi que RBAC, ABAC et le contrôle d'accès basé sur les relations.
- Décider quoi chiffrer, où et avec quelles clés, à l'aide du chiffrement d'enveloppe, d'un KMS, du hachage des mots de passe, de la gestion des secrets et de la tokenisation.
- Expliquer l'architecture zero trust et concevoir la sécurité de service à service avec le TLS mutuel, l'identité de charge de travail et des points d'application de politique.
- Réduire le risque lié à la chaîne d'approvisionnement logicielle grâce aux SBOM, au cadre SLSA, à la signature, à l'épinglage des versions et aux builds isolés, et résumer les obligations du règlement européen sur la cyberrésilience (Cyber Resilience Act).
- Concevoir en fonction des principes du RGPD, de la protection des données dès la conception et par défaut, et des droits des personnes concernées, y compris l'effacement dans les sauvegardes, les caches et les journaux d'événements.
- Planifier la journalisation de sécurité, la détection, la gestion des vulnérabilités et la réponse aux incidents, y compris la notification des violations dans les 72 heures prévue par le RGPD.

Avant de commencer

Les chapitres 6 et 7, en particulier les microservices, l'architecture pilotée par les événements et la propriété des données ; la section du chapitre 2 sur la sécurité en tant qu'attribut de qualité ; la section du chapitre 0 sur les réseaux, HTTPS et TLS.

Durée

Environ 4 heures de lecture ; 45 minutes pour le test.

Termes clés

frontière de confiance · moindre privilège · défense en profondeur · modélisation des menaces · STRIDE · OAuth 2.0 · OpenID Connect · PKCE · chiffrement d'enveloppe · zero trust · TLS mutuel · SBOM · pseudonymisation · crypto-shredding

La sécurité est une propriété de la structure

La plupart des violations graves ne sont pas dues à une attaque cryptographique exotique. Elles viennent de décisions structurelles : un service interne qui faisait confiance à tout appelant du réseau, une API d'administration qui omettait un contrôle d'autorisation, une

sauvegarde de base de données copiée dans un bucket de stockage lisible par n'importe qui, une clé d'accès commitée dans un dépôt, une dépendance dont personne ne savait que le système l'utilisait. Chacune de ces situations résultait d'un choix architectural, généralement implicite, sur l'emplacement de la confiance, sur ce que chaque composant peut faire et sur le chemin que suivent les données. C'est pourquoi ce chapitre traite la sécurité comme un attribut de qualité intégré à la structure, au sens des scénarios de sécurité du chapitre 2, et non comme une liste de produits achetés à la fin.

C'est aussi pourquoi la sécurité est indissociable des styles des chapitres 6 et 7. Chaque frontière de service, file, cache, réplique et copie analytique agrandit la surface d'attaque et crée un endroit de plus où les données doivent être protégées. La distribution multiplie le nombre de frontières de confiance ; l'architecture des données décide combien de copies des données à caractère personnel existent et combien de temps elles vivent. L'exemple fil rouge est la plateforme de livraison de repas utilisée tout au long du cours : des clients, des restaurants et des livreurs, une API gateway, des services de commandes, de paiements, d'affectation et de notification, un flux d'événements et un entrepôt de données.

Comment les huit sections s'articulent

Les quatre premières sections construisent le cœur de la conception sécurisée. **Principes de sécurité** pose le vocabulaire : la triade CIA, les principes de conception de Saltzer et Schroeder (1975) comme le moindre privilège et les valeurs par défaut sûres, la défense en profondeur, la sécurité par défaut, l'OWASP Top 10 et un cycle de développement sécurisé fondé sur le SSDF du NIST. **Modélisation des menaces** transforme ces principes en une activité de conception reproductible : dessiner un diagramme de flux de données avec ses frontières de confiance, se demander ce qui peut mal tourner avec STRIDE et les arbres d'attaque, évaluer les risques et décider des mesures d'atténuation. **Identité et accès** traite de qui appelle : les mots de passe tels que le NIST les recommande aujourd'hui, les passkeys, les sessions et les tokens, OAuth 2.0 avec PKCE, OpenID Connect, et les modèles d'autorisation RBAC, ABAC et le contrôle d'accès basé sur les relations. **Protéger les données** couvre le chiffrement en transit et au repos, le hachage des mots de passe, la gestion des clés avec le chiffrement d'enveloppe, les secrets, la tokenisation et la décision de ce qu'il faut chiffrer et où.

Les quatre sections suivantes élargissent le regard. **Zero trust** remplace l'idée d'un réseau interne de confiance par une authentification et une autorisation à chaque requête entre services, à l'aide du TLS mutuel, de certificats et de points de décision et d'application de politique. **Sécurité de la chaîne d'approvisionnement** traite les dépendances, les systèmes de build et les images de conteneurs comme faisant partie de la surface d'attaque, avec les nomenclatures logicielles, le cadre SLSA, la signature et le règlement européen sur la cyberrésilience. **Protection de la vie privée dès la conception** explique comment le RGPD et le RGPD britannique (UK GDPR) façonnent l'architecture : minimisation des données,

pseudonymisation, droits des personnes concernées, conservation et effacement dans les sauvegardes et les journaux d'événements. **Sécurité en exploitation** boucle la boucle avec la journalisation de sécurité, la détection, la gestion des vulnérabilités, la réponse aux incidents et la notification des violations.

Ce que les architectes décident, et ce qu'ils délèguent

Un architecte n'a pas besoin d'être testeur d'intrusion ni cryptographe, et ne doit pas inventer de cryptographie. Son rôle est de prendre les décisions coûteuses à changer et qu'aucune équipe ne peut prendre seule : où se situent les frontières de confiance, quel fournisseur d'identité et quel modèle de tokens utilise toute la plateforme, quel composant est le point de décision de politique pour quelles ressources, quelles données sont classées comme restreintes et donc chiffrées avec quelles clés, comment les services se prouvent mutuellement leur identité, et comment l'organisation détectera une attaque et y répondra. Ces décisions ont leur place dans les enregistrements de décision d'architecture et dans les concepts transversaux d'une description arc42, comme toute autre décision importante.

Les développeurs sont responsables d'un ensemble de préoccupations différent et tout aussi important : valider les entrées, encoder les sorties, utiliser correctement les bibliothèques d'authentification et d'autorisation de la plateforme, gérer les erreurs sans échouer en mode ouvert, tenir les dépendances à jour et écrire des tests pour le comportement de sécurité. Les deux se rejoignent dans la modélisation des menaces, dans les revues de conception sécurisée et dans les vérifications automatisées du pipeline de livraison. Lorsque les praticiens sont en désaccord, par exemple sur la part du travail de sécurité à transférer aux développeurs ou sur le degré de formalisme de la modélisation des menaces, le chapitre présente les deux positions.

Droit, normes et exactitude

La sécurité et la protection de la vie privée regorgent de normes numérotées et de dispositions juridiques, et ce chapitre les nomme avec précision : les publications spéciales du NIST, les RFC de l'IETF, les projets de l'OWASP, PCI DSS, le RGPD et le RGPD britannique, et le règlement européen sur la cyberrésilience. Les normes évoluent ; chaque mention donne donc l'édition en vigueur à la dernière revue indiquée sur la section et renvoie à la source qui fait autorité. Les passages sur la réglementation expliquent comment le droit influe sur l'architecture ; il s'agit d'informations, non de conseils juridiques, et ce sont les juristes de l'organisation et son délégué à la protection des données qu'il faut consulter pour une situation précise.

L’idée à retenir

Décidez délibérément, tôt et par écrit, à qui et à quoi votre système fait confiance, pour quoi et pourquoi, et rendez chacune de ces décisions de confiance vérifiable. La plupart des techniques de ce chapitre, des modèles de menaces et du moindre privilège aux scopes OAuth, au chiffrement d’enveloppe, au TLS mutuel, aux builds signés et aux journaux d’audit, sont des moyens de remplacer une confiance implicite par une confiance explicite et vérifiable. Un système conçu ainsi limite les dégâts de la violation qui finira par se produire, et peut montrer à ses clients, à ses auditeurs et aux autorités de régulation comment il les protège.

SECTION 8.1 SUR 8.8 · INTERMÉDIAIRE · 28 MIN DE LECTURE · DERNIÈRE RELECTURE : NOVEMBRE 2026

Principes de sécurité pour les architectes

AVANT DE LIRE

Cette section s'appuie sur :

- 0.4 Comment les ordinateurs communiquent
- 0.6 API et intégration
- 2.7 Sécurité, utilisabilité et autres qualités

Termes déjà vus qu'elle utilise : Triade CIA, API gateway (passerelle d'API), TLS et HTTPS, Limite de débit (rate limit)

L'IDÉE EN BREF

Pensez à la façon dont une agence bancaire est protégée. La chambre forte n'est pas gardée par une seule serrure mais par plusieurs : une porte d'entrée fermée à clé, un personnel qui n'a les clés que des pièces dont il a besoin, une chambre forte qu'il faut être deux pour ouvrir, des caméras, et une alarme qui appelle la police. Le bâtiment a été conçu avec tout cela en tête dès le départ ; personne n'a greffé une chambre forte sur une boutique après coup. La sécurité logicielle fonctionne de la même façon. Les architectes décident tôt où se trouvent les portes fermées à clé, qui reçoit quelles clés et ce qui se passe quand une serrure cède, afin qu'une seule erreur n'ouvre pas tout.

CE QU'IL FAUT SAVOIR

- La sécurité est une qualité de la structure : l'emplacement des frontières de confiance, ce que chaque composant peut faire et le chemin que suivent les données déterminent la plupart des propriétés de sécurité, et la plupart des violations exploitent de telles décisions plutôt que la cryptographie.
- La confidentialité, l'intégrité et la disponibilité, plus l'authenticité, l'imputabilité et la non-répudiation, doivent être nommées pour chaque actif parce qu'elles entrent en conflit : échouer en mode fermé protège les données mais coûte de la disponibilité.
- Les principes de Saltzer et Schroeder de 1975 (économie de mécanisme, valeurs par défaut sûres, médiation complète, conception ouverte, séparation des privilèges, moindre privilège, moindre mécanisme commun, acceptabilité psychologique) restent le fondement de la conception sécurisée.

- La défense en profondeur superpose des contrôles indépendants afin qu'une seule défaillance n'expose pas un actif ; la sécurité par défaut et une paved road (chemin balisé) font du chemin le plus facile le chemin sûr.
- Réduisez la surface d'attaque et placez l'authentification, l'autorisation et la validation à chaque frontière de confiance ; tout ce qui se trouve côté client est à l'extérieur de la frontière.
- L'OWASP Top 10:2025 est une liste de sensibilisation, pas une norme ; le SSDF du NIST (SP 800-218) décrit un cycle de développement sécurisé, et les architectes sont responsables des décisions de sécurité qui concernent plusieurs équipes, tandis que les équipes sont responsables de leur bonne application.

La sécurité, une qualité de la structure

Le chapitre 2 a présenté la sécurité comme un attribut de qualité : la capacité du système à protéger les données et les fonctions contre une utilisation non autorisée tout en servant ceux qui sont autorisés, exprimée par des scénarios de sécurité testables qui nomment un attaquant, une attaque, une cible et une réponse mesurable. Ce chapitre va plus loin et se demande comment l'architecture produit cette qualité. La réponse est que la plupart des propriétés de sécurité sont déterminées par la structure : quels composants existent, lesquels font confiance à quels autres, où les données sont stockées et copiées, et par quels rares points chaque requête doit passer.

Voyez comment se produisent les violations réelles. Un attaquant modifie le numéro de commande dans une URL et voit l'adresse d'un autre client, parce que le service a vérifié que l'utilisateur était connecté mais pas que la commande était la sienne. Un bucket de stockage contenant des exports de base de données est lisible depuis Internet parce qu'un paramètre par défaut a été modifié il y a des années. Un service interne de reporting accepte toute requête provenant de l'intérieur du réseau, si bien qu'un seul ordinateur portable compromis atteint chaque fiche client. Aucune de ces failles n'exige de cryptographie ingénieuse pour être exploitée ; chacune est une décision de conception sur la confiance que personne n'a prise explicitement. La sécurité dès la conception consiste à prendre ces décisions délibérément, tôt et par écrit, puis à les rendre difficiles à enfreindre par accident.

L'architecture de sécurité se range donc aux côtés des autres préoccupations transversales de la section 8 d'arc42 : un modèle d'identité, un modèle d'autorisation, un concept de protection des données et un concept de journalisation que chaque équipe applique. La suite de cette section pose le vocabulaire et les principes sur lesquels s'appuient les autres sections du chapitre.

Confidentialité, intégrité et disponibilité dans un système réel

La triade CIA est le point de départ. Sur la plateforme de livraison de repas, la **confidentialité** couvre les noms, adresses, numéros de téléphone et historiques de commandes des clients, la position en direct et les pièces d'identité des livreurs, les coordonnées bancaires de versement des restaurants et les propres secrets de la plateforme. L'**intégrité** couvre les prix, les totaux de panier, les remboursements, les versements aux livreurs et la carte que voit un client : un attaquant capable de ramener le prix d'un repas à zéro, ou de détourner les versements d'un restaurant vers son propre compte bancaire, n'a rien lu de secret mais a causé un tort sérieux. La **disponibilité** compte surtout pendant le pic du soir, quand une attaque par déni de service ou une infection par rançongiciel coûte du chiffre d'affaires à chaque minute.

L'ISO/IEC 25010 mentionne aussi l'**authenticité** (les identités peuvent être prouvées), l'**imputabilité** (les actions peuvent être rattachées à leur auteur), la **non-répudiation** (on peut prouver que les actions ont eu lieu) et, nouveauté de l'édition 2023, la **résistance** (le produit continue de fonctionner comme prévu sous attaque). Les architectes doivent nommer les propriétés qui comptent pour chaque actif, car elles tirent dans des directions différentes. Échouer en mode fermé protège la confidentialité et l'intégrité mais coûte de la disponibilité : si le service d'autorisation est en panne, un service des commandes qui échoue en mode fermé refuse les commandes. Échouer en mode ouvert préserve la disponibilité et joue tout le reste à pile ou face. Le bon choix diffère selon l'opération : la consultation d'une carte publique peut échouer en mode ouvert vers une copie en cache, alors que l'émission d'un remboursement doit échouer en mode fermé.

Les principes de conception de Saltzer et Schroeder

En 1975, Jerome Saltzer et Michael Schroeder, forts de leur expérience du système d'exploitation Multics au MIT, ont publié huit principes de conception pour les mécanismes de protection dans *The Protection of Information in Computer Systems*. Un demi-siècle plus tard, ils restent le fondement le plus cité de la conception sécurisée, parce qu'ils portent sur la structure plutôt que sur la technologie. Le tableau reformule chacun d'eux et donne un exemple moderne tiré de la plateforme de livraison de repas.

Les huit principes de conception, avec des exemples modernes

Principe	Ce qu'il demande	Exemple moderne
Économie de mécanisme	Garder la conception de sécurité aussi petite et simple que possible, afin de pouvoir l'inspecter et la tester à fond.	Une seule bibliothèque d'autorisation ou un seul moteur de politique, partagé et bien relu, au lieu de contrôles de permission écrits à la main différemment dans chaque service.
Valeurs par défaut sûres	Fonder l'accès sur une permission explicite ; en cas de doute, ou quand quelque chose échoue, la réponse est non.	Un nouvel endpoint d'API est refusé tant qu'aucune politique n'y donne accès ; un moteur de politique injoignable signifie refuser, pas autoriser.
Médiation complète	Vérifier chaque accès à chaque objet, à chaque fois, et pas seulement la première fois.	Le service des commandes vérifie la propriété à chaque `GET /orders/{id}` au lieu de faire confiance à n'importe quel identifiant venant d'un utilisateur connecté ; les décisions en cache expirent rapidement.
Conception ouverte	La sécurité ne doit pas dépendre du secret de la conception ; seuls les clés et les mots de passe doivent être secrets.	Des algorithmes publiés (AES, TLS 1.3) et des protocoles standard (OAuth 2.0) qui restent sûrs même si les attaquants lisent le code source.
Séparation des privilèges	Lorsque c'est possible, exiger deux conditions indépendantes (deux clés) avant d'accorder l'accès.	Les remboursements au-delà d'un seuil exigent un second approbateur ; une mise en production exige à la fois une revue de code et un pipeline au vert ; connexion multifacteur.
Moindre privilège	Ne donner à chaque programme et à chaque utilisateur que les privilèges nécessaires à sa tâche, et pour la durée nécessaire.	Le compte de base de données du service de la carte ne peut pas lire le schéma des paiements ; le rôle cloud d'un job de build peut pousser des images mais pas supprimer des clusters.
Moindre mécanisme commun	Réduire au minimum les mécanismes partagés entre utilisateurs, car chaque mécanisme partagé est une voie de fuite d'informations.	Des caches, des files ou des pools de connexions séparés par locataire ou par niveau de confiance ; un cache partagé dont les clés sont mal conçues sert les données d'un utilisateur à un autre.
Acceptabilité psychologique	La protection doit être facile à utiliser correctement, sinon les gens la contourneront.	Des passkeys au lieu de règles de mots de passe complexes ; une sécurité par défaut dans la plateforme ; des avertissements rares et donc lus.

D'OÙ CELA VIENT

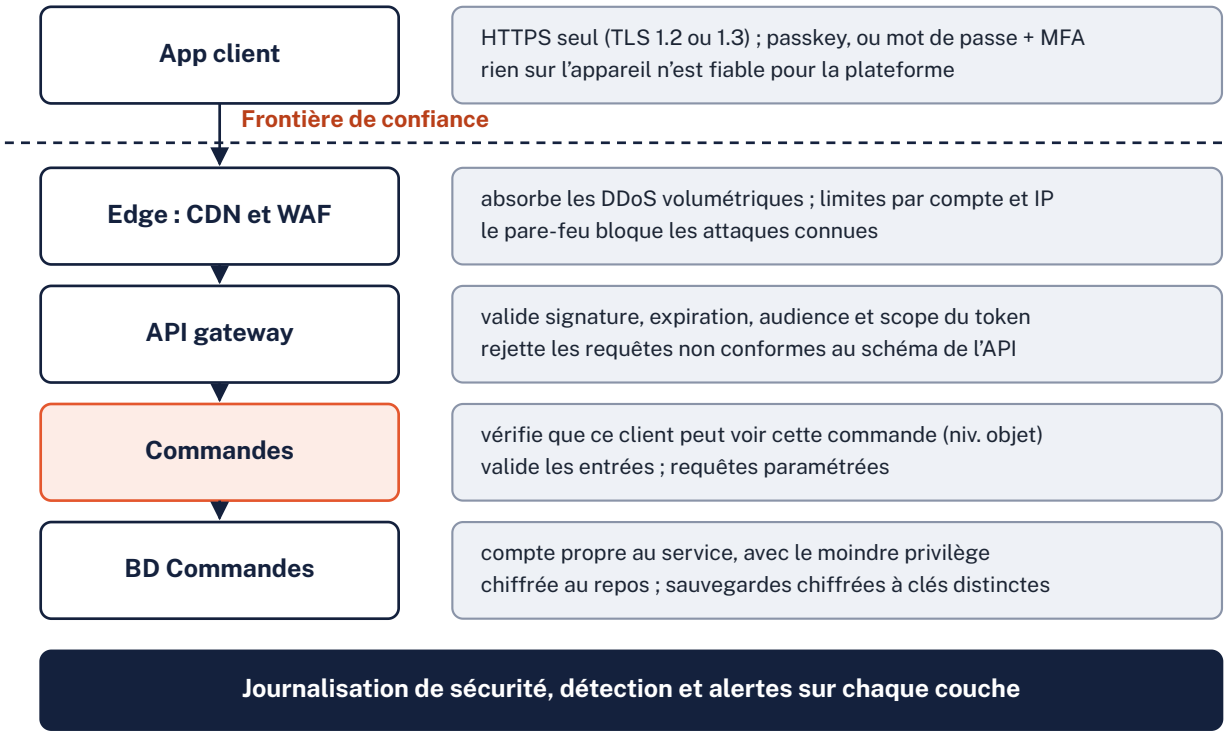
D'où viennent ces principes

L'article de Saltzer et Schroeder a paru dans les *Proceedings of the IEEE* en septembre 1975. Outre les huit principes, il en décrivait deux autres, le **facteur de travail** (comparer le coût du contournement d'un mécanisme aux ressources de l'attaquant) et l'**enregistrement des compromissions** (enregistrer qu'une compromission a eu lieu même si on ne peut pas l'empêcher), dont les auteurs notaient qu'ils ne s'appliquent qu'imparfaitement aux systèmes informatiques. Tous deux survivent aujourd'hui : le facteur de travail dans le hachage des mots de passe et la longueur des clés, l'enregistrement des compromissions dans les journaux d'audit qui révèlent toute altération. La conception ouverte reprend le principe énoncé en 1883 par Auguste Kerckhoffs pour les chiffres militaires : un système doit rester sûr même si tout ce qui le concerne, à l'exception de la clé, est public.

Défense en profondeur

La défense en profondeur consiste à superposer plusieurs contrôles indépendants, afin que la défaillance ou le contournement de l'un d'eux n'expose pas l'actif. L'idée vient de la fortification militaire et c'est une pratique courante en ingénierie de la sûreté comme en sécurité. Sa valeur dépend entièrement du mot *indépendants* : deux couches qui échouent pour la même raison, comme deux services qui font tous deux confiance à un en-tête falsifié, ne sont qu'une seule couche dessinée deux fois.

Le diagramme montre les couches sur le chemin d'une commande de la plateforme de livraison de repas. Si la validation des tokens par la passerelle a un bug, le contrôle au niveau objet du service des commandes empêche toujours un client de lire la commande d'un autre. Si un attaquant vole les instantanés de disque de la base de données, le chiffrement au repos, avec des clés conservées ailleurs, les protège. Si tout cela échoue, la supervision devrait remarquer le schéma inhabituel des requêtes, et quelqu'un devrait réagir. La défense en profondeur est aussi la réponse honnête à la question « et si ce contrôle échoue ? », que pose tout modèle de menaces.



Chaque couche applique ses propres contrôles ; le contournement d'un seul contrôle n'expose pas les données.

SI VOUS DÉBUTEZ

Pas à pas : quand deux serrures n'en font qu'une

L'idée qui demande le plus d'attention ici est que la défense en profondeur ne fonctionne que si les couches sont indépendantes, de sorte que chacune puisse arrêter une attaque qui a franchi les autres. Suivez une requête le long du chemin de commande du diagramme.

- Un client connecté demande à voir sa commande 1042. L'app envoie la requête avec le jeton d'accès du client, qui indique de qui il s'agit.
- L'API gateway vérifie que le token est authentique, non expiré et destiné à cette API, puis transmet la requête au service des commandes.
- Le client modifie alors la requête et remplace 1042 par 1043, la commande de quelqu'un d'autre. Le token est toujours valide, donc la passerelle la laisse passer : elle ne peut pas savoir à qui appartient chaque commande. Le service des commandes vérifie, constate que la commande 1043 n'est pas celle de ce client, et refuse.
- Ce second contrôle est une couche distincte. Même si un bug laissait passer un mauvais token à travers la passerelle, le service des commandes refuserait toujours de montrer à un client la commande d'un autre.
- Mais supposez que le service des commandes vérifie aussi le token, avec la même bibliothèque partagée que la passerelle, et qu'un défaut de cette bibliothèque accepte des tokens sans signature valide. Un seul token falsifié tromperait alors les deux contrôles à la fois. Deux contrôles qui échouent pour la même raison ne sont qu'une seule couche dessinée deux fois.
- Les couches inférieures suivent la même règle. Le chiffrement au repos ne protège les instantanés de disque volés que parce que ses clés sont conservées ailleurs, et la supervision est là pour remarquer une attaque si tout le reste échoue.
- **La leçon :** pour chaque contrôle, demandez-vous « et si celui-ci échoue ? », et assurez-vous que ce qui le met en échec ne mettrait pas aussi en échec les couches situées derrière lui.

Sécurisé par défaut et sécurisé dès la conception

Un produit est sécurisé par défaut lorsque sa configuration d'origine est la configuration sûre, de sorte qu'affaiblir la protection exige un acte délibéré. Beaucoup de violations de grande ampleur sont venues de valeurs par défaut non sécurisées laissées en place : des bases de données à l'écoute sur Internet sans mot de passe, des consoles d'administration avec les identifiants d'usine, des buckets de stockage qu'un seul paramètre suffisait à rendre publics. Les fournisseurs ont évolué : depuis avril 2023, par exemple, les nouveaux buckets Amazon S3

sont créés avec Block Public Access activé et les listes de contrôle d'accès désactivées. En 2023, la Cybersecurity and Infrastructure Security Agency (CISA) des États-Unis et des agences partenaires de plusieurs pays ont publié des recommandations communes exhortant les fabricants à rendre leurs produits sécurisés dès la conception et par défaut, et en 2024 la CISA a lancé un engagement volontaire, le Secure by Design pledge, destiné aux éditeurs de logiciels.

Pour les architectes, le principe « sécurisé par défaut » s'applique à la plateforme autant qu'aux produits. Si le modèle de service partagé inclut déjà la validation des tokens, une journalisation de sécurité structurée, un utilisateur de conteneur non root et un compte de base de données au moindre privilège, chaque nouveau service démarre sécurisé et les équipes doivent s'en écarter explicitement, ce qui est visible en revue. C'est ce que certaines organisations appellent une **paved road** (chemin balisé) : le chemin le plus facile est aussi le chemin sécurisé.

Un sujet proche est le comportement d'un système quand quelque chose tourne mal. L'OWASP Top 10:2025 a ajouté une catégorie pour la mauvaise gestion des conditions exceptionnelles, qui inclut l'échec en mode ouvert : une erreur dans un contrôle de permission qui est interceptée et ignorée, un délai d'expiration qui conduit par défaut à « autorisé », un échec d'analyse syntaxique qui saute la validation. Les valeurs par défaut sûres, appliquées dans le code, signifient que chaque chemin d'exception d'une décision de sécurité se termine par un refus.

Surface d'attaque et frontières de confiance

La surface d'attaque est tout ce qu'un attaquant peut atteindre et influencer : les endpoints publics et leurs paramètres, les ports ouverts, les envois de fichiers, les files de messages qui acceptent des entrées de partenaires, les webhooks, les consoles d'administration, les apps mobiles qui peuvent être décompilées, ainsi que le code tiers et les personnes auxquels le système fait confiance. Michael Howard, chez Microsoft, a popularisé l'idée de la mesurer au début des années 2000 ; la règle pratique est de la rendre aussi petite que possible et de savoir exactement ce qu'elle contient. Supprimer un endpoint inutilisé, fermer à Internet un port d'administration ou désactiver une fonctionnalité que personne n'utilise réduit le risque à moindre coût que n'importe quel contrôle ajouté. Chaque choix de style architectural du chapitre 6 a ici un coût de sécurité : chaque nouveau service accessible par le réseau ajoute des endpoints, des identifiants et du trafic à protéger.

Une frontière de confiance est l'endroit où des données ou le contrôle passent entre des parties de niveaux de confiance ou de privilège différents. La plus évidente se situe entre Internet et la plateforme, mais il y en a bien d'autres : entre l'app client et l'API, entre la tablette d'un restaurant et le service de la carte, entre des services détenus par des équipes différentes, entre la plateforme et un prestataire de paiement, entre le code applicatif et la base de

données, et entre les utilisateurs ordinaires et les administrateurs. Les frontières de confiance comptent parce que c'est là que les contrôles ont leur place : l'authentification, l'autorisation, la validation des entrées et l'encodage des sorties ont lieu quand les données franchissent une frontière, jamais en supposant que l'autre côté l'a déjà fait. Tracer les frontières de confiance sur un diagramme de flux de données est la première étape de la modélisation des menaces.

ATTENTION

Le client est à l'extérieur de la frontière

Tout ce qui s'exécute sur l'appareil du client, qu'il s'agisse d'une app mobile, d'une application web monopage ou de la tablette d'un restaurant, est sous le contrôle de l'attaquant. Les prix, remises, permissions et totaux de commande calculés côté client sont des suggestions ; le serveur doit les recalculer ou les vérifier. Masquer un bouton dans l'interface utilisateur n'est pas une autorisation. Les clés d'API intégrées dans une app mobile ne sont pas secrètes, car l'app peut être décompilée.

L'OWASP Top 10, une liste de sensibilisation

L'OWASP Top 10 est la liste classée, établie par l'Open Worldwide Application Security Project, des risques de sécurité les plus critiques pour les applications web, compilée à partir de données de vulnérabilités fournies par des contributeurs et d'une enquête auprès de la communauté. L'édition actuelle, l'**OWASP Top 10:2025**, est la huitième. Par rapport à l'édition 2021, le contrôle d'accès défaillant reste premier et inclut désormais la falsification de requêtes côté serveur (SSRF), la mauvaise configuration de sécurité monte à la deuxième place, l'ancienne catégorie des composants vulnérables et obsolètes s'élargit en **défaillances de la chaîne d'approvisionnement logicielle**, et une nouvelle catégorie couvre la **mauvaise gestion des conditions exceptionnelles**.

L'OWASP Top 10:2025 et les leviers architecturaux

Catégorie	Ce qui tourne mal	Levier architectural
A01 Broken Access Control (contrôle d'accès défaillant)	Des utilisateurs agissent en dehors de leurs permissions, par exemple en lisant la commande d'un autre client en modifiant un identifiant ; inclut la falsification de requêtes côté serveur.	Un seul modèle d'autorisation, refus par défaut, contrôles au niveau objet dans chaque service, accès réseau sortant restreint.
A02 Security Misconfiguration (mauvaise configuration de sécurité)	Valeurs par défaut non sécurisées, stockage ouvert, pages d'erreur trop bavardes, fonctionnalités inutiles laissées actives.	Infrastructure en tant que code, modèles durcis, vérifications automatisées des politiques dans le pipeline.
A03 Software Supply Chain Failures (défaillances de la chaîne d'approvisionnement logicielle)	Dépendances, outils de build et canaux de distribution vulnérables ou compromis.	Inventaire des dépendances (SBOM), épinglage des versions, provenance signée ; voir la sécurité de la chaîne d'approvisionnement.
A04 Cryptographic Failures (défaillances cryptographiques)	Données sensibles non chiffrées, algorithmes faibles, mauvaise gestion des clés.	Classification des données, TLS partout, un KMS et le chiffrement d'enveloppe.
A05 Injection	Des données non fiables interprétées comme du code : SQL, commandes du système d'exploitation, cross-site scripting.	Requêtes paramétrées, encodage des sorties adapté au contexte, API sûres des frameworks.
A06 Insecure Design (conception non sécurisée)	Des contrôles absents ou inefficaces qu'aucun codage correct, si soigné soit-il, ne peut corriger.	Modélisation des menaces, patrons de conception sécurisée, cas d'abus à côté des récits utilisateur.
A07 Authentication Failures (défaillances d'authentification)	Bourrage d'identifiants (credential stuffing), parcours de récupération faibles, défauts de gestion des sessions.	Un fournisseur d'identité central, MFA ou passkeys, limites de débit sur la connexion.
A08 Software or Data Integrity Failures (défaillances d'intégrité des logiciels ou des données)	Faire confiance à du code, à des mises à jour ou à des données sans en vérifier l'intégrité, par exemple des mises à jour non signées ou une désérialisation non sûre.	Vérification des signatures, contrôles d'intégrité sur les données qui franchissent des frontières.
A09 Security Logging and Alerting Failures (défaillances de journalisation et d'alerte de sécurité)	Les attaques ne sont pas enregistrées, ou enregistrées mais jamais remarquées ni traitées.	Une norme de journalisation de sécurité, des alertes, des runbooks d'incident.

Catégorie	Ce qui tourne mal	Levier architectural
A10 Mishandling of Exceptional Conditions (mauvaise gestion des conditions exceptionnelles)	Erreurs traitées de façon incohérente, systèmes qui échouent en mode ouvert, messages d'erreur qui divulguent des détails.	Échec en mode fermé, gestion centralisée des erreurs, limites de ressources.

Traitez la liste pour ce que l'OWASP dit qu'elle est : un document de sensibilisation, pas une norme. Elle est utile pour la formation, pour vérifier qu'un modèle de menaces n'a pas oublié toute une classe de problèmes et pour dialoguer avec les parties prenantes. Ce n'est pas un ensemble complet d'exigences, et « nous sommes conformes à l'OWASP Top 10 » ne veut pas dire grand-chose. Pour des exigences vérifiables, l'Application Security Verification Standard (ASVS) de l'OWASP énumère des contrôles testables selon trois niveaux de rigueur, et l'OWASP API Security Top 10, distinct, couvre les risques propres aux API, comme l'autorisation défaillante au niveau objet.

Un cycle de développement sécurisé et le shift left

Un cycle de développement sécurisé intègre des activités de sécurité à chaque étape de la livraison au lieu de compter sur un test juste avant la mise en production. Microsoft a rendu obligatoire son Security Development Lifecycle en 2004, après le mémo Trustworthy Computing de Bill Gates en 2002, et l'idée s'est répandue dans tout le secteur. La référence actuelle est le Secure Software Development Framework (SSDF) du NIST, SP 800-218, dont la version 1.1 a été publiée en février 2022 ; le NIST a soumis à consultation un projet de révision (version 1.2) en décembre 2025, qui n'avait pas été finalisé au moment de la rédaction. Le SSDF regroupe les pratiques en quatre familles : **préparer l'organisation** (rôles, formation, chaînes d'outils, exigences de sécurité), **protéger le logiciel** (protéger le code et l'intégrité du build, conserver la provenance), **produire un logiciel bien sécurisé** (concevoir pour satisfaire les exigences de sécurité, modéliser les menaces, relire et tester, configurer de manière sécurisée par défaut) et **répondre aux vulnérabilités** (identifier, corriger et analyser les causes profondes pour que la même classe ne se reproduise pas). Il décrit des résultats plutôt que de prescrire des outils, ce qui en fait une bonne liste de contrôle pour un architecte qui évalue un processus de livraison.

Faire du shift left, c'est déplacer le travail de sécurité plus tôt : exigences de sécurité et cas d'abus avec les récits utilisateur, modélisation des menaces lors de la conception, analyse statique et détection de secrets à chaque commit, vérification des dépendances dans le pipeline. L'argument économique est connu depuis le chapitre 1 : un contrôle d'autorisation manquant découvert en revue de conception coûte une heure ; la même faille découverte après une violation coûte une notification aux autorités de régulation. Les praticiens ne s'accordent

pas sur le degré auquel il faut le pousser. Ses partisans disent que les développeurs qui comprennent la sécurité écrivent moins de vulnérabilités. Ses détracteurs font remarquer que le simple fait de placer des outils de sécurité devant les développeurs les submerge de constats de faible qualité et fait de la sécurité le travail à temps partiel de tout le monde et l'expertise de personne ; ils plaident pour une sécurité par défaut intégrée à la plateforme (la paved road), afin que le chemin sécurisé ne demande aucun effort supplémentaire, et pour des spécialistes de la sécurité intégrés là où le risque est le plus élevé. La plupart des organisations matures font les deux, et continuent aussi de tester et de superviser en production, ce qui explique que certaines parlent désormais de « shift everywhere », la sécurité à chaque étape.

Tester le système en fonctionnement : DAST et fuzzing

L'analyse statique lit le code ; ces techniques le font fonctionner. Le **test dynamique de sécurité des applications** (DAST) sonde de l'extérieur une application en fonctionnement avec des requêtes hostiles, sans avoir besoin de son code source, à la recherche de failles comme l'injection SQL, le cross-site scripting, une authentification faible et une mauvaise configuration du serveur. Il convient à une exécution nocturne sur un environnement de test, mais il ne trouve que ce qu'il peut atteindre et il est généralement faible sur la logique métier, par exemple quand un client consulte la commande d'un autre. Le **fuzzing** soumet à un programme des entrées malformées ou inattendues jusqu'à ce qu'il échoue. Il a débuté avec le groupe de Barton Miller à l'université du Wisconsin à Madison, dont l'outil `fuzz` a fait planter ou bloquer entre un quart et un tiers environ des utilitaires UNIX testés. Il est particulièrement utile pour les analyseurs syntaxiques, comme le code d'importation de cartes de la plateforme. OSS-Fuzz de Google l'exécute à grande échelle pour les projets open source.

Ce dont les architectes sont responsables, et ce dont les développeurs sont responsables

Le travail de sécurité est partagé, et la confusion sur qui est responsable de quoi est en soi une vulnérabilité. L'architecte est responsable des décisions qui franchissent les frontières entre équipes et sont coûteuses à changer ; les équipes sont responsables de leur bonne application. Un schéma courant associe une petite équipe de sécurité centrale qui fixe les normes et exploite les services de sécurité de la plateforme, des référents sécurité (security champions) dans chaque équipe produit qui animent la modélisation des menaces et les revues, et des architectes qui veillent à ce que les décisions structurelles soient prises et consignées.

Répartir les responsabilités de sécurité

Préoccupation	Architecte (et architecture de sécurité)	Équipes de développement
Frontières de confiance et flux de données	Les définir, les documenter, animer la modélisation des menaces pour les changements importants.	Garder le code cohérent avec elles ; signaler les changements qui déplacent une frontière.
Identité et accès	Choisir le fournisseur d'identité, le modèle de tokens, le modèle d'autorisation et l'endroit où la politique est décidée.	Appliquer l'autorisation à chaque endpoint et à chaque objet ; la tester.
Protection des données	Classer les données ; décider de ce qui est chiffré où, avec quelles clés ; choisir le KMS et le coffre de secrets.	Utiliser les bibliothèques approuvées ; ne jamais journaliser de secrets ni, sans nécessité, de données à caractère personnel.
Traitement des entrées et des sorties	Fixer la stratégie de validation et d'encodage aux frontières ; fournir des bibliothèques partagées.	Valider, encoder et paramétrer dans chaque gestionnaire.
Dépendances et build	Fixer les contrôles du pipeline, les sources approuvées et la politique SBOM.	Tenir les dépendances à jour ; traiter les alertes dans les délais convenus.
Détection et réponse	Définir la journalisation de sécurité, les événements d'audit et les rôles en cas d'incident.	Émettre les événements convenus ; corriger les vulnérabilités dans les niveaux de service convenus.

EXEMPLE COMMENTÉ

Exemple commenté : appliquer les principes à une fonction de remboursement par le support

Les agents du support de la plateforme de livraison de repas doivent rembourser les clients pour les livraisons ratées depuis l'outil de back-office. Une première conception permet à n'importe quel agent d'appeler l'endpoint de remboursement du service Paiements avec n'importe quel numéro de commande et n'importe quel montant. Le tableau montre comment les principes la remodèlent.

Moindre privilège	Les agents reçoivent une permission `refund:create` uniquement pour les commandes de leur propre marché ; l'identité de service de l'outil de back-office peut appeler l'API de remboursement mais jamais lire les données de carte.
Valeurs par défaut sûres	Si le contrôle de politique ou la recherche de la commande échoue ou expire, le remboursement est refusé avec un message « réessayez » ; aucun chemin de repli ne saute les contrôles.
Médiation complète	Paiements revérifie lui-même chaque remboursement par rapport au débit initial (le remboursement ne peut pas dépasser le montant capturé moins les remboursements antérieurs), au lieu de faire confiance à l'outil de back-office.
Séparation des privilèges	Les remboursements de plus de £100, ou un troisième remboursement au même client dans la semaine, exigent l'approbation d'un superviseur dans le même outil.
Économie de mécanisme	Utilise le moteur de politique et la bibliothèque d'audit existants de la plateforme ; aucun code de permission spécifique n'est écrit pour les remboursements.
Acceptabilité psychologique	L'approbation se demande en un clic ; les motifs se choisissent dans une courte liste, si bien que le contrôle ne ralentit pas les agents honnêtes.
Imputabilité	Chaque remboursement est inscrit dans un journal d'audit en ajout seul, avec l'agent, l'approbateur, la commande, le montant et le motif.
Défense en profondeur	L'outil de back-office n'est accessible que par authentification unique (SSO) avec une MFA résistante à l'hameçonnage, depuis des appareils gérés.

Un agent malhonnête comme un attaquant qui a pris le contrôle du compte d'un agent sont désormais limités à de petits remboursements journalisés dans un seul marché.

Le coût est un court délai pour les gros remboursements et un peu de travail de conception supplémentaire, ce qui est bien moins cher que d'ajouter ces contrôles après coup, une fois la fraude découverte.

VÉRIFIEZ VOTRE COMPRÉHENSION

▼ **Pourquoi la défense en profondeur dépend-elle de l'indépendance des couches ?**

Si deux couches échouent pour la même raison, par exemple parce qu'elles font toutes deux confiance au même en-tête falsifié, une seule attaque les met toutes deux en échec, si bien qu'elles ne protègent pas mieux qu'une seule couche.

▼ **Quel principe de Saltzer et Schroeder est enfreint lorsqu'un service vérifie qu'un utilisateur est connecté, mais pas que la commande demandée lui appartient ?**

La médiation complète : chaque accès à chaque objet doit faire l'objet d'un contrôle d'autorité, et pas seulement le fait que l'utilisateur s'est authentifié.

▼ **Une équipe affirme que son application est « conforme à l'OWASP Top 10 ». Qu'est-ce qui ne va pas dans cette affirmation ?**

Le Top 10 est une liste de sensibilisation à des catégories de risques, pas une norme comportant des exigences testables ; la vérification nécessite quelque chose comme l'ASVS de l'OWASP et un modèle de menaces du système réel.

▼ **Donnez un exemple d'échec en mode ouvert et expliquez pourquoi il est dangereux.**

Un contrôle de permission qui traite un délai d'expiration du moteur de politique comme « autorisé ». Un attaquant capable de ralentir ou de casser le moteur de politique obtient alors l'accès à tout ce qu'il protège.

▼ **À vous : N'importe quel employé de billetterie peut appliquer une remise de 100 % à n'importe quelle réservation, et l'app se contente de masquer le bouton au personnel junior. Quels principes de sécurité cela enfreint-il ? Reconcevez cela brièvement, en précisant qui peut approuver les remises importantes.**

Masquer un bouton se fait côté client, donc hors de la frontière de confiance, et la médiation complète n'est pas respectée : le serveur doit vérifier chaque remise. Le moindre privilège n'est pas respecté non plus. Donnez aux employés une permission limitée, disons des remises jusqu'à 10 % dans leur propre salle, avec un refus par défaut. Exigez une seconde personne pour les remises plus importantes, ce qui relève de la séparation des privilèges, et consignez qui a fait quoi. Échouez en mode fermé si le contrôle de permission est indisponible, en acceptant une file plus lente.

SOURCES

- OWASP Top Ten project (OWASP Top 10:2025): https://owasp.org/www-project-top-ten/
- NIST SP 800-218: Secure Software Development Framework (SSDF) version 1.1: https://csrc.nist.gov/pubs/sp/800/218/final
- Saltzer and Schroeder, The Protection of Information in Computer Systems (1975): https://doi.org/10.1109/PROC.1975.9939
- OWASP SAMM: Software Assurance Maturity Model: https://owasp.org/www-project-samm/
- OWASP DevSecOps Guideline: Dynamic Application Security Testing: https://owasp.github.io/www-project-devsecops-guideline/latest/02b-Dynamic-Application-Security-Testing
- OWASP: Fuzzing: https://community.owasp.org/Fuzzing
- Miller, Fredriksen and So: An Empirical Study of the Reliability of UNIX Utilities: https://ftp1.cs.wisc.edu/par-distr-sys/papers/fuzz.pdf
- OSS-Fuzz documentation: https://google.github.io/oss-fuzz/

SECTION 8.2 SUR 8.8 · INTERMÉDIAIRE · 23 MIN DE LECTURE · DERNIÈRE RELECTURE : NOVEMBRE 2026

Modélisation des menaces : STRIDE, arbres d'attaque et diagrammes de flux de données

AVANT DE LIRE

Cette section s'appuie sur :

- 2.7 Sécurité, utilisabilité et autres qualités
- 4.3 Le modèle C4
- 8.1 Principes de sécurité

Termes déjà vus qu'elle utilise : Frontière de confiance, Diagramme de conteneurs, Webhook, API gateway (passerelle d'API), Enregistrement de décision d'architecture (ADR)

L'IDÉE EN BREF

Avant le départ d'une famille en vacances, quelqu'un fait généralement le tour de la maison en se demandant ce qui pourrait mal tourner : la porte de derrière est-elle fermée à clé, un voisin a-t-il une clé, la clé de secours est-elle sous le pot de fleurs, le courrier va-t-il s'entasser et montrer qu'il n'y a personne ? Puis on décide quoi faire pour chaque point : fermer la porte à clé, demander au voisin de relever le courrier, déplacer la clé. La modélisation des menaces, c'est ce même tour du propriétaire appliqué à une conception logicielle. L'équipe dessine comment les données circulent dans le système, se demande ce qu'un attaquant pourrait faire à chaque étape, et décide quoi corriger, quoi accepter et quoi supprimer.

CE QU'IL FAUT SAVOIR

- La modélisation des menaces répond aux quatre questions de Shostack : sur quoi travaillons-nous, qu'est-ce qui peut mal tourner, qu'allons-nous faire à ce sujet, et avons-nous fait un travail suffisant.
- Un diagramme de flux de données composé d'entités externes, de processus, de stockages de données, de flux de données et de frontières de confiance est le modèle habituel ; les flux qui franchissent des frontières de confiance méritent le plus d'attention.
- STRIDE (Kohnfelder et Garg, Microsoft, 1999) nomme six catégories de menaces, chacune portant atteinte à une propriété : l'usurpation d'identité ou spoofing (authenticité), la falsification ou tampering (intégrité), la répudiation (non-répudiation), la divulgation d'informations (confidentialité), le déni de service (disponibilité) et l'élévation de privilèges (autorisation).

- Les arbres d'attaque (Schneier, 1999) modélisent l'objectif d'un attaquant sous forme de nœuds ET et OU ; le chemin le moins coûteux à travers l'arbre montre où une mesure d'atténuation compte le plus.
- Évaluez les menaces selon leur probabilité et leur impact, avec des justifications écrites ; DREAD est tombé en désuétude parce que ses notes étaient subjectives, et CVSS évalue des vulnérabilités connues, pas des menaces de conception.
- Chaque menace se conclut par une atténuation, une élimination, un transfert ou une acceptation explicite par une personne qui en a l'autorité ; modélisez les menaces lors de la conception, puis à nouveau chaque fois que les frontières de confiance, les données ou l'authentification changent.

Ce qu'est la modélisation des menaces

La modélisation des menaces est l'activité qui consiste à examiner une conception pour trouver ce qui pourrait mal tourner du point de vue de la sécurité, et à décider quoi faire, avant que des attaquants ou des incidents ne le trouvent pour vous. C'est le principal moyen par lequel les principes de sécurité deviennent des décisions concrètes, et la réponse la plus directe à la catégorie OWASP de la conception non sécurisée : des failles qu'aucun codage correct, si soigné soit-il, ne peut corriger, parce que le contrôle nécessaire n'a jamais été prévu dans la conception.

Adam Shostack, qui a dirigé les travaux de modélisation des menaces chez Microsoft et écrit le livre de référence sur le sujet, ramène l'activité à quatre questions. **Sur quoi travaillons-nous ?** (un modèle du système, généralement un diagramme). **Qu'est-ce qui peut mal tourner ?** (les menaces, trouvées de façon systématique). **Qu'allons-nous faire à ce sujet ?** (atténuer, éliminer, transférer ou accepter chaque menace). **Avons-nous fait un travail suffisant ?** (revoir le modèle, les menaces et le suivi). Toute méthode qui répond aux quatre questions est de la modélisation des menaces ; la suite de cette section décrit les techniques que la plupart des équipes utilisent pour y répondre.

En 2020, un groupe de praticiens, dont Shostack, a publié le Threat Modeling Manifesto, sur le modèle du Manifeste agile. Il privilégie une culture qui consiste à trouver et à corriger les problèmes de conception plutôt que la conformité de pure forme, les personnes et la collaboration plutôt que les processus et les outils, un cheminement de compréhension plutôt qu'un instantané de sécurité ponctuel, le fait de pratiquer la modélisation des menaces plutôt que d'en parler, et l'amélioration continue plutôt qu'une livraison unique. Son message est que la modélisation des menaces est une habitude de l'équipe qui construit le système, et non un document produit une fois par an par des personnes extérieures.

Modéliser le système : diagrammes de flux de données et frontières de confiance

La première question exige un modèle du système qui montre où vont les données et où la confiance change. Le choix traditionnel est un diagramme de flux de données (DFD) comportant cinq types d'éléments : les **entités externes** (des personnes ou des systèmes hors de votre contrôle, dessinés sous forme de rectangles), les **processus** (du code que vous exécutez, dessiné sous forme de cercles ou d'ellipses), les **stockages de données** (bases de données, fichiers, files et caches, dessinés entre deux lignes parallèles), les **flux de données** (des flèches montrant les données qui circulent entre les autres éléments) et les **frontières de confiance** (des lignes ou des cadres en pointillés là où le niveau de confiance change).

Les DFD conviennent à la modélisation des menaces parce que les attaques suivent les données : chaque flux qui franchit une frontière de confiance est un endroit où un attaquant peut injecter, intercepter ou rejouer quelque chose. Un diagramme de conteneurs C4 du chapitre 4 est un bon point de départ : ajoutez les frontières de confiance, faites de chaque flèche un flux de données avec un sens et une étiquette, et incluez les stockages et les files que les diagrammes de conteneurs laissent parfois implicites. Gardez le diagramme au niveau où l'on trouve les menaces : un diagramme par fonctionnalité ou par flux important est plus utile qu'un seul diagramme géant de tout le système.

Trouver les menaces avec STRIDE

STRIDE a été créé chez Microsoft en 1999 par Loren Kohnfelder et Praerit Garg, comme moyen mnémotechnique pour aider les développeurs à réfléchir aux menaces qui pèsent sur leurs produits. Chaque lettre nomme une catégorie de menaces, et chaque catégorie correspond à l'atteinte à une propriété de sécurité, ce qui en fait l'utilité : c'est une incitation systématique, pas une liste d'attaques connues.

STRIDE et la propriété à laquelle chaque menace porte atteinte

Menace	Propriété atteinte	Exemple de livraison de repas	Mesures d'atténuation typiques
Spoofing (usurpation d'identité)	Authenticité	Un attaquant se connecte en tant que client grâce à un mot de passe obtenu par bourrage d'identifiants ; un processus malveillant appelle Paiements en se faisant passer pour Commandes.	Authentification forte (MFA, passkeys), limites de débit, TLS mutuel et identité de charge de travail entre services.
Tampering (falsification)	Intégrité	Un client envoie un total de panier modifié ; quelqu'un altère un webhook de paiement ou une ligne de base de données.	Recalcul côté serveur, signatures et MAC, TLS, accès en écriture au moindre privilège.
Répudiation	Non-répudiation (imputabilité)	Un client nie avoir passé une commande ; un agent du support nie avoir émis un remboursement, et rien ne prouve le contraire.	Journaux d'audit liés à des identités authentifiées, horodatages, stockage qui révèle toute altération.
Information disclosure (divulgation d'informations)	Confidentialité	Une API renvoie les adresses de clients à des livreurs qui ne les livrent pas ; des messages d'erreur révèlent des détails internes.	Autorisation sur chaque objet, minimisation des données, chiffrement, messages d'erreur génériques.
Déni de service	Disponibilité	Des bots inondent l'étape de paiement pendant le pic du soir ; une requête de recherche coûteuse épuise la base de données.	Limites de débit et quotas, délais d'expiration, cloisonnements, autoscaling, limites de coût des requêtes.
Elevation of privilege (élévation de privilèges)	Autorisation	Un client appelle un endpoint réservé aux gérants de restaurant ; une injection permet à un attaquant d'exécuter du code avec les droits du service.	Autorisation avec refus par défaut, validation des entrées, sandboxing, moindre privilège.

Une façon pratique d'appliquer STRIDE est le **STRIDE par élément** : parcourir chaque élément du DFD et ne poser que les questions qui s'appliquent à son type. Les entités externes peuvent être usurpées et peuvent répudier ; les processus sont exposés aux six catégories ; les stockages de données à la falsification, à la divulgation d'informations et au déni de service (et

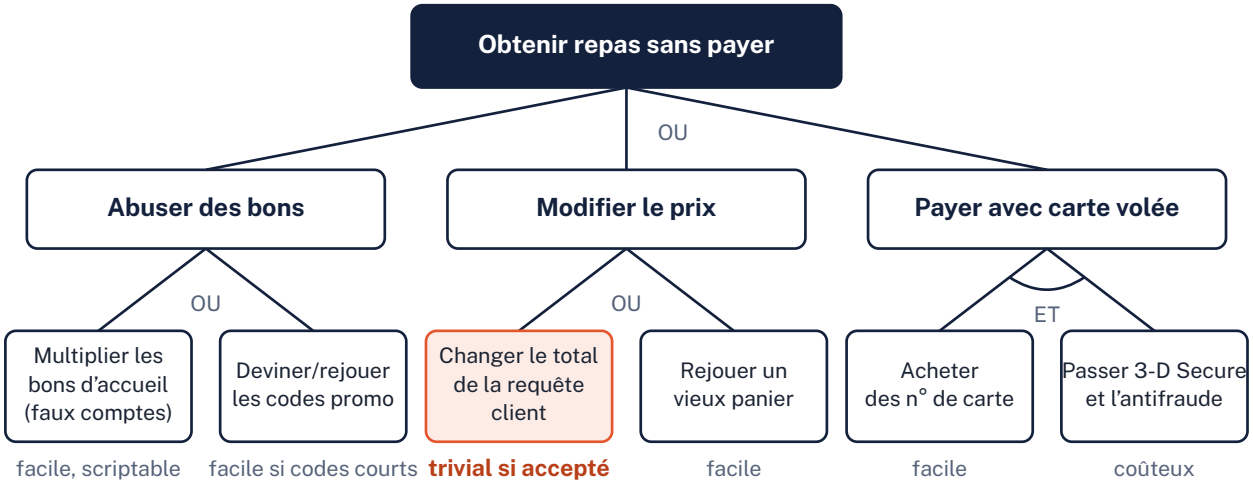
à la répudiation lorsque le stockage est un journal) ; les flux de données à la falsification, à la divulgation d'informations et au déni de service. Consacrer l'essentiel de l'attention aux flux qui franchissent des frontières de confiance permet de garder l'exercice ciblé.

STRIDE trouve des menaces ; il ne les classe pas et ne choisit pas les mesures d'atténuation. Ses catégories se recoupent aussi en pratique (une identité usurpée mène généralement à une divulgation d'informations ou à une élévation de privilèges), ce qui n'a pas d'importance : le but est de trouver la menace, pas de la classer parfaitement. Pour les menaces pesant sur la vie privée, comme le fait de relier ou d'identifier des personnes à partir de données d'apparence anodine, la méthode LINDDUN de la KU Leuven joue le rôle que STRIDE joue pour la sécurité ; elle est présentée avec la protection de la vie privée dès la conception.

Arbres d'attaque

Un arbre d'attaque, popularisé par Bruce Schneier dans un article de 1999 du *Dr. Dobb's Journal*, regarde le système du côté de l'attaquant. La racine est un objectif, comme obtenir des repas sans payer ; chaque enfant est un moyen d'atteindre son parent. Les enfants d'un nœud OU sont des alternatives ; tous les enfants d'un nœud ET sont nécessaires ensemble. Les feuilles peuvent être annotées de propriétés comme le coût, la compétence nécessaire ou le fait que l'étape soit possible ou non, et ces valeurs remontent l'arbre : le coût d'un nœud OU est celui de son enfant le moins coûteux, le coût d'un nœud ET la somme de ses enfants.

Les arbres d'attaque complètent bien STRIDE. STRIDE parcourt la conception élément par élément et assure une bonne couverture ; un arbre d'attaque suit un attaquant motivé et montre bien quel chemin est le moins coûteux, donc où une mesure d'atténuation rapporte le plus. Dans l'arbre ci-dessous, payer avec une carte volée est relativement coûteux parce qu'il faut deux choses à la fois, alors que modifier le prix dans la requête du client est trivial si le serveur lui fait confiance. Le chemin le moins coûteux détermine le risque, si bien que recalculer chaque prix sur le serveur vaut davantage que n'importe quel réglage de la lutte contre la fraude à la carte.



Le chemin le moins cher fixe le risque : le serveur doit recalculer chaque prix

Un arbre d'attaque pour un objectif d'attaquant ; la feuille la moins coûteuse montre où une mesure d'atténuation compte le plus.

SI VOUS DÉBUTEZ

Pas à pas : lire un arbre d'attaque

Le point qui fait le plus souvent trébucher ici est la façon dont un arbre d'attaque montre où une correction compte le plus. Un attaquant prend le chemin le moins coûteux, si bien qu'un objectif n'est pas plus difficile à atteindre que son chemin le moins coûteux. Remontez l'arbre du diagramme en partant du bas.

- Commencez par les feuilles, chacune représentant une seule étape avec un coût approximatif. Créer en masse des bons avec de faux comptes est bon marché ; faire passer une carte volée à travers 3-D Secure (la vérification supplémentaire de la banque au moment du paiement) et les contrôles antifraude est coûteux ; modifier le total dans la requête du client est trivial si le serveur lui fait confiance.
- « Payer avec une carte volée » est un nœud ET : l'attaquant a besoin à la fois des numéros de carte et d'un moyen de passer les contrôles. Son coût est la somme des deux, donc la seule étape coûteuse rend toute la branche coûteuse.
- « Modifier le prix » est un nœud OU : n'importe laquelle de ses feuilles suffit, donc son coût est celui de son enfant le moins coûteux. La branche reste triviale tant que le serveur fait confiance aux prix envoyés par le client.
- L'objectif au sommet est lui aussi un nœud OU, qui coûte ce que coûte sa branche la moins coûteuse : modifier le prix. C'est le vrai risque de la plateforme, aussi bons que soient ses contrôles antifraude.
- L'équipe fait donc recalculer chaque prix par le serveur. Un total modifié n'aboutit plus à rien, et cette feuille cesse d'être une option.
- Remontez à nouveau l'arbre. Les feuilles restantes les moins coûteuses, comme la création en masse de bons avec de faux comptes, fixent désormais le risque, et c'est là que la prochaine correction rapporte le plus.
- **La leçon :** rendre un chemin coûteux plus difficile ne change rien tant qu'il reste un chemin bon marché. Trouvez le chemin le moins coûteux, fermez-le, puis regardez à nouveau, car un attaquant motivé passe simplement au suivant.

Évaluer le risque

Un modèle de menaces trouve généralement plus de menaces qu'une équipe ne peut en traiter à la fois, si bien que chacune doit être évaluée. L'approche habituelle consiste à juger la **probabilité** (facilité, attrait, exposition) et l'**impact** (ce qui est perdu si la menace se réalise : argent, données à caractère personnel, disponibilité, réputation, exposition juridique) sur des échelles simples, par exemple faible, moyen et élevé, et à les combiner dans une matrice. La valeur de l'évaluation tient moins à la précision du chiffre qu'à une justification écrite que

d’autres peuvent contester. Les organisations dotées d’une gestion des risques mature utilisent parfois des méthodes quantitatives comme FAIR (Factor Analysis of Information Risk) pour exprimer le risque sous forme d’une fourchette de pertes attendues.

Deux raccourcis populaires méritent d’être connus parce qu’ils sont souvent mal utilisés. **DREAD** (damage, reproducibility, exploitability, affected users, discoverability, soit dommages, reproductibilité, exploitabilité, utilisateurs touchés et découvrabilité), autrefois utilisé chez Microsoft, notait chaque menace sur cinq facteurs (souvent de 1 à 10) et en faisait la moyenne. Il est tombé en désuétude, y compris chez Microsoft, parce que différentes personnes notaient la même menace de façon très différente et que la moyenne donnait une fausse impression de précision ; la découvrabilité, en particulier, récompense l’obscurité, ce qui contredit la conception ouverte. CVSS, le Common Vulnerability Scoring System maintenu par le FIRST (version 4.0, 2023), est le bon outil pour une autre tâche : évaluer la gravité d’une vulnérabilité connue dans un produit, comme une faille dans une bibliothèque. Son score de base décrit la vulnérabilité, pas le risque d’une menace de conception dans votre contexte ; il ne faut donc pas l’utiliser pour classer les résultats d’un modèle de menaces.

Chaque menace se conclut par l’une de quatre décisions : **atténuer** (ajouter un contrôle), **éliminer** (supprimer la fonctionnalité ou les données, souvent la meilleure option), **transférer** (laisser quelqu’un de mieux placé la porter, comme un prestataire de paiement qui détient les données de carte) ou **accepter** (consigner qu’une personne habilitée à le faire a accepté le risque résiduel, et pourquoi). L’acceptation non consignée est l’échec le plus courant : la menace a été trouvée, personne ne l’a corrigée, et personne n’a décidé de ne pas le faire.

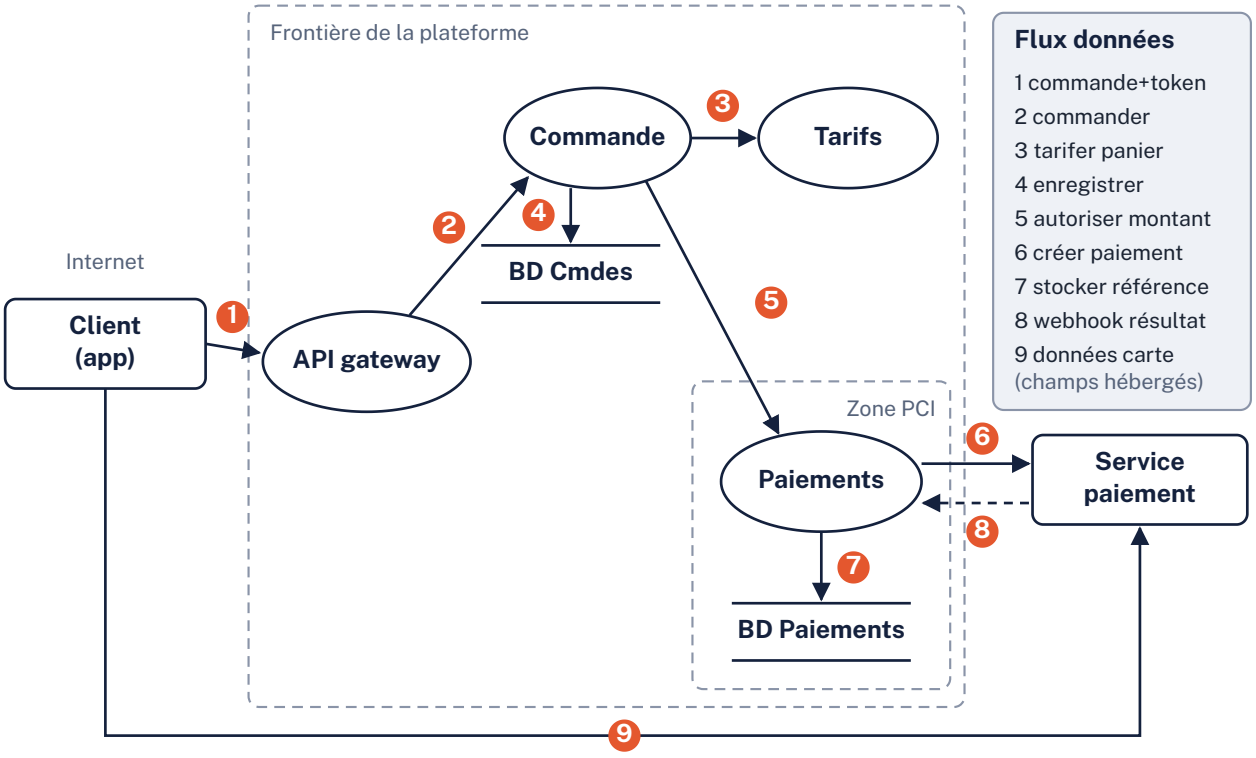
EXEMPLE COMMENTÉ

Exemple commenté : modéliser les menaces de l'étape de paiement

L'équipe de la plateforme de livraison de repas refond l'étape de paiement pour que les clients puissent payer dans l'app par carte ou avec un portefeuille électronique. L'architecte anime une séance de 90 minutes avec les développeurs des commandes et des paiements, un product owner et un référent sécurité. Le DFD ci-dessous répond à la première question, sur quoi travaillons-nous. Il montre deux frontières de confiance : la plateforme dans son ensemble, et une zone de paiement plus étroite et strictement restreinte (étiquetée zone PCI) autour du service des paiements et de sa base de données, qui détiennent des tokens et des références de paiement et communiquent avec le prestataire de paiement ; les isoler permet de garder réduit le périmètre PCI DSS de la plateforme. Les données de carte elles-mêmes (flux 9) vont directement des champs de paiement hébergés dans l'app au prestataire de paiement et n'entrent jamais dans la plateforme.

1. Sur quoi travaillons-nous ?	Le DFD de l'étape de paiement ci-dessous : app client, API gateway, Commandes, Tarification, Paiements, deux bases de données et le prestataire de paiement, avec deux frontières de confiance (la plateforme et la zone PCI) et neuf flux numérotés.
2. Qu'est-ce qui peut mal tourner ?	STRIDE par élément sur chaque flux qui franchit une frontière (1, 5, 6, 8 et 9) et sur l'API des commandes, plus l'arbre d'attaque présenté plus haut pour obtenir des repas sans payer.
3. Qu'allons-nous faire à ce sujet ?	Chaque menace est évaluée et reçoit une décision (atténuer, éliminer, transférer ou accepter) et un responsable ; les plus importantes figurent dans le tableau ci-dessous.
4. Avons-nous fait un travail suffisant ?	Une revue un mois plus tard compare le modèle à ce qui a été construit et vérifie que chaque risque élevé a été atténué ou accepté par écrit.

Le moyen le moins coûteux de protéger les numéros de carte est de ne pas les détenir : le flux 9 les garde entièrement hors de la plateforme, ce qui élimine des menaces au lieu de les atténuer.



Le diagramme de flux de données de l'étape de paiement, avec ses deux frontières de confiance et ses flux de données numérotés.

Menaces choisies dans le modèle de menaces de l'étape de paiement

ID et emplacement	STRIDE	Menace	Risque	Décision et atténuation
T1, flux 1	S	Le bourrage d'identifiants permet de prendre le contrôle de comptes qui ont des cartes enregistrées.	Élevé	Atténuer : passkeys, ou MFA sur les nouveaux appareils ; limites de débit par compte et par IP ; vérification des mots de passe ayant fuité.
T2, flux 1 et 2	T	Le client envoie un prix, une remise ou des frais de livraison modifiés.	Élevé	Atténuer : Commandes ignore les prix du client ; Tarification recalcule à partir des cartes et des bons ; le paiement est créé pour le total calculé par le serveur.
T3, flux 5	S, E	Un autre service interne, ou un attaquant à l'intérieur du réseau, demande à Paiements d'autoriser ou de rembourser des montants arbitraires.	Élevé	Atténuer : TLS mutuel avec identité de charge de travail ; Paiements n'accepte les autorisations que de l'identité de Commandes, liées à un ID de commande et à un montant.
T4, flux 8	S, T	Un faux webhook « paiement réussi » marque comme payée une commande impayée.	Élevé	Atténuer : vérifier la signature et l'horodatage du prestataire ; confirmer le statut auprès de l'API du prestataire avant de libérer la commande ; traiter chaque événement une seule fois.
T5, `GET /orders/{id}`	I	L'énumération des ID de commande révèle les adresses d'autres clients (autorisation défaillante au niveau objet).	Élevé	Atténuer : contrôle de propriété à chaque requête ; identifiants aléatoires uniquement comme mesure secondaire.

ID et emplacement	STRIDE	Menace	Risque	Décision et atténuation
T6, BD Commandes	I	Des enregistrements de commandes contenant des adresses fuient depuis une sauvegarde ou via un compte d'analyse aux droits trop larges.	Moyen	Atténuer : chiffrement au repos avec des clés KMS, comptes au moindre privilège, sauvegardes dans un compte distinct.
T7, flux 1	D	Des bots épuisent la capacité de l'étape de paiement au pic du soir.	Moyen	Atténuer : limites de débit et détection des bots en périphérie, capacité réservée à l'étape de paiement, file d'attente virtuelle si nécessaire.
T8, Commandes	R	Un client conteste avoir passé une commande, et il n'existe aucune preuve.	Faible	Atténuer : événement d'audit avec ID utilisateur, appareil, heure et détails de la commande, conservé conformément à la politique.
T9, flux 9	I	Des numéros de carte transitant par la plateforme feraient entrer chaque composant dans le périmètre PCI DSS.	Conception	Éliminer et transférer : les champs hébergés envoient les données de carte directement au prestataire ; la plateforme ne stocke que des tokens et des références.

Les réponses aux deux dernières questions sont aussi importantes que le tableau. Chaque mesure d'atténuation devient un élément du backlog avec un responsable ; T3 et T4 deviennent aussi des scénarios de sécurité assortis de tests automatisés (un webhook avec une signature invalide doit être rejeté ; un appel à Paiements depuis toute identité autre que Commandes doit échouer). Les deux décisions architecturales, les champs de paiement hébergés et le TLS mutuel entre Commandes et Paiements, sont consignées sous forme d'ADR. Pour la quatrième question, l'équipe revoit le modèle un mois plus tard par rapport à ce qui a réellement été construit, et le référent sécurité vérifie que chaque élément à risque élevé a été soit atténué, soit explicitement accepté.

Quand et à quelle fréquence modéliser les menaces

La modélisation des menaces rapporte le plus au moment de la conception, quand changer une décision coûte un coup d'effaceur sur un tableau blanc. Elle doit ensuite être répétée chaque fois que quelque chose d'important change : une nouvelle frontière de confiance ou intégration externe, un nouveau type de données sensibles, une modification de l'authentification ou de l'autorisation, un nouvel environnement de déploiement, ou le passage d'un monolithe à des services. Une règle raisonnable consiste à faire de « ce changement modifie-t-il le modèle de menaces ? » une question des revues de conception et du modèle d'enregistrement de décision d'architecture, et à planifier une revue du modèle de chaque système au moins une fois par an et après chaque incident de sécurité.

L'effort doit être proportionné, dans l'esprit de l'approche guidée par les risques du chapitre 1. Un nouveau microservice qui traite des paiements mérite une séance d'une demi-journée avec un DFD complet ; ajouter un champ à une carte publique demande cinq minutes et un rapide coup d'œil STRIDE au flux modifié. Beaucoup d'équipes conservent leur DFD sous forme de code à côté du système (OWASP Threat Dragon et le Threat Modeling Tool de Microsoft sont des outils de diagrammes gratuits, et des bibliothèques comme OWASP pytm génèrent des diagrammes et des menaces candidates à partir du code), afin que le modèle soit relu et mis à jour avec le système qu'il décrit.

ATTENTION

Erreurs courantes de modélisation des menaces

- Vouloir modéliser toute l'entreprise d'un coup ; commencez par une fonctionnalité, un flux ou un service.
- La faire une seule fois pour un audit et classer le document ; le modèle doit évoluer avec le système.
- La laisser à la seule équipe de sécurité ; les personnes qui construisent le système savent où sont ses points faibles.
- Spéculer sur l'identité de l'attaquant au lieu de se demander ce qui peut mal tourner dans la conception.
- Énumérer des menaces sans décisions, sans responsables ni suivi, ou accepter des risques sans consigner qui les a acceptés.
- Accepter des centaines de menaces génériques produites par un outil sans juger lesquelles s'appliquent.

POUR ALLER PLUS LOIN

Autres méthodes

PASTA (Process for Attack Simulation and Threat Analysis, décrite par Tony UcedaVélez et Marco Morana en 2015) est une méthode en sept étapes, centrée sur le risque, qui part des objectifs métier et aboutit à la simulation d'attaques ; elle convient aux organisations qui veulent une modélisation des menaces étroitement liée au risque métier. MITRE ATT&CK est une base de connaissances des tactiques et techniques réelles des attaquants, plus utile pour l'ingénierie de la détection dans les opérations de sécurité que pour la conception. Les systèmes qui intègrent des modèles d'apprentissage automatique ajoutent des menaces comme l'injection de prompt et l'empoisonnement des données d'entraînement, traitées au chapitre 12.

VÉRIFIEZ VOTRE COMPRÉHENSION

▼ Quelles catégories STRIDE s'appliquent à un flux de données dans le STRIDE par élément ?

La falsification, la divulgation d'informations et le déni de service : des données en transit peuvent être modifiées, lues ou bloquées.

▼ Pourquoi CVSS n'est-il pas le bon outil pour classer les menaces trouvées dans un modèle de menaces ?

CVSS note la gravité d'une vulnérabilité connue précise dans un produit ; il ne tient pas compte de votre conception, de vos actifs ni de votre contexte, et ne mesure donc pas le risque d'une menace au niveau de la conception.

▼ Dans un arbre d'attaque, comment le coût d'un nœud ET se calcule-t-il à partir de ses enfants, et pourquoi est-ce important ?

C'est la somme des coûts de ses enfants, car toutes les étapes sont nécessaires. Les nœuds ET sont donc généralement des chemins coûteux, tandis que l'enfant le moins coûteux d'un nœud OU fixe le risque.

▼ Quel est l'échec le plus courant à la quatrième étape de la modélisation des menaces ?

Des menaces trouvées mais ni corrigées ni explicitement acceptées : personne n'en est responsable et personne n'a décidé de vivre avec.

▼ **À vous : L'étape de paiement a un webhook par lequel le prestataire de paiement signale la réussite. À l'aide de STRIDE, nommez la menace si n'importe qui peut y envoyer des données, une autre menace, et une mesure d'atténuation pour chacune. Comment votre travail se termine-t-il pour chaque menace ?**

Usurpation : un faux message de réussite marque une commande impayée comme payée. Atténuez-la en vérifiant la signature du prestataire et en confirmant le paiement auprès de lui. Déni de service : inonder le point d'accès bloque les vraies confirmations ; atténuez-le avec une limitation de débit et une file. La falsification du montant est couverte par la signature. Le point d'accès franchit une frontière de confiance : il mérite donc de l'attention. Chaque menace doit finir atténuée, éliminée, transférée ou explicitement acceptée par une personne habilitée.

SOURCES

- Threat Modeling Manifesto (2020): https://www.threatmodelingmanifesto.org/
- OWASP Threat Modeling Cheat Sheet: https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html
- Microsoft Threat Modeling Tool: STRIDE threat categories: https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats
- Bruce Schneier, Attack Trees (Dr. Dobb's Journal, December 1999): https://www.schneier.com/academic/archives/1999/12/attack_trees.html
- FIRST: Common Vulnerability Scoring System: https://www.first.org/cvss/

SECTION 8.3 SUR 8.8 · AVANCÉ · 29 MIN DE LECTURE · DERNIÈRE RELECTURE : NOVEMBRE 2026

Identité et accès : authentification, autorisation, OAuth 2.0 et OpenID Connect

AVANT DE LIRE

Cette section s'appuie sur :

- 0.4 Comment les ordinateurs communiquent
- 0.6 API et intégration
- 8.1 Principes de sécurité

Termes déjà vus qu'elle utilise : Jeton d'accès (access token), API gateway (passerelle d'API), TLS et HTTPS, HTTP

L'IDÉE EN BREF

Pensez à un grand immeuble de bureaux. À l'accueil, vous montrez une pièce d'identité avec photo et recevez un badge visiteur : c'est l'authentification, prouver qui vous êtes. Le badge ouvre ensuite certaines portes et pas d'autres : c'est l'autorisation, décider de ce que vous avez le droit de faire. Si vous envoyez un coursier récupérer un colis pour vous, vous ne lui donnez pas votre propre badge ; l'accueil lui remet un laissez-passer limité qui n'ouvre que le local du courrier, et seulement aujourd'hui. OAuth fonctionne comme ce laissez-passer limité, pour les applications. Et les machines de l'immeuble elles-mêmes, comme les ascenseurs et le système d'alarme, ont elles aussi besoin de leurs propres badges, pour pouvoir se prouver les unes aux autres qu'elles sont authentiques.

CE QU'IL FAUT SAVOIR

- L'authentification établit qui appelle et gagne à être centralisée dans un seul fournisseur d'identité ; l'autorisation décide de ce que l'appelant peut faire sur quel objet et doit être appliquée là où se trouvent les données, dans chaque service.
- La norme NIST SP 800-63B (révision 4, 2025) privilégie la longueur aux règles de composition, exige une vérification par liste de blocage contre les mots de passe courants et compromis, interdit les changements périodiques forcés et attend des options résistantes à l'hameçonnage, comme les passkeys.
- Les sessions côté serveur sont faciles à révoquer ; les JWT autonomes passent à l'échelle mais ne peuvent pas être révoqués avant leur expiration : gardez-les donc de courte durée, fixez l'algorithme, vérifiez l'émetteur, l'audience et l'expiration, et ne les stockez pas dans localStorage.

- OAuth 2.0 (RFC 6749) délègue un accès limité au moyen de jetons ; les applications des utilisateurs doivent utiliser le flux de code d'autorisation avec PKCE (RFC 7636), les services l'octroi par identifiants client (client credentials) ; OAuth 2.1 est toujours un Internet-Draft en 2026.
- OpenID Connect ajoute un jeton d'identité (ID token) qui indique au client qui s'est connecté ; le jeton d'identité est destiné au client, le jeton d'accès à l'API.
- RBAC, ABAC et le contrôle d'accès basé sur les relations (Google Zanzibar, 2019) répondent à des besoins différents ; les moteurs de politiques comme OPA séparent la décision de son application, et l'identité de charge de travail (SPIFFE, mTLS) remplace les secrets partagés entre services.

Authentification et autorisation

Chaque requête adressée à la plateforme de livraison de repas soulève deux questions distinctes. L'authentification demande *qui est-ce ?* et y répond par des preuves : un mot de passe, une passkey, un certificat, un jeton signé. L'autorisation demande *l'appelant peut-il faire ceci, sur cet objet, maintenant ?* et y répond par une politique : ce client peut lire cette commande parce qu'il l'a passée ; ce livreur peut voir cette adresse de retrait parce que la livraison lui est affectée. HTTP les distingue lui aussi : `401 Unauthorized` signifie que la requête ne comporte pas d'authentification valide, tandis que `403 Forbidden` signifie que l'appelant est connu mais n'a pas le droit d'agir.

Cette distinction compte sur le plan architectural, car les deux problèmes se résolvent à des endroits différents. L'authentification gagne à être centralisée dans un seul fournisseur d'identité (IdP) auquel toutes les applications font confiance, de sorte que les règles de mot de passe, l'authentification multifacteur, la récupération de compte et le verrouillage soient mis en œuvre une seule fois. L'autorisation est par nature distribuée : seul le service des commandes sait à qui appartient une commande, si bien que la décision finale concernant une commande doit être prise là, même si des vérifications grossières ont lieu plus tôt, au niveau de l'API gateway (passerelle d'API). La plupart des violations du contrôle d'accès recensées dans l'OWASP Top 10 viennent de la confusion entre les deux : un endpoint qui vérifie que le jeton est valide, puis renvoie n'importe quel objet dont on lui donne l'identifiant.

Mots de passe, authentification multifacteur et passkeys

Les mots de passe restent l'authentificateur le plus courant, et les Digital Identity Guidelines du NIST sont les règles les plus largement suivies en la matière. La révision 4 de la SP 800-63B, publiée en juillet 2025, renverse plusieurs habitudes de longue date. La longueur compte davantage que la composition : un mot de passe utilisé comme seul facteur doit compter au moins 15 caractères (au moins 8 lorsqu'il n'est utilisé que dans le cadre d'une authentification

multifacteur), et les vérificateurs devraient accepter au moins 64 caractères, espaces et caractères Unicode compris. Les vérificateurs ne doivent pas imposer de règles de composition, comme l'obligation d'inclure un chiffre et un symbole, qui rendent les mots de passe plus difficiles à retenir sans les rendre beaucoup plus difficiles à deviner. Ils doivent vérifier les nouveaux mots de passe par rapport à une liste de blocage de mots de passe courants, prévisibles et compromis. Ils ne doivent pas imposer de changements périodiques, seulement un changement en cas de preuve de compromission. Ils doivent autoriser les gestionnaires de mots de passe et devraient autoriser le collage, ne doivent pas proposer d'indices de mot de passe lisibles avant la connexion et ne doivent pas demander de réponses à des questions de sécurité. Les tentatives échouées doivent faire l'objet d'une limitation de débit (rate limiting).

L'authentification multifacteur (MFA) combine des facteurs de natures différentes : quelque chose que vous savez (un mot de passe), quelque chose que vous possédez (un téléphone ou une clé de sécurité) et quelque chose que vous êtes (une empreinte digitale, généralement utilisée seulement pour déverrouiller un appareil). Elle déjoue l'essentiel du bourrage d'identifiants (credential stuffing), où des attaquants essaient des couples nom d'utilisateur et mot de passe divulgués par d'autres sites. Tous les seconds facteurs ne se valent pas : les codes à usage unique envoyés par SMS peuvent être volés par échange de carte SIM (SIM swapping), et aucun code à usage unique, qu'il vienne d'un SMS ou d'une application d'authentification, ne résiste à l'hameçonnage, car l'utilisateur peut être amené par la ruse à le saisir sur un faux site qui le relaie immédiatement. Les lignes directrices du NIST exigent donc que les services au niveau d'assurance de l'authentification 2 proposent au moins une option résistante à l'hameçonnage, et que ceux du niveau 3 en utilisent une.

Une passkey (clé d'accès) est l'option résistante à l'hameçonnage que la plupart des utilisateurs peuvent réellement utiliser. Les passkeys sont des identifiants FIDO2, fondés sur la Web Authentication API du W3C (WebAuthn) et sur le Client to Authenticator Protocol (CTAP) de la FIDO Alliance. À l'enregistrement, l'appareil de l'utilisateur crée une nouvelle paire de clés pour ce site web ; le serveur ne stocke que la clé publique. À la connexion, le serveur envoie un défi aléatoire, et l'appareil le signe avec la clé privée une fois que l'utilisateur l'a déverrouillé par empreinte digitale, reconnaissance faciale ou code PIN. Comme le navigateur lie l'identifiant à l'origine du site web, un site d'hameçonnage qui imite le vrai ne peut pas obtenir de signature valide, et comme le serveur ne détient aucun secret partagé, une violation de sa base de données ne fournit rien qu'un attaquant puisse rejouer. Les passkeys peuvent être liées à un appareil (sur une clé de sécurité matérielle) ou synchronisées entre les appareils d'un utilisateur par une plateforme ou un gestionnaire de mots de passe ; les recommandations 2025 du NIST acceptent les passkeys synchronisées au niveau 2, tandis que le niveau 3 exige une clé non exportable.

Le point faible architectural de tout dispositif d'authentification est la **récupération de compte**. Si une passkey perdue peut être remplacée en cliquant sur un lien reçu par e-mail, la solidité de tout le système est celle du compte e-mail. Les parcours de récupération méritent leur propre modèle de menaces, des limites de débit, des notifications sur les autres appareils de l'utilisateur et, pour les comptes à forte valeur, des délais ou des vérifications humaines.

Sessions et jetons

Une fois l'utilisateur authentifié, le système doit reconnaître ses requêtes suivantes sans lui redemander qui il est. Il existe deux conceptions de base. Dans une **session côté serveur**, le serveur stocke la session (utilisateur, expiration, éventuellement rôles) et donne au navigateur un long identifiant de session aléatoire dans un cookie. Dans une conception à **jeton autonome**, l'émetteur signe un jeton qui porte les revendications (claims), généralement un JSON Web Token (JWT, RFC 7519), et tout service qui détient la clé publique de l'émetteur peut le valider sans rien consulter.

Sessions côté serveur et jetons autonomes comparés

Aspect	Session côté serveur	Jeton autonome (JWT)
Où se trouve l'état	Dans un magasin de sessions sur le serveur ; le navigateur ne détient qu'un identifiant aléatoire.	Dans le jeton signé ; le serveur ne conserve rien par utilisateur.
Révocation	Supprimer la session : effet immédiat.	Impossible avant l'expiration sans liste de refus ni consultation, si bien que les durées de vie doivent être courtes.
Passage à l'échelle entre services	Nécessite un magasin de sessions partagé ou un routage persistant (sticky).	Tout service peut valider la signature localement.
Taille sur le réseau	Un petit cookie.	De quelques centaines d'octets à quelques kilo-octets à chaque requête.
Usage typique	Sessions de navigateur pour les applications web et les backends for frontends.	Jetons d'accès pour les API, les applications mobiles et les appels entre services.
Principaux risques	Fixation de session et falsification de requête intersites (atténuées en régénérant l'identifiant à la connexion et par les cookies SameSite).	Vol par injection de script s'il est conservé dans le stockage du navigateur ; erreurs de validation ; revendications périmées.

Pour les applications de navigateur, une session côté serveur dans un cookie marqué `HttpOnly` (invisible pour JavaScript), `Secure` (HTTPS uniquement) et `SameSite` est difficile à battre. Les applications monopages utilisent de plus en plus un **backend for frontend** (BFF) : le navigateur ne détient qu'un cookie de session pour son propre petit backend, qui détient les jetons OAuth et appelle les API pour son compte, une approche que recommandent les lignes directrices du groupe de travail OAuth de l'IETF pour les applications fonctionnant dans le navigateur. Les JWT excellent entre services et dans les applications mobiles, où une vérification sans état à chaque saut évite un goulet d'étranglement central.

ATTENTION

Les pièges des JWT

- **Pas de révocation.** Un JWT volé reste valide jusqu'à son expiration. Gardez les jetons d'accès de courte durée (quelques minutes), utilisez des jetons de rafraîchissement (refresh tokens) que le serveur d'autorisation peut révoquer, et tenez une liste de refus des identifiants de jetons pour les urgences, ou utilisez des jetons opaques vérifiés par introspection (RFC 7662) là où une révocation immédiate compte.
- **Confusion d'algorithme.** Certaines bibliothèques acceptaient autrefois l'algorithme indiqué dans l'en-tête du jeton, y compris `none`, ou vérifiaient un jeton RS256 comme un jeton HS256 en utilisant la clé publique comme secret HMAC, ce qui permettait à des attaquants de forger des jetons. Fixez toujours les algorithmes attendus dans le code, comme le conseille la RFC 8725 (bonnes pratiques actuelles pour les JWT).
- **Stocker les jetons dans `localStorage`.** Tout script exécuté sur la page peut le lire, si bien qu'une seule faille de script intersite (cross-site scripting) divulgue le jeton de chaque utilisateur. Préférez des cookies HttpOnly via un BFF, ou gardez les jetons en mémoire et liez-les au client avec des jetons liés à l'expéditeur (sender-constrained tokens), comme DPoP (RFC 9449).
- **Signé ne veut pas dire chiffré.** La charge utile d'un JWT est simplement encodée en Base64url ; quiconque détient le jeton peut la lire. N'y mettez pas de données à caractère personnel.
- **Sauter la vérification des revendications.** Validez l'émetteur, l'audience et l'expiration, pas seulement la signature, sinon un jeton émis pour une API sera accepté par une autre.

```
import jwt  # PyJWT
from jwt import PyJWKClient

ISSUER = "https://id.fooddelivery.example/"
jwks = PyJWKClient(ISSUER + ".well-known/jwks.json")  # issuer's public keys

def verify_access_token(token: str) → dict:
    signing_key = jwks.get_signing_key_from_jwt(token)
    return jwt.decode(
        token,
        signing_key.key,
        algorithms=["RS256"],      # pinned: never read from the token header
        audience="orders-api",      # the token must be meant for this API
        issuer=ISSUER,
        options={"require": ["exp", "iat", "sub"]},
    )  # raises an exception on any failure: fail closed
```

Valider un jeton d'accès dans une API Python avec PyJWT : l'algorithme, l'audience et l'émetteur sont fixés par l'API, jamais tirés du jeton.

SI VOUS DÉBUTEZ

Pas à pas : un jeton que personne ne peut reprendre

L'idée la plus difficile de cette section est de comprendre pourquoi un jeton autonome ne peut pas simplement être annulé, et ce que font les concepteurs face à cela. Suivez un même vol dans chacune des deux conceptions du tableau.

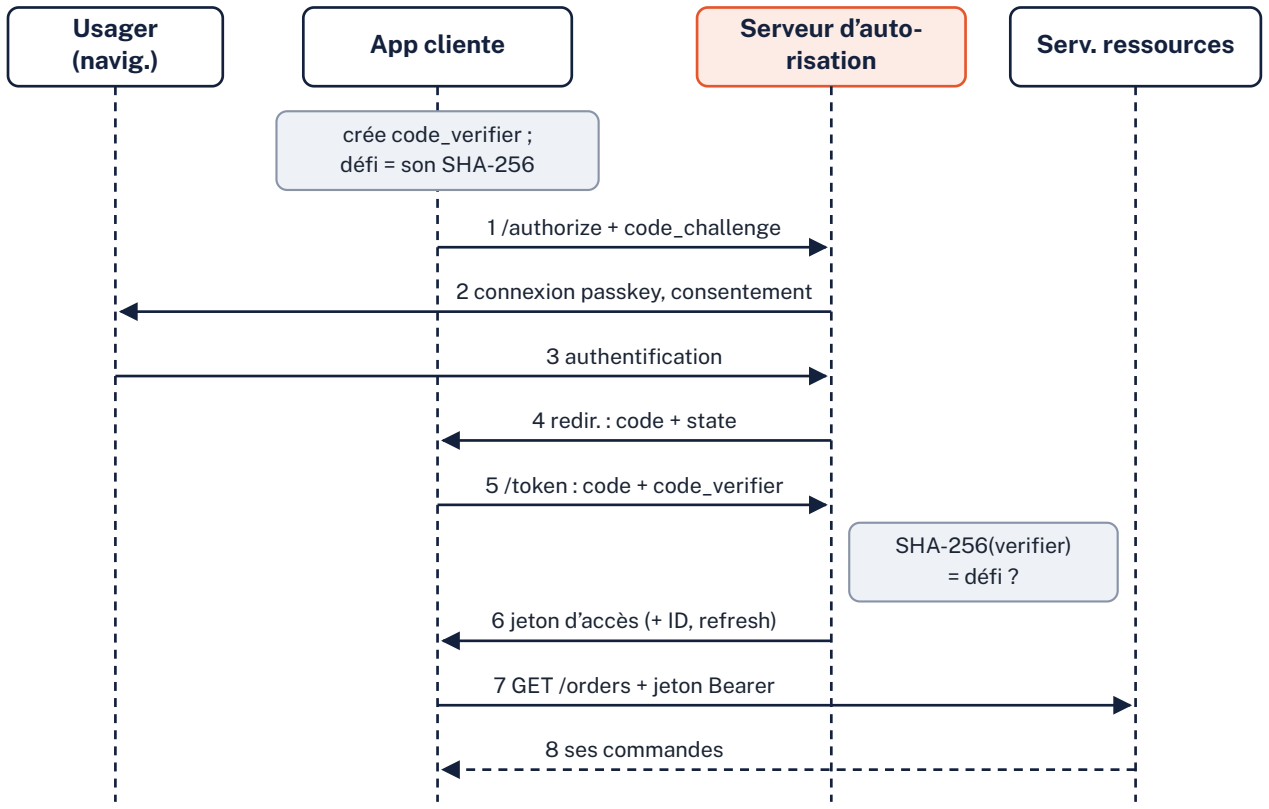
- Avec une session côté serveur, le serveur enregistre la connexion du client dans son magasin de sessions et ne donne au navigateur qu'un long identifiant aléatoire dans un cookie, qu'il consulte à chaque requête.
- Un attaquant vole l'identifiant. Dès que le vol est remarqué, la plateforme supprime l'enregistrement de session, si bien que la requête suivante de l'attaquant ne trouve rien et est refusée : l'annulation est immédiate.
- Avec l'autre conception, l'application mobile du client détient un JSON Web Token signé qui indique qui est le client et quand le jeton expire. Le service des commandes vérifie la signature, un peu comme un commerçant vérifie le filigrane d'un billet de banque, ainsi que l'émetteur, l'audience et l'expiration du jeton, puis lui fait confiance sans rien demander à personne.
- Le jeton est maintenant volé. Il n'y a aucun enregistrement à supprimer, si bien que chaque service continue de l'accepter, puisqu'il est toujours authentique, jusqu'à son expiration.
- D'où les remèdes décrits dans « Les pièges des JWT » : faites expirer les jetons d'accès en quelques minutes et renouvelez-les avec un jeton de rafraîchissement que l'émetteur peut révoquer ; là où l'annulation immédiate compte, tenez une liste de refus des jetons volés, ou utilisez des jetons que chaque service doit vérifier auprès de l'émetteur, ce qui réintroduit précisément la consultation que les jetons permettent d'éviter.
- C'est donc un compromis : les jetons permettent à chaque service de vérifier seul un appelant, ce qui convient aux applications mobiles et aux appels entre services ; les sessions sont faciles à annuler et conviennent aux applications de navigateur.
- **La leçon :** à lui seul, un jeton autonome est une promesse qui ne peut pas être retirée avant son expiration. Gardez la promesse courte, et n'ajoutez une consultation que là où l'annulation immédiate compte.

OAuth 2.0 : l'autorisation déléguée

OAuth 2.0 (RFC 6749, 2012) résout un problème précis : permettre à une application d'agir au nom d'un utilisateur sans connaître son mot de passe. Avant OAuth, un service qui voulait importer vos contacts vous demandait le mot de passe de votre messagerie électronique. OAuth définit quatre rôles : le **propriétaire de la ressource** (l'utilisateur), le **client** (l'application

qui veut obtenir l'accès), le **serveur d'autorisation** (qui authentifie l'utilisateur, lui demande son consentement et émet les jetons) et le **serveur de ressources** (l'API qui détient les données). Le client reçoit un **jeton d'accès** limité par des **portées** (scopes, comme `orders:read`), une durée de vie et une audience, et souvent un **jeton de rafraîchissement** qui permet d'obtenir de nouveaux jetons d'accès sans redemander à l'utilisateur.

OAuth définit plusieurs manières d'obtenir un jeton, appelées octrois (grants). Pour toute application qui agit pour un utilisateur, qu'il s'agisse d'une application web côté serveur, d'une application monopage ou d'une application mobile, l'octroi recommandé est le **flux de code d'autorisation avec PKCE** (Proof Key for Code Exchange, RFC 7636, 2015). Le client crée d'abord un `code_verifier` (vérificateur) aléatoire et n'envoie que son hachage SHA-256, le `code_challenge` (défi), lorsqu'il redirige l'utilisateur vers le serveur d'autorisation. Une fois que l'utilisateur s'est connecté et a donné son consentement, le serveur d'autorisation le redirige vers le client avec un code d'autorisation de courte durée. Le client échange ensuite ce code contre des jetons par une requête directe sur le canal arrière (back channel), en envoyant le vérificateur d'origine ; le serveur vérifie que son hachage correspond au défi. Un attaquant qui intercepte le code, par exemple une application mobile malveillante enregistrée pour la même adresse de redirection, ne peut pas l'échanger sans le vérificateur, qui n'a jamais quitté le client.



Le flux de code d'autorisation avec PKCE : seul le hachage du vérificateur transite dans les redirections du navigateur (le canal frontal) ; le vérificateur lui-même est envoyé directement à l'endpoint de jeton.

L'**octroi par identifiants client** (client credentials grant) couvre le cas où il n'y a pas d'utilisateur : un service obtient un jeton en son propre nom, par exemple le service d'affectation qui appelle l'API d'un partenaire cartographique, ou le système d'un partenaire qui appelle l'API de statut des commandes de la plateforme. Le client devrait s'authentifier auprès de l'endpoint de jeton avec une clé privée (une assertion JWT signée) ou avec le TLS mutuel (RFC 8705) plutôt qu'avec un secret partagé. Deux octrois plus anciens sont désormais déconseillés : l'**octroi implicite**, qui renvoyait les jetons dans l'URL du navigateur, et l'**octroi par mot de passe du propriétaire de la ressource**, qui faisait collecter le mot de passe de l'utilisateur par le client et allait ainsi à l'encontre du but même d'OAuth. La RFC 9700, l'OAuth 2.0 Security Best Current Practice publiée en janvier 2025, consigne ces leçons, notamment la correspondance exacte des adresses de redirection et les jetons liés à l'expéditeur.

OAuth 2.1 regroupe OAuth 2.0 et ces bonnes pratiques en un seul document : PKCE est obligatoire pour le flux de code d'autorisation, les octrois implicite et par mot de passe sont supprimés, les adresses de redirection doivent correspondre exactement et les jetons porteurs (bearer tokens) ne peuvent plus être envoyés dans les chaînes de requête des URL. Lors de la

dernière revue de cette section, c'était toujours un Internet-Draft de l'IETF, et non une RFC (vérifiez son statut actuel), mais ses règles décrivent déjà ce que font les serveurs d'autorisation bien gérés.

GUIDE DE DÉCISION

Guide de décision : quel flux OAuth 2.0 choisir ?

Pour chaque type d'appelant des API de livraison de repas.

Q1 **L'appareil est-il incapable d'afficher un navigateur, ou peu pratique pour la saisie, comme l'écran de cuisine d'un restaurant ?**

- **Oui** Utilisez l'octroi d'autorisation d'appareil (device authorization grant, RFC 8628) : l'appareil affiche un code court que l'utilisateur approuve sur son téléphone.
- **Non** Passez à la question suivante.

Q2 **Une personne est-elle présente, et l'application agit-elle en son nom ?**

- **Oui** Utilisez le flux de code d'autorisation avec PKCE, pour les applications web, monopages et mobiles sans distinction ; jamais les octrois implicite ou par mot de passe.
- **Non** Passez à la question suivante.

Q3 **S'agit-il d'un de vos propres services qui en appelle un autre au sein de votre plateforme ?**

- **Oui** Préférez l'identité de charge de travail (TLS mutuel avec des identités émises par la plateforme) ; si l'API attend des jetons OAuth, utilisez les identifiants client avec une authentification du client par clé.
- **Non** Passez à la question suivante.

Q4 **S'agit-il du système d'un partenaire qui appelle votre API en son propre nom ?**

- **Oui** Utilisez les identifiants client avec des jetons de courte durée et de portée étroite, un client par partenaire et des limites de débit propres à chaque partenaire.
- **Non** Réexaminez l'exigence : une conception dans laquelle votre client a besoin du mot de passe d'un utilisateur est un signal d'alarme.

OpenID Connect, authentification unique et SAML

OAuth émet des jetons d'accès pour les API ; il n'indique pas, à lui seul, au client qui est l'utilisateur. Utiliser un jeton d'accès comme preuve de connexion est une erreur bien connue, car un jeton émis pour une application peut être rejoué auprès d'une autre qui ne vérifie pas son audience. OpenID Connect (OIDC, Core 1.0, 2014) ajoute la couche d'identité manquante.

Avec la portée `openid`, le serveur d'autorisation, qui joue alors le rôle de fournisseur OpenID, renvoie aussi un **jeton d'identité** (ID token) : un JWT signé destiné au client, qui indique l'émetteur, l'identifiant stable de l'utilisateur (`sub`), l'audience (l'identifiant du client), les dates d'émission et d'expiration, un `nonce` qui le rattache à cette connexion et, facultativement, comment et quand l'utilisateur s'est authentifié. OIDC normalise aussi un endpoint d'informations sur l'utilisateur et des documents de découverte qui publient les endpoints et les clés de signature du fournisseur. La règle empirique : le jeton d'identité est destiné au client, pour savoir qui s'est connecté ; le jeton d'accès est destiné à l'API, pour autoriser les appels.

L'authentification unique (SSO) en découle naturellement : dès qu'un utilisateur a une session auprès du fournisseur d'identité, chaque application qui lui fait confiance peut le connecter sans lui redemander de mot de passe, et la désactivation du compte en un seul endroit coupe l'accès partout. En entreprise, le SSO utilise encore souvent **SAML 2.0** (une norme OASIS de 2005), qui échange des assertions XML signées par l'intermédiaire du navigateur. La plateforme de livraison de repas pourrait utiliser OIDC pour ses propres applications et accepter SAML de la part de grandes chaînes de restaurants qui veulent que leur personnel se connecte au portail des restaurants avec ses comptes d'entreprise. Le provisionnement et le déprovisionnement des comptes entre systèmes sont généralement automatisés avec SCIM (RFC 7643 et 7644).

Modèles d'autorisation : RBAC, ABAC et relations

L'authentification indique au système qui appelle ; un modèle d'autorisation décide de ce que l'appelant peut faire. Trois modèles dominent, et les systèmes réels les combinent généralement.

Trois modèles d'autorisation

Modèle	Comment il décide	Points forts	Faiblesses
Basé sur les rôles (RBAC)	Les permissions sont accordées à des rôles ; des rôles sont attribués aux utilisateurs. Formalisé par David Ferraiolo et Richard Kuhn au NIST en 1992.	Simple à comprendre et à auditer ; correspond à des fonctions comme celle d'agent du support.	Ne peut pas exprimer « seulement les siens » sans règles supplémentaires ; le découpage par établissement ou par marché mène à une explosion des rôles.
Basé sur les attributs (ABAC)	Une politique évalue des attributs du sujet, de la ressource, de l'action et de l'environnement ; décrit dans la NIST SP 800-162 (2014).	Très expressif : marché, classification des données, heure, état de sécurité de l'appareil, montant.	Plus difficile de répondre à « qui peut accéder à ceci ? » ; les politiques peuvent devenir opaques et lentes.
Basé sur les relations (ReBAC)	L'accès suit des relations stockées sous forme de graphe : un utilisateur gère un établissement, un établissement appartient à un restaurant, une commande appartient à un établissement.	Naturel pour le partage et les hiérarchies ; passe à l'échelle jusqu'à des permissions fines, objet par objet.	Nécessite un magasin de relations dédié et cohérent, et une modélisation soignée.

Le contrôle d'accès basé sur les relations est surtout connu par l'article de Google de 2019 sur **Zanzibar**, le système d'autorisation mondial qui sous-tend Google Drive, Calendar, YouTube et bien d'autres produits. Zanzibar stocke des tuples de relation de la forme *object#relation@user*, évalue les permissions en les suivant, et utilise des jetons de cohérence pour qu'une vérification de permission ne soit jamais évaluée sur des données plus anciennes que le contenu qu'elle protège ; c'est ce qu'on appelle le problème du « nouvel ennemi » (new enemy problem) : retirer quelqu'un d'un document puis y ajouter un contenu sensible ne doit pas lui permettre de le voir. Parmi les systèmes open source qui s'en inspirent figurent SpiceDB, OpenFGA et Ory Keto.

Quel que soit le modèle, il est utile de séparer le point de décision de politique (PDP), qui évalue la politique et répond « autoriser » ou « refuser », du **point d'application de politique** (PEP), le code de la passerelle ou du service qui pose la question et applique la réponse. Les moteurs de politiques généralistes comme Open Policy Agent (OPA, projet diplômé de la Cloud Native Computing Foundation, dont les politiques s'écrivent dans son langage Rego) et Cedar (publié en open source par Amazon Web Services en 2023) permettent aux politiques de vivre dans des fichiers versionnés, d'être testées comme du code et d'être évaluées dans le processus ou

dans un sidecar, au plus près du service. Sur la plateforme, un découpage en couches raisonnable est le suivant : l'API gateway rejette les requêtes sans jeton valide ou sans la bonne portée (contrôle grossier), et chaque service tranche les questions au niveau de l'objet avec des données que lui seul possède (contrôle fin). La faille Broken Object Level Authorization (autorisation défaillante au niveau de l'objet) est le premier risque de l'OWASP API Security Top 10 (édition 2023), précisément parce que le contrôle fin est celui que les équipes oublient.

EXEMPLE COMMENTÉ

Exemple commenté : qui peut modifier une carte, marquer une commande comme prête ou voir une adresse de retrait

Les restaurants de la plateforme vont du café unique à des chaînes de plusieurs dizaines d'établissements. Exigences : les propriétaires de chaîne gèrent tous les établissements ; les gérants d'établissement modifient la carte de leur propre établissement ; le personnel de cuisine marque comme prêtes les commandes de son établissement ; un livreur ne voit les informations de retrait que pour les commandes qui lui sont affectées ; les gérants peuvent accorder des remboursements commerciaux jusqu'à £20.

Première tentative : RBAC	Rôles `owner`, `manager`, `staff`, `courier`. Chaque vérification a alors besoin de savoir « pour quel établissement ? » ou « quelle commande ? », ce qui mène à des rôles comme `manager_branch_1742` : des milliers de rôles, un par établissement et par fonction.
Tuples ReBAC	`restaurant:bella#owner@user:ana` ; `branch:bella-soho#parent@restaurant:bella` ; `branch:bella-soho#manager@user:ben` ; `branch:bella-soho#staff@user:kai` ; `order:9812#branch@branch:bella-soho` ; `order:9812#courier@user:cara`
Règles du modèle	`branch.edit_menu` = gérant, ou propriétaire du restaurant parent. `order.mark_ready` = personnel ou gérant de l'établissement de la commande. `order.view_pickup` = le livreur affecté, ou le personnel de l'établissement de la commande.
Vérifications	ben peut-il modifier la carte de bella-soho ? Oui (gérant). Et ana ? Oui (propriétaire du parent). cara peut-elle voir le retrait de la commande 9812 ? Oui, jusqu'à ce qu'Affectation supprime son tuple de livreur à la fin de la livraison ; ensuite, non.
ABAC pour le plafond de remboursement	La règle de remboursement dépend d'un attribut de la requête, et non d'une relation : le moteur de politiques autorise `refund` si l'utilisateur gère l'établissement de la commande et si le montant ne dépasse pas £20.
Lieu d'application	La passerelle vérifie la validité et la portée du jeton ; les services de la carte, des commandes et d'affectation demandent chacun au PDP la décision au niveau de l'objet avant d'agir.

Les relations répondent directement aux questions d'audit : lister toutes les personnes qui peuvent modifier la carte d'un établissement est une requête sur le graphe, et non une revue de code.

Les données de relation ont désormais elles aussi un responsable : Affectation écrit les tuples de livreur, le portail des restaurants écrit les tuples du personnel, et les tuples périmés sont un bug de sécurité.

Identité de charge de travail : quand l'appelant est un service

Les services s'authentifient les uns auprès des autres bien plus souvent que les personnes ne se connectent, et pourtant les identifiants des services sont souvent les plus faibles : une clé d'API de longue durée dans une variable d'environnement, partagée par plusieurs services et jamais renouvelée. Si cette clé fuit par un journal, un dépôt ou une image de conteneur, l'attaquant devient le service. L'identité de charge de travail (workload identity) remplace ces secrets par des identités que la plateforme émet pour la charge de travail en cours d'exécution, après avoir vérifié ce qu'est cette charge de travail.

Le standard ouvert est **SPIFFE** (Secure Production Identity Framework for Everyone), un projet diplômé de la Cloud Native Computing Foundation. Il nomme chaque charge de travail par une URI comme `spiffe://fooddelivery.example/ns/prod/sa/orders` et émet un document de courte durée, renouvelé automatiquement, qui prouve cette identité ; ce document, appelé SVID, est généralement un certificat X.509 (ou un JWT). Son implémentation de référence, SPIRE, atteste le nœud et la charge de travail (par exemple son espace de noms et son compte de service Kubernetes) avant d'émettre l'identité. Les services utilisent ensuite les certificats pour le TLS mutuel, si bien que le service des paiements sait que l'appelant est réellement le service des commandes, et peut autoriser en fonction de l'identité. Les plateformes cloud proposent des équivalents (rôles pour les charges de travail, identités managées, fédération d'identité de charge de travail), et les systèmes de CI comme GitHub Actions peuvent échanger un jeton OIDC signé contre des identifiants cloud de courte durée, si bien que les pipelines n'ont besoin d'aucune clé stockée. La façon dont ces identités servent à construire un réseau sans confiance implicite est le sujet du zero trust.

Accès administratif juste-à-temps et accès d'urgence (break-glass)

Un compte administrateur permanent est une cible permanente. Avec l'accès **juste-à-temps** (just-in-time), un ingénieur ne détient aucun privilège permanent, seulement une éligibilité : pour réparer une base de données de production, il active un rôle pour une durée limitée, donne une raison, passe l'authentification multifacteur et, pour les rôles sensibles, obtient une approbation, et chaque étape est journalisée. Microsoft Entra Privileged Identity Management en est un exemple. Les comptes d'**urgence** (break-glass) couvrent le jour où ce processus échoue, par exemple parce que le fournisseur d'identité est hors service. Les recommandations de Microsoft prévoient au moins deux comptes, tenus à l'écart des dépendances de connexion habituelles, protégés par une authentification résistante à l'hameçonnage, déclenchant une alerte à chaque utilisation et testés régulièrement. Chaque utilisation est ensuite examinée, et la voie d'urgence constitue elle-même un risque à gérer.

L’autorisation sur les topics de messages

L’authentification auprès du courtier ne suffit pas : les événements (section 6.4) déclenchent de vraies actions, si bien qu’un service capable de publier un faux événement `PaymentCaptured` peut faire expédier des commandes impayées. Accordez les permissions par topic et par sens : seul Paiements peut écrire dans ce topic, et seuls des consommateurs nommés peuvent le lire, ce qui limite aussi les personnes qui voient les données personnelles qu’il contient. Apache Kafka l’exprime sous forme de listes de contrôle d’accès indiquant qu’un principal peut effectuer une opération, comme Read ou Write, sur une ressource ; lorsqu’un authorizer est configuré, une ressource sans liste correspondante n’est accessible qu’aux super-utilisateurs, sauf si `allow.everyone.if.no.acl.found` est défini. Les valeurs par défaut diffèrent : dans RabbitMQ, les permissions de topic sont facultatives (opt-in). Vérifiez la documentation de votre courtier, et conservez ces règles sous gestion de versions.

VÉRIFIEZ VOTRE COMPRÉHENSION

▼ **Une API renvoie 403 à un appelant muni d’un jeton valide. Qu’est-ce que cela vous apprend ?**

L’appelant a été authentifié mais n’est pas autorisé pour cette action ou cet objet ; 401 signifierait une authentification absente ou invalide.

▼ **Contre quoi PKCE protège-t-il dans le flux de code d’autorisation ?**

Contre l’interception ou l’injection du code d’autorisation : sans le vérificateur de code, qui ne transite jamais dans les redirections du navigateur, un code volé ne peut pas être échangé contre des jetons.

▼ **Pourquoi un service ne devrait-il pas accepter un jeton d’identité OpenID Connect comme jeton d’accès à une API ?**

Le jeton d’identité est adressé au client (son audience est l’identifiant du client) pour lui indiquer qui s’est connecté ; les API ne devraient accepter que des jetons d’accès émis pour elles, dont l’audience et la portée sont vérifiées.

▼ **Pourquoi un rôle comme manager_branch_1742 laisse-t-il penser que le modèle d’autorisation n’est pas le bon ?**

Il encode une relation (gérant d’un établissement donné) sous forme de rôle, ce qui mène à une explosion des rôles ; des règles basées sur les relations ou sur les attributs l’expriment directement.

▼ À vous : Choisissez le flux OAuth 2.0 pour : l'application mobile native d'un restaurant utilisée par le personnel, et une tâche backend nocturne qui appelle l'API de paiements. Un développeur envoie aussi le jeton d'identité à l'API de commandes. Donnez votre avis sur les trois.

L'app mobile agit pour un utilisateur et ne peut pas garder de secret : utilisez donc le flux de code d'autorisation avec PKCE. Le job nocturne n'a pas d'utilisateur : utilisez donc les identifiants client ou, mieux, l'identité de charge de travail. Le développeur se trompe : le jeton d'identité indique au client qui s'est connecté, et l'API doit recevoir le jeton d'accès, en vérifiant l'émetteur, l'audience et l'expiration. Gardez des jetons de courte durée, et laissez chaque service effectuer ses propres vérifications au niveau des objets.

SOURCES

- RFC 6749: The OAuth 2.0 Authorization Framework: https://www.rfc-editor.org/rfc/rfc6749
- RFC 7636: Proof Key for Code Exchange by OAuth Public Clients: https://www.rfc-editor.org/rfc/rfc7636
- RFC 9700: Best Current Practice for OAuth 2.0 Security: https://www.rfc-editor.org/rfc/rfc9700
- NIST SP 800-63B-4: Authentication and Authenticator Management: https://pages.nist.gov/800-63-4/sp800-63b.html
- Zanzibar: Google's Consistent, Global Authorization System (USENIX ATC 2019): https://research.google/pubs/zanzibar-googles-consistent-global-authorization-system/
- Microsoft Entra Privileged Identity Management: https://learn.microsoft.com/entra/id-governance/privileged-identity-management/pim-configure
- Microsoft: Manage emergency access accounts: https://learn.microsoft.com/en-us/entra/identity/role-based-access-control/security-emergency-access
- Apache Kafka: Authorization and ACLs: https://kafka.apache.org/44/security/authorization-and-acls/
- RabbitMQ: Authentication, authorisation and access control: https://www.rabbitmq.com/docs/access-control

SECTION 8.4 SUR 8.8 · AVANCÉ · 28 MIN DE LECTURE · DERNIÈRE RELECTURE : NOVEMBRE 2026

Chiffrement, gestion des clés et secrets

AVANT DE LIRE

Cette section s'appuie sur :

- 0.4 Comment les ordinateurs communiquent
- 0.5 Stocker les données
- 8.3 Identité et accès

Termes déjà vus qu'elle utilise : TLS et HTTPS, Identité de charge de travail (workload identity), TLS mutuel (mTLS), Passkey (clé d'accès), JSON Web Token (JWT)

L'IDÉE EN BREF

Pensez à un hôtel. Les objets de valeur des clients vont dans les coffres des chambres, et ces coffres ne valent que par le contrôle exercé sur ceux qui détiennent les codes. L'hôtel ne consigne pas le code de chaque client dans un registre unique à la réception ; le passe-partout se trouve dans une armoire fermée à clé que seul le directeur peut ouvrir, et chaque ouverture est consignée dans un journal. Les objets les plus précieux, l'hôtel ne les garde pas du tout : il remet au client un ticket numéroté, et c'est la banque d'à côté qui conserve l'objet. La protection des données fonctionne de la même façon : chiffrez-les, gardez les clés avec plus de soin que les données, et, lorsque c'est possible, ne détenez pas du tout les données les plus sensibles.

CE QU'IL FAUT SAVOIR

- Chaque couche de chiffrement protège contre des acteurs particuliers : TLS contre le réseau, le chiffrement du stockage contre le vol de supports et de sauvegardes, le chiffrement au niveau des champs contre les administrateurs de bases de données et les copies en aval ; la protection se ramène toujours au contrôle des clés.
- Les algorithmes symétriques (AES-GCM) chiffrent les données en masse ; la cryptographie asymétrique fournit l'accord de clés et les signatures numériques ; les premières normes post-quantiques du NIST (FIPS 203 à 205, 2024) font de l'agilité cryptographique une préoccupation architecturale.
- Le hachage, les MAC, les signatures, le chiffrement et l'encodage sont des outils différents ; les mots de passe exigent un hachage lent et salé, comme Argon2id avec les paramètres actuels de l'OWASP, ou mieux, des passkeys via un fournisseur d'identité.

- Le chiffrement d'enveloppe enveloppe des clés de données propres à chaque enregistrement sous une clé de chiffrement de clés conservée dans un KMS adossé à des HSM, ce qui rend la rotation peu coûteuse, limite le rayon d'impact et permet le crypto-shredding, au prix d'une dépendance critique.
- Les secrets ont leur place dans un coffre-fort, de préférence sous forme d'identifiants dynamiques de courte durée, ou remplacés par l'identité de charge de travail ; un secret divulgué doit d'abord être révoqué, puis nettoyé ensuite.
- La tokenisation par un prestataire de paiement garde les numéros de carte hors de la plateforme et réduit le périmètre PCI DSS v4.0.1 ; un schéma de classification des données détermine où le chiffrement au niveau des champs vaut ce qu'il coûte.

Ce que le chiffrement protège, et ce qu'il ne protège pas

Le chiffrement est souvent traité comme une case à cocher : « les données sont chiffrées en transit et au repos ». Cette formule masque la question qui compte, à savoir **contre qui**. Chaque couche de chiffrement protège les données contre certains acteurs et pas contre d'autres, et dans tous les cas la protection est exactement aussi solide que le contrôle exercé sur ceux qui peuvent utiliser la clé. Le chiffrement ne supprime pas le problème du contrôle d'accès ; il le déplace des données vers les clés, où il est généralement plus facile à gérer, à auditer et à révoquer.

Le chiffrement **en transit** empêche que des données qui traversent un réseau soient lues ou modifiées en chemin. Sur le web, cela signifie TLS, présenté au chapitre 0 ; entre services, cela signifie TLS ou TLS mutuel, abordé avec le zero trust. Le chiffrement **au repos** protège les données stockées, mais « au repos » ne désigne pas une seule chose : le tableau montre les couches courantes sur la plateforme de livraison de repas.

Les couches de chiffrement et les menaces auxquelles chacune répond

Couche	Protège contre	Ne protège pas contre
TLS en transit	L'écoute clandestine et la falsification sur le réseau, y compris à l'intérieur du centre de données.	Un service ou un endpoint compromis, qui voit les données en clair.
Chiffrement du disque ou du volume (par défaut chez les grands fournisseurs de cloud)	Les disques volés ou mis au rebut, le matériel rendu au fournisseur.	Quiconque atteint les données par le système d'exploitation ou la base de données en fonctionnement.
Chiffrement de la base de données ou du service de stockage (par exemple le chiffrement transparent des données)	Les copies de fichiers de données, les instantanés et les sauvegardes réalisés au niveau de la couche de stockage.	Tout compte de base de données qui peut exécuter des requêtes, y compris celui de l'application elle-même et un attaquant qui utilise une injection SQL.
Chiffrement au niveau de l'application (au niveau des champs)	Les administrateurs de bases de données, les comptes aux droits trop larges, les copies analytiques et les sauvegardes divulguées, car ils ne détiennent que du texte chiffré.	L'application elle-même si elle est compromise, puisqu'elle peut utiliser la clé ; la base de données ne peut plus rechercher ni trier sur ce champ.
Chiffrement de bout en bout	Tous les intermédiaires, y compris l'opérateur de la plateforme.	Rien entre les deux, mais la plateforme ne peut alors plus rien faire d'utile des données.

En lisant le tableau de haut en bas, chaque couche protège contre davantage d'acteurs et coûte davantage de fonctionnalités. Une conception saine utilise partout les couches bon marché et transparentes (TLS et le chiffrement du stockage sont presque gratuits aujourd'hui) et les couches coûteuses de façon sélective, pour les données dont l'exposition causerait le plus de tort. Décider de quelles données il s'agit relève de la classification des données, traitée plus loin dans cette section avec la tokenisation et un exemple commenté.

Cryptographie symétrique et asymétrique

Le chiffrement symétrique utilise une seule clé secrète pour chiffrer et déchiffrer. Il est assez rapide pour les données en masse, et l'algorithme standard est AES (FIPS 197, 2001), avec des clés de 128 ou 256 bits. Les conceptions modernes utilisent le **chiffrement authentifié**, qui détecte les falsifications en plus de masquer le contenu : AES en mode Galois/Counter (AES-GCM) ou ChaCha20-Poly1305 (RFC 8439). Les modes authentifiés exigent un nonce unique pour chaque message chiffré sous la même clé ; réutiliser un nonce avec AES-GCM peut révéler le texte en clair et permettre des falsifications, ce qui est l'une des nombreuses raisons d'utiliser des bibliothèques éprouvées plutôt que d'assembler soi-même des primitives.

La cryptographie asymétrique utilise une paire de clés : une clé publique que tout le monde peut détenir et une clé privée gardée secrète. Elle permet deux choses dont la cryptographie symétrique est incapable. L'**accord de clés** permet à deux parties qui ne se sont jamais rencontrées d'établir une clé symétrique partagée sur un réseau ouvert (Diffie-Hellman sur courbes elliptiques dans TLS 1.3). Les **signatures numériques** (RSA, ECDSA, Ed25519) permettent à quiconque possède la clé publique de vérifier que des données proviennent du détenteur de la clé privée et n'ont pas été modifiées, ce qui apporte l'intégrité, l'authenticité et une base pour la non-répudiation. Les opérations asymétriques sont lentes, si bien que les protocoles réels sont hybrides : TLS utilise la cryptographie asymétrique pour authentifier le serveur et convenir des clés, puis le chiffrement symétrique pour les données.

Un grand ordinateur quantique casserait les algorithmes RSA et à courbes elliptiques actuels, mais pas AES avec des clés suffisamment longues. En août 2024, le NIST a publié ses premières normes post-quantiques : FIPS 203 (ML-KEM, pour l'établissement de clés), FIPS 204 (ML-DSA) et FIPS 205 (SLH-DSA), ces deux dernières pour les signatures. Les données qui doivent rester confidentielles pendant de nombreuses années sont menacées dès aujourd'hui par des attaquants qui enregistrent maintenant le trafic chiffré pour le déchiffrer plus tard ; la réponse architecturale est donc l'**agilité cryptographique** : tenez un inventaire des endroits où chaque algorithme est utilisé, évitez de les coder en dur, et préférez des bibliothèques, des protocoles et des fournisseurs capables d'en changer, comme les grands navigateurs et les bibliothèques TLS ont commencé à le faire avec l'échange de clés hybride post-quantique.

Hachage, MAC, signatures et chiffrement sont des outils différents

Confondre ces primitives est une source courante de défauts de conception, comme « chiffrer » des mots de passe pour pouvoir les déchiffrer, ou se fier à un simple hachage pour prouver qu'un webhook provient d'un prestataire de paiement. Le tableau les présente côte à côte.

Choisir la bonne primitive

Outil	Réversible ?	Clé	À utiliser pour	Exemple
Encodage (Base64, hexadécimal)	Oui, par n'importe qui	Aucune	Représenter des octets sous forme de texte. N'apporte aucune sécurité.	La charge utile d'un JWT
Hachage cryptographique (SHA-256)	Non	Aucune	Empreintes, intégrité des fichiers face à une modification accidentelle, adressage par le contenu. MD5 et SHA-1 sont cassés en matière de résistance aux collisions.	Somme de contrôle d'une version téléchargée
Code d'authentification de message (HMAC-SHA-256)	Non	Secret partagé	Prouver qu'un message provient d'une personne qui détient la clé et n'a pas été modifié.	Vérifier le webhook d'un prestataire de paiement
Signature numérique (Ed25519, ECDSA)	Non (elle est vérifiée)	La clé privée signe, la clé publique vérifie	Une preuve d'origine que chacun peut vérifier sans pouvoir la forger.	Jetons d'accès JWT, images de conteneur signées
Chiffrement (AES-GCM)	Oui, avec la clé	Clé secrète (ou paire de clés, pour les schémas hybrides)	Confidentialité, plus intégrité dans les modes authentifiés.	La pièce d'identité d'un livreur, une fois stockée
Hachage de mot de passe (Argon2id)	Non, et volontairement lent	Sel propre à chaque mot de passe, poivre (pepper) facultatif	Stocker les mots de passe de sorte qu'une base de données volée résiste aux tentatives pour les deviner.	Le magasin d'identifiants du fournisseur d'identité

Stocker les mots de passe

Une fonction de hachage généraliste comme SHA-256 est conçue pour être rapide, ce qui est exactement l'inverse de ce qu'il faut pour les mots de passe : les processeurs graphiques peuvent tester des milliards d'essais par seconde contre une table volée de hachages rapides. Le hachage des mots de passe utilise des fonctions volontairement lentes et, idéalement, gourmandes en mémoire (memory-hard), afin que chaque essai coûte à l'attaquant du temps et

de la mémoire bien réels, avec un **sel** aléatoire unique par mot de passe, pour que des mots de passe identiques produisent des hachages différents et que les tables précalculées soient inutiles.

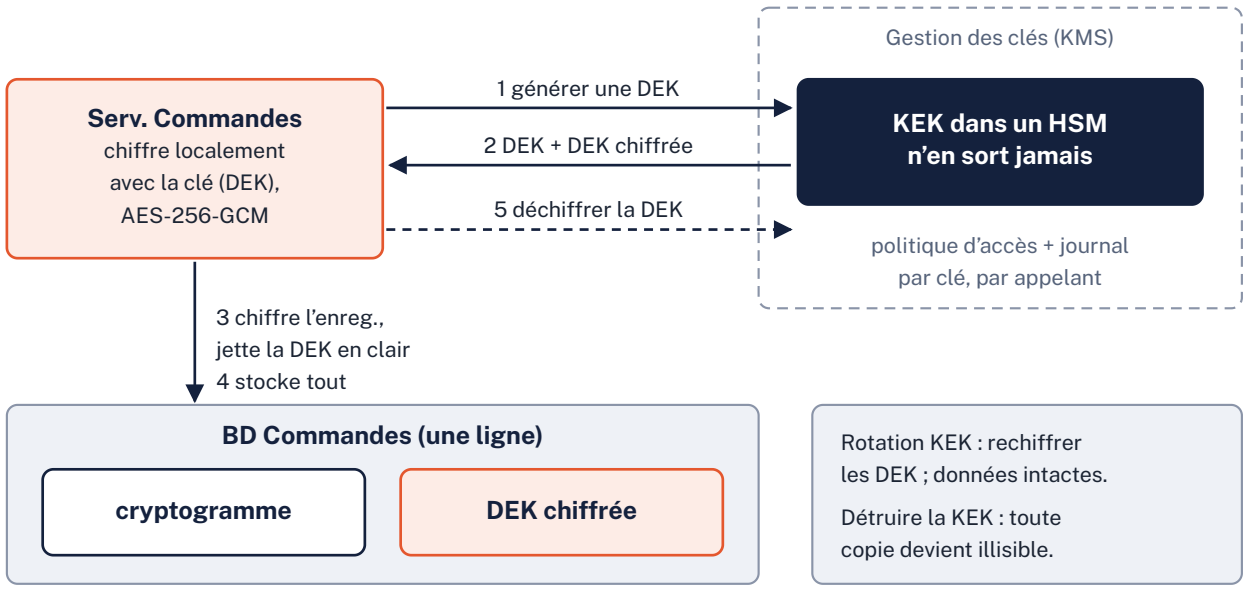
L'OWASP Password Storage Cheat Sheet donne les recommandations actuelles par ordre de préférence. **Argon2id** (une variante d'Argon2, lauréat de la Password Hashing Competition de 2015), avec au minimum 19 Mio de mémoire, deux itérations et un degré de parallélisme de un (ou un compromis équivalent, comme 46 Mio avec une itération). **scrypt** lorsque Argon2id n'est pas disponible, avec un paramètre de coût de 2^{17} (128 Mio), une taille de bloc de 8 et un parallélisme de 1. **bcrypt** uniquement pour les systèmes hérités, avec un facteur de travail d'au moins 10, en gardant à l'esprit que la plupart des implémentations ignorent les entrées au-delà de 72 octets. Lorsque la conformité FIPS 140 est exigée, **PBKDF2** avec HMAC-SHA-256 et 600 000 itérations. Ces valeurs augmentent à mesure que le matériel progresse : consultez donc la version actuelle de la cheat sheet plutôt que de recopier des chiffres dans un document de conception.

Trois points de conception en découlent. Stockez l'algorithme et les paramètres avec chaque hachage (les formats encodés standard le font), pour pouvoir relever les paramètres plus tard : lorsqu'un utilisateur se connecte avec succès, vérifiez le mot de passe par rapport à l'ancien hachage et recalculez immédiatement le hachage avec les nouveaux paramètres. Envisagez un **poivre** (pepper), un secret ajouté à chaque mot de passe avant le hachage et conservé dans un coffre-fort ou un HSM plutôt que dans la base de données, ce qui met en échec un attaquant qui ne vole que la base de données. Surtout, demandez-vous si vous devez vraiment stocker des mots de passe : déléguer l'authentification à un fournisseur d'identité et proposer des passkeys retire le secret le plus attaqué de la plupart de vos services.

Gestion des clés et chiffrement d'enveloppe

Les clés ont un cycle de vie, décrit dans la NIST SP 800-57 Part 1 : génération (à partir d'une source aléatoire appropriée), stockage, distribution, utilisation, rotation à la fin d'une cryptopériode définie, révocation en cas de compromission, et destruction. Faire tout cela de façon sûre à la main est irréaliste, c'est pourquoi les architectures s'appuient sur un service de gestion de clés (KMS) comme AWS KMS, Azure Key Vault ou Google Cloud KMS, ou sur un système auto-hébergé comme HashiCorp Vault. Un KMS conserve les clés maîtresses dans des modules de sécurité matériels (HSM), des dispositifs résistants aux manipulations validés selon FIPS 140, et effectue des opérations avec ces clés à la demande, sous réserve d'une politique d'accès, de sorte que les applications ne voient jamais les clés maîtresses et que chaque utilisation est journalisée.

Envoyer chaque enregistrement au KMS pour le chiffrer serait lent et coûteux, et les API de KMS limitent la taille des requêtes. Le chiffrement d'enveloppe (envelope encryption) résout ce problème. L'application demande au KMS une nouvelle **clé de chiffrement des données** (DEK) ; le KMS la renvoie deux fois, en clair et enveloppée (chiffrée) sous une **clé de chiffrement de clés** (KEK) qui ne quitte jamais le HSM. L'application chiffre les données localement avec la DEK en clair, efface la DEK de sa mémoire et stocke le texte chiffré avec la DEK enveloppée. Pour déchiffrer, elle n'envoie au KMS que la petite DEK enveloppée ; le KMS vérifie la permission de l'appelant, journalise la requête et renvoie la DEK.



Chiffrement d'enveloppe : les données en masse ne vont jamais au KMS, et la clé maîtresse n'en sort jamais.

```
import os
from cryptography.hazmat.primitives.ciphers.aead import AESGCM

def encrypt_field(plaintext: bytes, kms, kek_id: str, context: bytes) → dict:
    dek = AESGCM.generate_key(bit_length=256)           # fresh data key
    nonce = os.urandom(12)                              # never reuse with this key
    ct = AESGCM(dek).encrypt(nonce, plaintext, context)  # context binds it to its row
    wrapped = kms.encrypt(key_id=kek_id, plaintext=dek)  # KEK stays inside the KMS
    return {"ct": ct, "nonce": nonce, "wrapped_dek": wrapped, "kek": kek_id}

def decrypt_field(rec: dict, kms, context: bytes) → bytes:
    dek = kms.decrypt(key_id=rec["kek"], ciphertext=rec["wrapped_dek"])  # checked,
logged
    return AESGCM(dek).decrypt(rec["nonce"], rec["ct"], context)
```

Chiffrement d'enveloppe d'un champ avec la bibliothèque Python cryptography ; kms désigne n'importe quel client KMS (les vraies API de KMS combinent les deux premières étapes en un seul appel generate-data-key).

Le chiffrement d'enveloppe présente plusieurs avantages architecturaux. **La rotation est peu coûteuse** : la rotation d'une KEK crée une nouvelle version de clé pour les nouvelles opérations d'enveloppement, tandis que les anciennes versions restent disponibles pour le déchiffrement ; les DEK existantes peuvent être réenveloppées sans toucher aux données, et un rechiffrement complet n'est nécessaire que si une DEK elle-même a pu être exposée. **Le rayon d'impact est limité** : avec une DEK par enregistrement, par fichier ou par partition, une DEK divulguée expose peu de choses. **La séparation des tâches** devient possible : le service des commandes peut utiliser sa KEK pour désenvelopper, mais ni modifier sa politique ni la supprimer, tandis que les administrateurs de la sécurité peuvent gérer les clés mais pas déchiffrer les données. **Les hiérarchies de clés** épousent l'organisation : une KEK par service et par classe de données, ou par locataire, si bien qu'une grande chaîne de restaurants pourrait même fournir et contrôler sa propre clé. Enfin, détruire une clé est un outil puissant : le crypto-shredding (effacement cryptographique) rend illisible toute copie des données chiffrées sous cette clé, y compris les sauvegardes et les journaux d'événements immuables, ce qui compte pour l'effacement au titre du RGPD et pour les systèmes en event sourcing.

Le prix à payer est une nouvelle dépendance critique. Si le KMS est indisponible, les données chiffrées ne peuvent pas être lues ; si une clé est supprimée par erreur, les données sont perdues pour de bon. Les KMS des fournisseurs de cloud imposent donc un délai d'attente avant suppression, et les architectes devraient traiter la suppression de clés et les modifications de politique de clé comme des opérations privilégiées, journalisées et à deux personnes, mettre brièvement en cache les DEK désenveloppées là où la latence des appels au KMS compte, et inclure le KMS dans les plans de reprise après sinistre.

SI VOUS DÉBUTEZ

Pas à pas : une clé qui verrouille d'autres clés

L'idée qui demande le plus d'attention ici est le chiffrement d'enveloppe : les données sont verrouillées avec leur propre clé de données, et cette clé est elle-même verrouillée avec une clé maîtresse qui ne quitte jamais le service de gestion de clés (KMS). Suivez les coordonnées bancaires de versement d'un restaurant à leur entrée dans le stockage et à leur sortie.

- Pour stocker les nouvelles coordonnées bancaires d'un restaurant, le service des paiements demande au KMS une nouvelle clé de données. Le KMS la renvoie deux fois : en clair, et enveloppée, c'est-à-dire verrouillée sous la clé maîtresse des paiements, qui reste dans le matériel inviolable du KMS.
- Le service verrouille les coordonnées avec la clé de données en clair sur sa propre machine, efface cette clé de sa mémoire, et stocke les coordonnées verrouillées avec la clé enveloppée. Les coordonnées ne transitent jamais par le KMS.
- Quiconque copie cette ligne détient des coordonnées verrouillées et une clé verrouillée, inutiles sans la permission d'utiliser la clé maîtresse.
- Quand la tâche de versement a besoin des coordonnées, elle n'envoie au KMS que la petite clé enveloppée ; le KMS vérifie que cette tâche a le droit d'utiliser la clé maîtresse, journalise la requête et renvoie la clé de données en clair.
- Deux avantages en découlent. La rotation de la clé maîtresse n'a pas besoin de toucher aux coordonnées stockées : il suffit de réenvelopper les petites clés de données. Et détruire une clé rend illisible toute copie des données chiffrées sous cette clé, sauvegardes comprises : c'est le crypto-shredding.
- Le prix à payer est une nouvelle dépendance critique. Si le KMS est indisponible, les coordonnées ne peuvent pas être lues ; si une clé est supprimée par erreur, les données sont perdues pour de bon.
- **La leçon :** gardez les données en masse en local et les clés au centre. Chaque enregistrement peut avoir sa propre clé de données, tandis que la clé maîtresse reste dans le KMS, où chaque utilisation est vérifiée et journalisée.

Gestion des secrets

Un secret est tout identifiant qui donne un accès : mots de passe de bases de données, clés d'API du prestataire de paiement, clés privées, clés de signature, secrets clients OAuth, secrets de webhook. La gestion des secrets part d'une règle : **les secrets ne vivent jamais dans le code source**, ni dans les images de conteneur, ni dans les fichiers de configuration commités dans

des dépôts, car ceux-ci sont copiés, mis en cache, forkés et lus par beaucoup plus de personnes et de systèmes que ne devrait l'être le secret. Une fois dans l'historique d'un dépôt, un secret y reste même après la suppression de la ligne.

Les secrets ont leur place dans un coffre-fort de secrets (HashiCorp Vault, AWS Secrets Manager, Azure Key Vault, Google Secret Manager) qui les stocke chiffrés, ne les remet qu'aux charges de travail authentifiées conformément à une politique, journalise chaque accès et prend en charge la rotation. Mieux encore que de renouveler des secrets de longue durée : ne pas en avoir. Un coffre-fort peut émettre des **secrets dynamiques**, comme un compte de base de données créé pour une instance de service avec un bail d'une heure, et l'identité de charge de travail permet aux services de s'authentifier auprès des API cloud et entre eux sans aucun secret stocké. Les secrets injectés à l'exécution devraient être lus dans des fichiers ou par l'API du coffre-fort plutôt que dans des variables d'environnement largement visibles, qui ont tendance à finir dans des vidages mémoire après plantage et dans des journaux de débogage.

La prévention exige de l'automatisation, car les gens font des erreurs. Les hooks de pré-commit et les scanners de pipeline (comme gitleaks et detect-secrets) interceptent les secrets avant qu'ils ne soient poussés ; les hébergeurs de dépôts recherchent les formats de secrets connus, et la push protection de GitHub bloque les push qui en contiennent. Les dépôts publics sont scannés par des attaquants dans les minutes qui suivent un push. Les conséquences sont bien documentées : en 2016, des attaquants qui avaient obtenu l'accès à un dépôt GitHub privé utilisé par des ingénieurs d'Uber y ont trouvé une clé d'accès au stockage cloud d'Uber et ont téléchargé les données à caractère personnel d'environ 57 millions de passagers et de chauffeurs.

Liste de contrôle : quand un secret a fuité

Considérez comme compromis tout secret qui a atteint un dépôt, un journal, un ticket ou une messagerie instantanée, que le dépôt soit privé ou non.

- [ ] **Révoquez ou renouvelez le secret immédiatement**
  Faites-le en premier. Supprimer le commit ou le message ne sert à rien : les clones, les forks, les caches et les scanners l'ont déjà.
- [ ] **Établissez ce à quoi il donnait accès**
  Listez les systèmes, les données et les permissions que le secret accordait, ainsi que tout autre secret accessible à partir de là.
- [ ] **Examinez l'utilisation depuis l'exposition**
  Consultez les journaux d'accès du fournisseur et du coffre-fort à la recherche d'utilisations par des identités, depuis des adresses ou à des heures inattendues, depuis la première apparition du secret.
- [ ] **Contenez et nettoyez**
  Renouvelez les identifiants qui en dépendent, invalidez les sessions et les jetons émis grâce à lui, puis retirez le secret de l'historique et des caches (après la rotation, pas à la place de celle-ci).
- [ ] **Décidez de la notification**
  Si des données à caractère personnel ont pu être consultées, lancez l'évaluation de la violation ; au titre du RGPD, le délai de notification à l'autorité de contrôle court à partir du moment où vous en avez pris connaissance.
- [ ] **Corrigez la cause**
  Placez le secret dans un coffre-fort, remplacez-le par une identité de charge de travail ou un secret dynamique, et activez l'analyse des secrets et la push protection pour que la même erreur soit bloquée la prochaine fois.

La tokenisation et la réduction du périmètre PCI DSS

La tokenisation remplace une valeur sensible par un jeton qui n'a aucun lien mathématique avec elle ; la valeur réelle est conservée dans un coffre-fort séparé et étroitement protégé, et seul ce coffre-fort peut faire la correspondance inverse. Contrairement au chiffrement, un jeton volé ne peut pas être inversé avec une clé, car il n'y a pas de clé, seulement une consultation dans un système que l'attaquant ne contrôle pas.

Son usage le plus important concerne les paiements par carte. La Payment Card Industry Data Security Standard (PCI DSS, actuellement en version 4.0.1, publiée en juin 2024) s'applique à tout composant de système qui stocke, traite ou transmet des données de titulaires de carte, ou qui est connecté à ces composants ou pourrait en affecter la sécurité. Une plateforme qui laisse les numéros de carte transiter par ses serveurs fait entrer dans le périmètre son API gateway, ses services, ses journaux, ses réseaux et ses administrateurs, avec des centaines d'exigences et une évaluation annuelle. Une plateforme qui utilise les champs de paiement hébergés ou la redirection du prestataire de paiement, de sorte que les numéros de carte vont directement de l'appareil du client au prestataire, ne reçoit qu'un jeton et éventuellement les quatre derniers chiffres, et son périmètre PCI DSS se réduit considérablement : un petit commerçant peut avoir droit au questionnaire d'auto-évaluation le plus court, et même les plus grands commerçants, qui ont toujours besoin d'une évaluation annuelle par un évaluateur qualifié, ont beaucoup moins de systèmes à évaluer. C'est la conception retenue dans le modèle de menaces de l'étape de paiement : les données de carte les moins chères à protéger sont celles que vous ne détenez jamais. Chiffrer vous-même les numéros de carte ne permet pas la même réduction, car quiconque peut les déchiffrer reste dans le périmètre.

La même idée fonctionne pour d'autres identifiants sensibles. Un service interne de tokenisation peut remplacer les numéros de sécurité sociale ou de passeport par des jetons dans toute la plateforme, de sorte qu'un seul petit service, fortement protégé, voie jamais les valeurs réelles.

Classification des données

Les décisions de chiffrement, de tokenisation, de journalisation, de conservation et d'accès dépendent toutes de la connaissance du degré de sensibilité de chaque donnée. Un schéma de **classification des données** définit quelques niveaux assortis de règles de traitement, pour que les équipes ne tranchent pas chaque cas à partir de zéro. Quatre niveaux sont courants ; l'important est que chaque jeu de données du catalogue de données ou du schéma de la plateforme porte un niveau, et que les règles soient appliquées par des modèles et des contrôles plutôt que de mémoire.

Un schéma de classification des données pour la plateforme de livraison de repas

Niveau	Exemples	Règles de traitement
Public	Cartes, prix, horaires d'ouverture des restaurants	Protection de l'intégrité uniquement ; mise en cache possible n'importe où.
Interne	Chiffres de ventes agrégés, configuration des services sans secrets	Personnel et services uniquement ; jamais dans des journaux publics.
Confidentiel	Noms, adresses, numéros de téléphone et historiques de commandes des clients, positions des livreurs	TLS et chiffrement du stockage ; accès selon le moindre privilège ; masquage dans les outils de support ; minimisation dans l'analytique ; durées de conservation limitées.
Restreint	Coordonnées bancaires de versement, pièces d'identité des livreurs, secrets, jetons de paiement	Tout ce qui précède, plus un chiffrement au niveau des champs avec une clé dédiée, un accès réservé à des services nommément désignés, et la journalisation de chaque accès.

Chiffrement au niveau des champs

Le chiffrement au niveau des champs, la couche applicative du premier tableau, chiffre des champs individuels avant leur écriture, généralement avec un chiffrement d'enveloppe et une KEK par classe de données. Sa force est que tout ce qui se trouve en aval de l'application ne détient que du texte chiffré : les administrateurs de bases de données, les répliques en lecture, les sauvegardes et l'entrepôt de données alimenté par capture des données modifiées. Une violation de l'un d'eux n'expose rien de lisible, et l'accès au texte en clair se réduit aux services qui peuvent appeler le KMS avec cette clé, chaque appel étant journalisé.

Ses coûts sont réels. La base de données ne peut plus indexer, rechercher, trier ni faire de jointure sur le champ chiffré. Les recherches d'égalité peuvent être rétablies avec un **index aveugle** (blind index), une colonne séparée qui contient un hachage à clé (HMAC) de la valeur normalisée, au prix de révéler quelles lignes partagent une même valeur. Le chiffrement déterministe présente la même fuite, tandis que les requêtes par plage nécessitent des schémas spécialisés ou ne devraient tout simplement pas être faites sur de tels champs. Certaines bases de données proposent un chiffrement côté client au niveau des champs ou un chiffrement interrogeable, qui intègrent ces techniques. En raison de ces coûts, le chiffrement au niveau des champs devrait être réservé aux données restreintes, dont les champs sont de toute façon rarement interrogés.

EXEMPLE COMMENTÉ

Exemple commenté : quoi chiffrer, où, et avec quelles clés

L'architecte de la plateforme de livraison de repas passe en revue les principaux jeux de données au regard du schéma de classification. Chaque service utilise déjà TLS (TLS mutuel en interne) et un stockage chiffré avec une clé KMS par service : la question est donc de savoir ce qui a besoin de plus.

Numéros de carte	Jamais détenus. Les champs de paiement hébergés les envoient au prestataire de paiement ; Paiements stocke le jeton du prestataire, la marque de la carte et les quatre derniers chiffres. Le périmètre PCI DSS se réduit à l'intégration de l'étape de paiement avec le prestataire.
Mots de passe des clients	Non stockés par les services produit. Le fournisseur d'identité propose des passkeys ; les mots de passe restants sont hachés avec Argon2id aux paramètres de l'OWASP, avec un poivre conservé dans un HSM.
Adresses et numéros de téléphone (niveau confidentiel)	Chiffrement du stockage avec la KEK de Commandes ; comptes selon le moindre privilège ; masquage dans l'outil de support ; l'entrepôt ne reçoit que les secteurs de code postal, pas les adresses complètes.
Coordonnées bancaires de versement des restaurants (niveau restreint)	Chiffrement d'enveloppe au niveau des champs sous une KEK dédiée de Paiements ; seule la tâche de versement peut désenvelopper ; toute modification exige une réauthentification et prévient le propriétaire du restaurant, ce qui protège l'intégrité autant que la confidentialité.
Pièces d'identité des livreurs (niveau restreint)	Stockées sous forme d'objets, chacun avec sa propre DEK enveloppée sous une KEK d'intégration des livreurs ; seul le service de vérification peut déchiffrer ; chaque accès est journalisé et revu chaque mois ; suppression à l'issue de la durée de conservation légale.
Données à caractère personnel dans les événements et le journal des événements	Les événements transportent des identifiants de clients plutôt que des adresses ; les quelques champs personnels qui doivent circuler sont chiffrés avec une clé par client, de sorte que détruire la clé les efface du journal conservé.
Sauvegardes	Chiffrées avec une KEK distincte conservée dans un compte cloud distinct, de sorte que la compromission des identifiants de production n'expose pas les sauvegardes ; les restaurations sont testées chaque trimestre.
Secrets des services	Dans le coffre-fort ; les bases de données utilisent des identifiants dynamiques avec des baux d'une heure ; la clé d'API du prestataire de paiement n'est lisible que par l'identité de charge de travail de Paiements.

Une dizaine de KEK, chacune appartenant à un service et renouvelée automatiquement chaque année, couvrent toute la plateforme ; le levier réside dans les décisions, pas dans le nombre de clés.

Deux décisions suppriment des données au lieu de les protéger (les numéros de carte, et les adresses complètes dans l'analytique), ce qui est toujours l'option la plus solide.

La conception est consignée dans un ADR par jeu de données restreint et, dans la description d'architecture, sous forme de concept de protection des données.

VÉRIFIEZ VOTRE COMPRÉHENSION

▼ Contre quels acteurs le chiffrement transparent de la base de données ne protège-t-il pas ?

Contre quiconque peut interroger la base de données au moyen d'un compte valide, y compris l'application, un attaquant qui utilise une injection SQL et les administrateurs de bases de données, car la base de données déchiffre les données pour eux.

▼ Pourquoi la rotation d'une clé de chiffrement de clés est-elle peu coûteuse avec le chiffrement d'enveloppe ?

Seules les petites clés de données enveloppées doivent être réenveloppées sous la nouvelle version de la clé ; les données en masse restent chiffrées sous leurs clés de données inchangées.

▼ Un développeur propose de stocker les mots de passe chiffrés avec AES pour que le personnel du support puisse indiquer aux utilisateurs leur mot de passe. Qu'est-ce qui ne va pas ?

Les mots de passe doivent être stockés avec un hachage de mot de passe à sens unique, lent et salé, pour que personne, ni le personnel ni quiconque vole la clé, ne puisse les retrouver ; la récupération doit réinitialiser, pas révéler.

▼ Pourquoi utiliser les champs hébergés d'un prestataire de paiement réduit-il davantage le périmètre PCI DSS que de chiffrer soi-même les numéros de carte ?

Avec des champs hébergés, les numéros de carte n'atteignent jamais les systèmes de la plateforme, qui sont donc hors du périmètre ; des numéros de carte chiffrés que la plateforme peut déchiffrer maintiennent ses systèmes dans le périmètre.

▼ À vous : La plateforme utilise déjà TLS et le stockage chiffré. Décidez ce qu'il faut faire de plus pour les numéros de carte des clients, leurs mots de passe et leurs notes sur les allergies. Pour chacun, donnez la protection et la menace à laquelle elle répond. Supposez que la classification considère les notes sur les allergies comme sensibles.

Numéros de carte : ne les détenez jamais ; utilisez les champs de paiement hébergés et le jeton du prestataire, ce qui réduit le périmètre PCI DSS. Mots de passe : pas de chiffrement, mais un hachage lent et salé tel qu'Argon2id, ou des passkeys via un fournisseur d'identité, de sorte qu'une table volée soit peu utile. Notes sur les allergies : chiffrement au niveau des champs avec des clés d'enveloppe dans un KMS, ce qui protège contre les administrateurs et les copies en aval. Cela coûte la possibilité de recherche et ajoute une dépendance au KMS : réservez-le donc aux champs sensibles.

SOURCES

- OWASP Password Storage Cheat Sheet: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html
- OWASP Secrets Management Cheat Sheet: https://cheatsheetseries.owasp.org/cheatsheets/Secrets_Management_Cheat_Sheet.html
- NIST SP 800-57 Part 1 Rev. 5: Recommendation for Key Management: https://csrc.nist.gov/pubs/sp/800/57/pt1/r5/final
- Google Cloud KMS: Envelope encryption: https://cloud.google.com/kms/docs/envelope-encryption
- PCI Security Standards Council: https://www.pcisecuritystandards.org/

SECTION 8.5 SUR 8.8 · AVANCÉ · 34 MIN DE LECTURE · DERNIÈRE RELECTURE : NOVEMBRE 2026

Communication sécurisée et zero trust

AVANT DE LIRE

Cette section s'appuie sur :

- 0.4 Comment les ordinateurs communiquent
- 8.3 Identité et accès
- 8.4 Protéger les données

Termes déjà vus qu'elle utilise : TLS et HTTPS, OAuth 2.0, Identité de charge de travail (workload identity), Illusions de l'informatique distribuée, API gateway (passerelle d'API)

L'IDÉE EN BREF

Imaginez un hôtel où la réception vérifie votre identité une seule fois, puis vous remet une clé qui ouvre toutes les chambres du bâtiment. Si un voleur passe la réception, toutes les chambres lui sont ouvertes. Un meilleur hôtel vous donne une carte-clé qui n'ouvre que votre chambre, uniquement pour les nuits réservées, et qui enregistre chaque porte qu'elle ouvre. Le zero trust applique la seconde idée au logiciel : aucune requête n'est considérée comme fiable parce qu'elle vient de l'intérieur du réseau de l'entreprise. Chaque personne et chaque programme doit prouver qui il est et être autorisé à faire cette action précise, à chaque fois.

CE QU'IL FAUT SAVOIR

- Le modèle périmétrique fait confiance à tout ce qui se trouve dans le réseau : un seul identifiant volé ou une seule machine compromise permet donc un mouvement latéral ; le cloud, le mobile et le travail à distance ont en outre supprimé tout périmètre unique à défendre.
- NIST SP 800-207 (août 2020) définit le zero trust comme le fait de n'accorder aucune confiance implicite fondée sur l'emplacement réseau ou la propriété, avec sept principes, dont la sécurisation de toutes les communications, l'accès par session et une politique dynamique appliquée strictement.
- Un point d'application de politique placé devant chaque ressource interroge un point de décision de politique (moteur de politique et administrateur de politique), qui décide en fonction de l'identité, de la posture de l'appareil ou de la charge de travail, du renseignement sur les menaces et de la politique.

- BeyondCorp, chez Google (2014), a appliqué le zero trust à l'accès du personnel avec un inventaire des appareils, un proxy d'accès et des niveaux de confiance, et a supprimé le réseau de bureau privilégié et le VPN.
- Le TLS mutuel apporte à chaque appel de service le chiffrement et une identité vérifiée aux deux extrémités ; SPIFFE normalise des identités de charge de travail à courte durée de vie, et les service meshes automatisent le mTLS et la politique par appel, pour un coût d'exploitation réel.
- Les certificats doivent être émis et renouvelés automatiquement (ACME, RFC 8555), d'autant que la durée de vie des certificats TLS publics descend à 200 jours en 2026, 100 jours en 2027 et 47 jours en 2029.
- Les API gateways authentifient les jetons, limitent le débit et valident les entrées, mais l'autorisation au niveau des objets doit toujours avoir lieu dans le service qui possède les données.

Pourquoi le périmètre réseau a cessé de fonctionner

Pendant des décennies, la plupart des organisations ont protégé leurs systèmes comme un château entouré de douves. Un pare-feu séparait l'intérieur de confiance de l'internet non fiable, le personnel se connectait à l'intérieur par un réseau privé virtuel (VPN), et tout ce qui se trouvait déjà à l'intérieur était considéré comme amical. Les services du centre de données s'appelaient les uns les autres en HTTP simple, sans authentification, car le réseau lui-même tenait lieu d'identifiant. Le chapitre 6 rangeait «le réseau est sécurisé» parmi les illusions de l'informatique distribuée ; le modèle périmétrique est cette illusion transformée en architecture.

Trois évolutions l'ont mis en échec. D'abord, l'intérieur est devenu trop vaste pour être défendu : les ordinateurs portables voyagent, le personnel travaille depuis chez lui, des partenaires se connectent, et un seul e-mail d'hameçonnage (phishing) ou un seul mot de passe VPN volé place un attaquant à l'intérieur des douves. Une fois là, l'attaquant se déplace de machine en machine, ce qu'on appelle le mouvement latéral. Lors de la violation de 2013 chez le distributeur américain Target, les attaquants ont d'abord utilisé des identifiants réseau volés à un sous-traitant en chauffage et ventilation, puis ont atteint de là les terminaux de paiement, emportant les données d'environ 40 millions de cartes. Ensuite, l'intérieur s'est dissous : les charges de travail tournent désormais sur plusieurs comptes cloud, des produits SaaS et des API tierces, si bien qu'il n'existe plus de mur unique sur lequel poser un pare-feu. Enfin, l'intérieur est devenu dynamique : des conteneurs apparaissent et disparaissent à chaque minute et les adresses IP sont réutilisées, si bien que des règles écrites sous la forme *autoriser 10.0.4.0/24 à joindre 10.0.7.12* ne décrivent rien de stable.

L'idée qu'il faut abandonner le périmètre est plus ancienne que son nom. Le Jericho Forum, un groupe de responsables de la sécurité d'entreprises, plaidait pour la *dépérimétrisation* dès 2004 ; John Kindervag, alors analyste chez Forrester Research, a forgé l'expression *zero trust* vers 2010 ; et les articles de Google sur BeyondCorp, à partir de 2014, ont montré que l'approche fonctionnait à grande échelle. Le slogan est *ne jamais faire confiance, toujours vérifier* : aucune requête n'est considérée comme fiable en raison de sa provenance.

> **SI VOUS DÉBUTEZ**
>
> ### Si vous débutez
>
> Pensez à un immeuble de bureaux. L'ancienne approche contrôle votre badge à l'entrée, puis vous laisse accéder à toutes les pièces. Le zero trust contrôle votre badge, et vérifie si vous devez être là à ce moment précis, à chaque porte que vous ouvrez. L'immeuble n'a plus d'intérieur unique de confiance : un voleur qui se faufile derrière quelqu'un à l'entrée ne peut toujours pas entrer dans le bureau de la comptabilité.
>
> La suite de cette section emploie des termes vus dans des sections précédentes : TLS et certificats (section 0.4), jetons, OAuth 2.0 et identité de charge de travail (section 8.3) et chiffrement et clés (section 8.4).

Le zero trust selon la définition du NIST

Le texte de référence est la publication spéciale 800-207 du NIST, *Zero Trust Architecture*, parue en août 2020 (Rose, Borchert, Mitchell et Connelly). Elle décrit le zero trust (confiance zéro) comme un ensemble de principes qui déplacent les défenses de périmètres statiques, fondés sur le réseau, vers les utilisateurs, les actifs et les ressources, et une architecture zero trust comme le plan d'une organisation pour les appliquer. Son affirmation centrale est qu'aucune confiance implicite n'est accordée à un actif ou à un compte utilisateur sur la seule base de son emplacement physique ou réseau, ni de son propriétaire. L'authentification et l'autorisation sont des fonctions distinctes, exécutées avant l'établissement d'une session vers une ressource.

La publication énonce sept principes. Ils méritent d'être connus dans leur formulation propre, car de nombreux produits se disent zero trust alors qu'ils n'en respectent qu'un ou deux.

Les sept principes de NIST SP 800-207, appliqués à la plateforme de livraison de repas

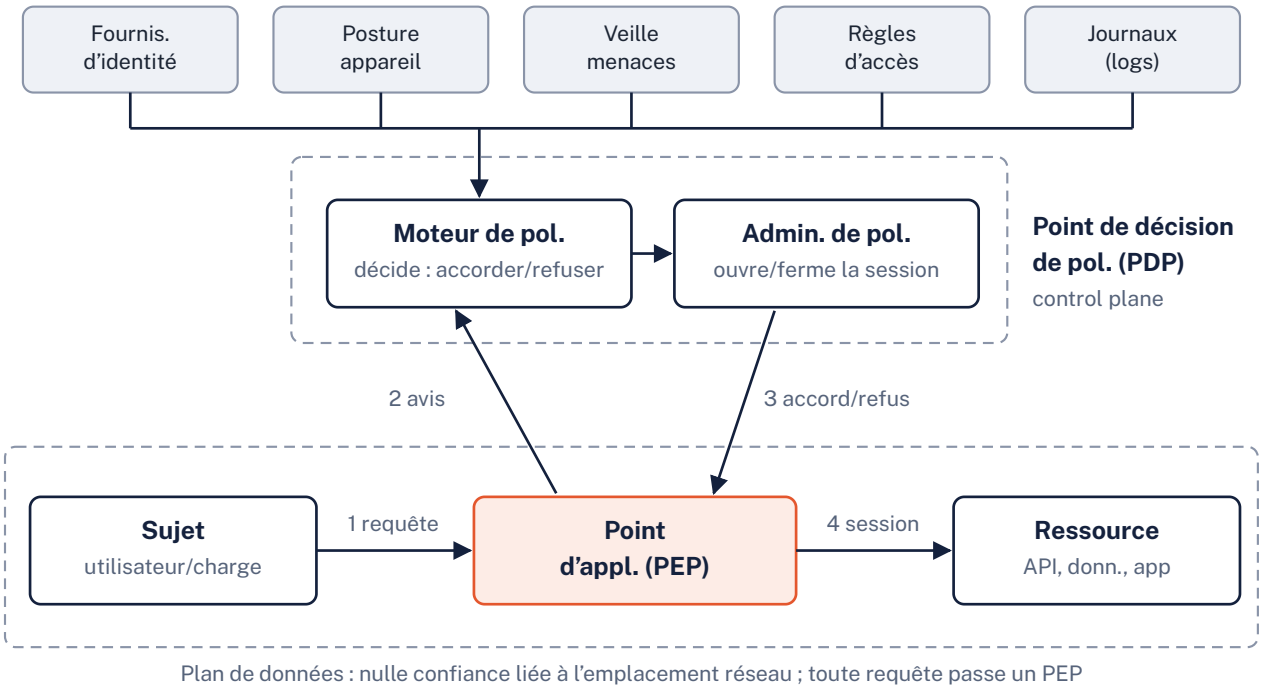
Principe (reformulé)	Ce que cela signifie	Exemple de livraison de repas
1. Toutes les sources de données et tous les services informatiques sont des ressources	Un ordinateur portable, un conteneur, un outil SaaS et une base de données sont tous à protéger, pas seulement les serveurs du centre de données.	Le backend de l'app du livreur, l'API des tablettes des restaurants et le bucket d'analytique sont chacun des ressources dotées de leur propre politique.
2. Toutes les communications sont sécurisées, quel que soit l'emplacement réseau	Le trafic à l'intérieur du centre de données bénéficie du même chiffrement et de la même authentification que le trafic qui passe par internet.	Commandes appelle Paiements en TLS mutuel, alors même que les deux tournent dans le même cluster.
3. L'accès est accordé par session	La confiance est évaluée pour chaque session avec le moindre privilège nécessaire, et l'accès à une ressource ne donne accès à rien d'autre.	La session d'un agent du support peut lire l'historique des commandes d'un client ; elle ne peut pas atteindre en plus l'API des remboursements.
4. L'accès est décidé par une politique dynamique	La décision s'appuie sur l'identité, l'état de l'appareil ou de la charge de travail et d'autres attributs comme l'heure, le lieu et le comportement.	Un remboursement au-delà d'un certain montant depuis un ordinateur portable non géré est refusé ; la même demande depuis un appareil géré, avec une MFA récente, est acceptée.
5. L'entreprise surveille l'intégrité et la posture de sécurité de tous les actifs	Aucun appareil ni aucune charge de travail n'est fiable par défaut ; son niveau de correctifs, sa configuration et son intégrité sont mesurés en continu.	Une image de conteneur sans signature valide, ou présentant une CVE critique non corrigée, n'est pas admise dans le cluster.
6. L'authentification et l'autorisation sont dynamiques et strictement appliquées avant tout accès	L'identité est vérifiée avant chaque accès et revérifiée au cours des longues sessions ; les identifiants ont une courte durée de vie et font l'objet d'une rotation.	Les certificats des services vivent quelques heures et sont renouvelés automatiquement ; les sessions des utilisateurs sont réévaluées quand le risque change.
7. L'entreprise recueille autant d'informations que possible sur l'état des actifs, du réseau et des communications	La télémétrie alimente en retour la politique, de sorte que les décisions s'améliorent avec le temps.	Les journaux d'accès de la passerelle, du service mesh et du fournisseur d'identité alimentent la plateforme de supervision de la sécurité.

Points de décision et d'application de politique

SP 800-207 décrit ce mécanisme à l'aide d'un petit ensemble de composants logiques. Un point d'application de politique (PEP) se place devant une ressource et ne laisse passer le trafic que sur instruction : en pratique, un proxy tenant compte de l'identité (identity-aware proxy), une API gateway, un sidecar de service mesh ou un proxy de base de données. Le point de décision de politique (PDP) prend la décision. Le NIST le divise en un moteur de politique, qui évalue la requête au regard de la politique et des données d'entrée et décide, et un administrateur de politique, qui exécute la décision en ouvrant ou en fermant le chemin, par exemple en émettant un jeton de session à courte durée de vie ou en configurant le PEP. Les données d'entrée comprennent le fournisseur d'identité, la posture de l'appareil et de la charge de travail (niveau de correctifs, configuration, intégrité), le renseignement sur les menaces, les politiques d'accès elles-mêmes et les journaux d'activité.

Séparer la décision de l'application est le même geste que séparer la politique du mécanisme dans les systèmes d'exploitation, et il est payant pour la même raison : les politiques peuvent être écrites une seule fois, revues, testées et modifiées de manière centralisée, tandis que l'application est répartie partout où circule le trafic. Les moteurs de politique de la section 8.3, comme Open Policy Agent, sont souvent le moteur qui se trouve derrière un PDP. La question de conception pour un architecte est de savoir où placer les points d'application pour qu'aucun chemin ne les contourne, et comment garder la décision rapide et disponible, car un PDP sur le chemin critique de chaque requête devient une dépendance avec son propre budget de disponibilité et de latence. Les réponses courantes consistent à mettre les décisions en cache pendant quelques secondes, à pousser des politiques compilées vers le PEP pour qu'il puisse décider localement, et à décider à l'avance de ce qui se passe si le PDP est injoignable (normalement refuser, avec une courte liste d'autorisations pour les contrôles de santé).

Les composants logiques d'une architecture zero trust



Les requêtes ne passent que par un point d'application ; la décision est prise dans un plan de contrôle distinct, alimenté par des données d'identité, de posture, de menaces et de politique.

BeyondCorp : le zero trust pour les personnes

Google a commencé à abandonner son réseau d'entreprise privilégié après l'intrusion de 2009 connue sous le nom d'Opération Aurora, et en a décrit le résultat dans *BeyondCorp: A New Approach to Enterprise Security* (Rory Ward et Betsy Beyer, ;login:, décembre 2014), suivi de plusieurs autres articles. La conception comporte trois parties principales. Chaque appareil est enregistré dans un inventaire des appareils et identifié par un certificat. Un proxy d'accès, joignable depuis internet, se place devant chaque application interne ; les employés atteignent les applications à travers lui depuis n'importe quel réseau, y compris le réseau du bureau, qui ne bénéficie d'aucun privilège particulier. Un composant d'inférence de confiance combine l'identité de l'utilisateur et l'état de l'appareil pour attribuer un niveau de confiance, et chaque application déclare le niveau minimal qu'elle exige.

La conséquence qui a surpris a été la suppression du VPN : un employé dans un café et un employé à son bureau passent par les mêmes contrôles. L'idée est aujourd'hui vendue sous forme de produits appelés identity-aware proxy (proxy tenant compte de l'identité) et accès réseau zero trust (ZTNA). Le Zero Trust Maturity Model de l'agence américaine de cybersécurité et de sécurité des infrastructures (CISA) (version 2.0, avril 2023) est un outil

d'autoévaluation utile : il note cinq piliers (identité, appareils, réseaux, applications et charges de travail, et données) de « traditionnel » à « optimal », avec la visibilité, l'automatisation et la gouvernance en transversal.

TLS mutuel, certificats et identité de charge de travail

Le zero trust entre services exige deux choses à chaque appel : le chiffrement, et une identité forte pour les deux extrémités. Transport Layer Security assure les deux. TLS 1.3 (RFC 8446, août 2018) achève sa poignée de main en un seul aller-retour : le client envoie les algorithmes qu'il prend en charge et une part de clé, le serveur répond avec sa part de clé et son certificat, et tous deux dérivent des clés de session par Diffie-Hellman éphémère, de sorte que chaque session bénéficie de la confidentialité persistante (forward secrecy : une clé de serveur volée ne permet pas de déchiffrer un trafic passé enregistré). TLS 1.3 a supprimé les options faibles des versions antérieures, dont l'échange de clés RSA, les chiffrements en mode CBC, RC4, les signatures SHA-1, la compression et la renégociation ; TLS 1.0 et 1.1 ont été formellement dépréciés par la RFC 8996 en 2021. Un mode facultatif sans aller-retour (0-RTT) permet à un client qui revient d'envoyer des données immédiatement, mais ces données précoces peuvent être rejouées par un attaquant : il faut donc les limiter aux requêtes idempotentes. Les principaux navigateurs et fournisseurs cloud ont aussi commencé à utiliser un échange de clés hybride post-quantique qui combine X25519 et ML-KEM (normalisé par le NIST sous le nom FIPS 203 en 2024), ce qui protège le trafic enregistré aujourd'hui contre un futur ordinateur quantique.

Un certificat lie une clé publique à un nom et est signé par une autorité de certification (CA) ; le client vérifie la chaîne de signatures jusqu'à une racine en laquelle il a déjà confiance. Ce système de CA, de certificats, de magasins de confiance et de révocation est une infrastructure à clés publiques (PKI). Les certificats publics sont aussi inscrits dans des journaux Certificate Transparency, si bien qu'un certificat émis à tort pour votre domaine peut être repéré. Le HTTPS ordinaire n'authentifie que le serveur. Avec le TLS mutuel (mTLS), le client présente lui aussi un certificat, et le serveur apprend ainsi quelle charge de travail l'appelle. Les certificats internes proviennent d'une CA privée gérée par l'organisation, en général au sein de sa plateforme de secrets ou de service mesh, car les CA publiques n'émettent pas de certificats pour des noms internes.

Quel nom mettre dans le certificat ? Les noms d'hôte et les adresses IP sont de piètres identités pour des conteneurs. SPIFFE (Secure Production Identity Framework for Everyone, projet diplômé de la Cloud Native Computing Foundation) normalise l'identité de charge de travail : chaque charge de travail reçoit une URI comme `spiffe://fooddelivery.example/ns/prod/sa/orders` et un SPIFFE Verifiable Identity Document à courte durée de vie, soit un certificat X.509 pour le mTLS, soit un JWT. Son implémentation de référence, SPIRE, atteste une charge de travail (vérifie de quel compte de

service Kubernetes ou de quelle instance cloud il s'agit réellement) avant d'émettre le document, et en assure la rotation automatiquement, en général toutes les heures environ. Les règles d'autorisation peuvent alors dire *orders peut appeler payments* au lieu de désigner des adresses.

Les service meshes

Implémenter le mTLS, la rotation des certificats, les nouvelles tentatives et l'autorisation par appel dans chaque service, dans chaque langage, coûte cher et donne des résultats incohérents. Un service mesh (maillage de services) déplace ce travail dans l'infrastructure. Dans la conception classique, chaque instance de service reçoit un proxy sidecar (Envoy dans Istio, un micro-proxy en Rust dans Linkerd) qui intercepte son trafic ; un plan de contrôle émet les identités, distribue les politiques et collecte la télémétrie. Des conceptions plus récentes sans sidecar, comme le mode ambient d'Istio, confient le mTLS à un proxy partagé par nœud et n'ajoutent un proxy distinct que là où une politique au niveau HTTP est nécessaire. NIST SP 800-207A (septembre 2023) décrit ce patron comme la manière d'appliquer le zero trust aux applications cloud native : une segmentation fondée sur l'identité, avec des politiques appliquées par le service mesh à chaque appel.

Un service mesh est une décision de plateforme importante. Il ajoute de la latence à chaque saut (généralement faible, mais mesurable au p99), de la mémoire par pod, un plan de contrôle qui doit lui-même être hautement disponible et sécurisé, et une courbe d'apprentissage pour chaque équipe. Il ne protège en outre que le trafic qui le traverse : un traitement qui se connecte directement à la base de données le contourne. Les petits systèmes de quelques services s'en sortent souvent mieux avec un mTLS fourni par une bibliothèque ou par les équilibreurs de charge de la plateforme. Les questions décisives sont le nombre de services et de langages que vous avez, et l'existence ou non d'une équipe plateforme pour l'exploiter (voir la section 10.7).

Automatiser les certificats

Les certificats expirent, et les certificats expirés sont l'une des causes les plus courantes de pannes évitables. Le 6 décembre 2018, un certificat expiré dans un logiciel Ericsson utilisé par l'opérateur mobile britannique O2 a privé environ 32 millions de clients de données mobiles pendant la majeure partie d'une journée. Le remède est l'automatisation. Le protocole ACME (Automatic Certificate Management Environment, RFC 8555, 2019), développé pour l'autorité de certification Let's Encrypt, permet à un logiciel de prouver qu'il contrôle un domaine (en servant un jeton en HTTP, en ajoutant un enregistrement DNS ou en répondant à un défi TLS), d'obtenir un certificat et de le renouveler sans intervention humaine. Let's Encrypt, une CA gratuite et à but non lucratif qui a commencé à émettre des certificats en 2015, a fait des

certificats à courte durée de vie renouvelés automatiquement la norme ; des outils comme cert-manager font de même dans Kubernetes, et tous les grands équilibreurs de charge cloud peuvent gérer les certificats pour vous.

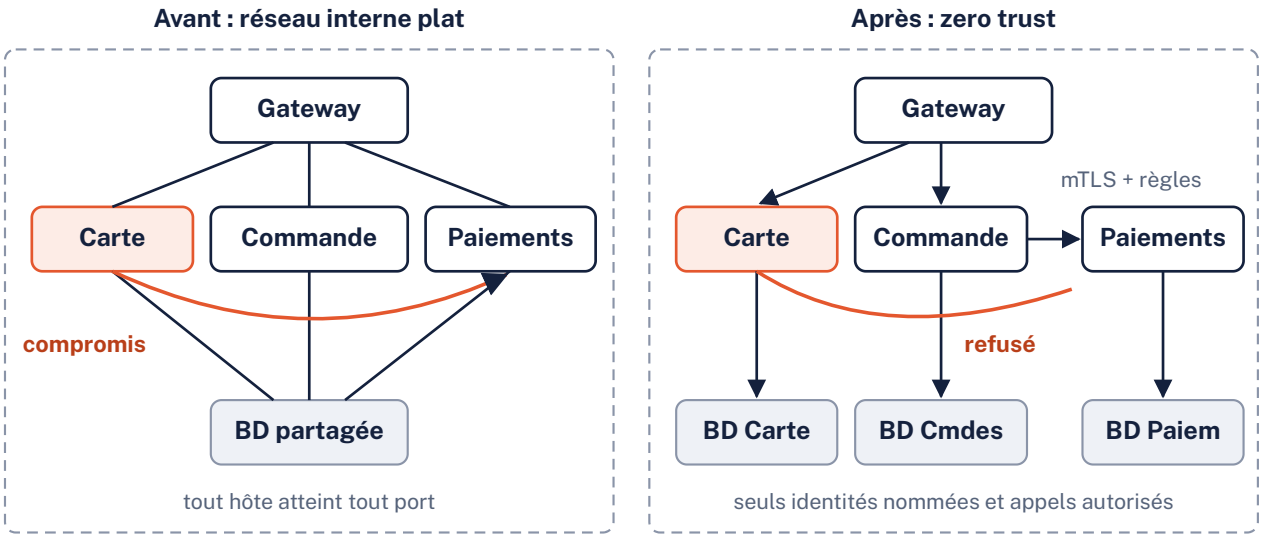
L'automatisation n'est plus facultative. En vertu du vote SC-081 du CA/Browser Forum, la durée de vie maximale des certificats TLS reconnus publiquement est passée à 200 jours pour les certificats émis à partir du 15 mars 2026, et passera à 100 jours à partir du 15 mars 2027 et à 47 jours à partir du 15 mars 2029. Des durées de vie courtes limitent les dégâts d'une clé volée et réduisent la dépendance à la révocation, qui a toujours mal fonctionné. Sur le plan de l'architecture, cela signifie que chaque certificat, public ou interne, a besoin d'un propriétaire, d'un renouvellement automatisé, d'une surveillance des dates d'expiration avec des alertes bien avant l'échéance, et d'une procédure manuelle de secours répétée.

Segmentation et microsegmentation

Le zero trust ne signifie pas que le réseau ne compte plus ; il signifie que le réseau n'est plus le seul contrôle. La segmentation du réseau divise l'infrastructure en zones reliées par des chemins contrôlés : sous-réseaux publics pour les équilibreurs de charge, sous-réseaux privés pour les services, sous-réseaux isolés pour les bases de données, et comptes cloud séparés pour la production, le staging et les outils de sécurité. La microsegmentation réduit les zones à des charges de travail individuelles, de sorte que chaque service ne peut recevoir du trafic que des appelants dont il a besoin et n'en envoyer qu'à ses dépendances déclarées.

Dans Kubernetes, tous les pods peuvent joindre tous les autres pods, sauf si des ressources NetworkPolicy les en empêchent (et ces politiques ne prennent effet que si le plugin réseau du cluster les applique) ; la première étape est donc une politique de refus par défaut pour chaque namespace, suivie d'autorisations explicites. Le contrôle du trafic sortant (egress) compte autant que celui du trafic entrant (ingress) : un service compromis qui ne peut pas ouvrir de connexions vers internet ne peut pas facilement exfiltrer des données volées ni télécharger d'autres outils. Les groupes de sécurité cloud, les network policies Kubernetes et les politiques d'autorisation du service mesh agissent à des couches différentes (adresses et ports, pods et labels, identités de charge de travail et méthodes HTTP) et s'utilisent au mieux ensemble : les couches extérieures donnent une protection grossière qui survit à une erreur de configuration du service mesh, la couche intérieure exprime l'intention avec précision. C'est la défense en profondeur (section 8.1) appliquée au réseau.

D'un réseau plat à une segmentation fondée sur l'identité



Le zero trust n'empêche pas un service d'être compromis ; il limite ce que le service compromis peut atteindre.

Sécuriser l'API gateway

L'API gateway (passerelle d'API) est le principal point d'application entre internet et les services ; elle assume donc plusieurs fonctions. **Authentification :** terminer le TLS, puis valider chaque jeton d'accès (signature avec un algorithme imposé, émetteur, audience, expiration et date de début de validité), rejeter les requêtes qui n'en ont pas, et transmettre aux services une identité vérifiée, jamais les affirmations brutes du client. **Limitation de débit :** limiter les requêtes par client, par utilisateur et par route, généralement avec un token bucket (seau à jetons), en répondant HTTP 429 avec un en-tête `Retry-After` ; des limites distinctes, plus strictes, protègent la connexion, la réinitialisation du mot de passe et l'utilisation des bons de réduction contre le bourrage d'identifiants (credential stuffing) et l'énumération. **Validation des entrées :** imposer des tailles de requête et des nombres d'en-têtes maximaux, n'accepter que les types de contenu déclarés, et valider les corps de requête par rapport au schéma OpenAPI pour que les requêtes malformées n'atteignent jamais un service. Un pare-feu applicatif web (WAF) peut ajouter des signatures d'attaque génériques, mais c'est un complément, pas un substitut à un code correct.

Sachez ce qu'une passerelle ne peut pas faire. Elle voit la route et le jeton, mais pas si la commande 81723 appartient au client qui la demande. La faille d'autorisation au niveau des objets (broken object level authorisation), où un utilisateur modifie un identifiant dans l'URL et lit les données de quelqu'un d'autre, figure en première place de l'OWASP API Security Top 10 dans ses éditions 2019 et 2023, et seul le service qui possède les données peut l'empêcher. Les services doivent donc autoriser eux-mêmes chaque requête, à l'aide de l'identité vérifiée que transmet la passerelle. Quand un service en appelle un autre pour le compte d'un utilisateur, il

doit transmettre un jeton qui nomme à la fois l'utilisateur et le service appelant, obtenu par exemple par l'échange de jetons OAuth 2.0 (RFC 8693), plutôt que son propre identifiant tout-puissant ; sinon, un bogue dans un service en fait un député confus (confused deputy) capable d'agir pour n'importe qui.

EXEMPLE COMMENTÉ

Exemple commenté : faire passer la plateforme de livraison de repas d'un réseau plat au zero trust

La plateforme fait tourner une vingtaine de services dans un seul cluster Kubernetes et un seul compte cloud. Les services s'appellent en HTTP simple, un même utilisateur de base de données est partagé par six services, le personnel accède à la console d'administration par un VPN, et le modèle de menaces de la section 8.2 a désigné le mouvement latéral après une compromission quelconque comme le principal risque. Un basculement en une seule fois casserait le trafic : la migration se fait donc par étapes, et chaque étape est réversible.

1. Inventorier et observer (semaines 1 à 3)	Recenser chaque service, son propriétaire et ses appelants. Installer le service mesh en mode permissif, qui accepte à la fois le trafic en clair et le trafic mTLS, et utiliser sa télémétrie pour construire le graphe d'appels réel. Surprise : le traitement d'analytique appelle directement la base de données de Paiements.
2. Donner une identité à chaque charge de travail (semaines 3 à 5)	Un compte de service Kubernetes par service ; le service mesh émet des certificats SPIFFE valables 24 heures, renouvelés automatiquement. L'utilisateur de base de données partagé est remplacé par un identifiant par service, délivré par le gestionnaire de secrets pour une heure à la fois.
3. Tout chiffrer (semaines 5 à 8)	Passer les namespaces un par un en mTLS strict, en commençant par les moins risqués. Des tableaux de bord affichent les appelants qui communiquent encore en clair ; chacun est corrigé avant la bascule de son namespace. Le retour arrière tient en une modification de configuration par namespace.
4. Autoriser chaque appel (semaines 8 à 12)	Rédiger des politiques d'autorisation à partir du graphe d'appels observé : Commandes peut appeler Paiements sur `POST /authorisations` ; Carte ne peut rien appeler dans Paiements. Les faire tourner en mode audit pendant deux semaines, examiner les refus, puis les appliquer. Ajouter des NetworkPolicies Kubernetes de refus par défaut et des règles de trafic sortant comme couche extérieure.
5. Remplacer le VPN du personnel (semaines 10 à 16)	Placer la console d'administration et les outils internes derrière un identity-aware proxy qui exige l'authentification unique (SSO), une MFA résistante à l'hameçonnage et un appareil géré et à jour de ses correctifs. Les remboursements au-delà d'un seuil exigent une nouvelle étape de MFA. Le VPN est coupé dès que plus rien ne l'utilise.
6. Renforcer la périphérie (en parallèle)	La passerelle valide les jetons, applique des limites de débit par route et une validation par schéma, et transmet une identité vérifiée ; chaque service vérifie quand même que l'appelant a le droit de voir la commande ou le compte précis.

7. Mesurer et maintenir dans la durée	Des fonctions d'aptitude font échouer le build si un nouveau service n'a pas de politique d'autorisation, et des alertes se déclenchent sur les appels refusés et sur les certificats à moins de sept jours de leur expiration. Chaque nouvelle dépendance est une modification de politique revue.

L'ordre compte : observer avant d'appliquer, l'identité avant la politique, le mode audit avant le refus. Chaque étape peut être annulée séparément, ce qui maintient le risque de la migration en dessous du risque qu'elle élimine.

Le résultat n'est pas que plus rien ne peut être compromis. Un service Carte compromis peut encore faire mauvais usage de ses propres données et des appels qui lui sont permis, mais il ne peut plus atteindre les paiements ni les bases de données des autres services, et ses tentatives sont journalisées et déclenchent des alertes.

ATTENTION

Idées reçues courantes sur le zero trust

- **Ce n'est pas un produit.** Le NIST décrit le zero trust comme des principes et une architecture ; les fournisseurs vendent des composants (proxys, fournisseurs d'identité, service meshes) qui en implémentent une partie.
- **Il ne supprime pas les contrôles réseau.** La segmentation et les pare-feu restent des couches extérieures ; ils ne sont simplement plus la seule chose entre un attaquant et les données.
- **Le mTLS seul n'est pas du zero trust.** Un chiffrement avec identité mais une politique qui autorise tout laisse encore n'importe quel service compromis appeler n'importe quoi ; c'est l'autorisation par appel qui compte.
- **Il ne s'achève pas en un seul projet.** Le modèle de maturité de la CISA existe parce que les organisations progressent pilier par pilier, sur plusieurs années.
- **Il ne doit pas rendre la sécurité inutilisable.** L'acceptabilité psychologique de Saltzer et Schroeder s'applique toujours : des demandes fréquentes apprennent aux gens à tout approuver, il faut donc réserver les contrôles renforcés aux actions risquées.

Questions fréquentes

Avons-nous encore besoin d'un VPN ?
Pour les personnes qui accèdent à des applications web, un identity-aware proxy le remplace généralement. Les VPN ou les liaisons privées restent raisonnables pour les connexions de site à site et pour les protocoles qu'un proxy ne sait pas gérer, mais ils ne doivent pas accorder une confiance large à tout ce qui se trouve de l'autre côté.

Un service mesh est-il indispensable au zero trust ?
Non. C'est un moyen pratique parmi d'autres d'obtenir partout le mTLS, l'identité de charge de travail et une politique par appel. Les équilibreurs de charge de la plateforme, des bibliothèques et les fonctions d'identité du cloud peuvent obtenir le même résultat pour un petit nombre de services.

Le zero trust ralentit-il le système ?
Chaque contrôle ajoute un peu de latence, généralement bien moins d'une milliseconde pour un proxy local de service mesh, et quelques millisecondes quand une décision à distance est nécessaire. Mettre les décisions en cache, pousser la politique vers le point d'application et réutiliser les connexions permettent de la limiter ; mesurez-la au p99 sur les chemins critiques.

La sécurité du navigateur : la politique de même origine, CORS et CSRF

Jusqu'ici, les appelants étaient des services et des apps qui détiennent leurs propres identifiants. Un navigateur est différent : il exécute côte à côte du code de nombreux sites et joint automatiquement les cookies de l'utilisateur. Le site web client et le portail des restaurants de la plateforme ont donc une frontière de confiance à l'intérieur du navigateur du client, là où aucune passerelle ne peut atteindre. Le point de départ est la **politique de même origine** (same-origin policy). Deux URL ne partagent une origine que si le schéma, l'hôte et le port correspondent tous, et le script d'une origine est limité dans sa façon d'accéder aux ressources d'une autre. La restriction porte surtout sur la *lecture* : les lectures entre origines sont généralement bloquées, tandis que les écritures entre origines, comme les soumissions de formulaires, sont généralement autorisées.

Le **Cross-Origin Resource Sharing** (CORS) est l'assouplissement contrôlé de cette règle. Si le site `app.example.com` appelle `api.example.com`, ce sont des origines différentes : l'API doit donc répondre avec `Access-Control-Allow-Origin` en nommant le site. Pour tout ce qui va au-delà d'une requête simple (un `PUT` ou un `DELETE`, un en-tête personnalisé, un corps JSON), le navigateur demande d'abord la permission par une requête préalable `OPTIONS` (preflight). Trois points sont souvent mal compris. Le navigateur applique CORS, et le serveur ne fait que

déclarer une politique : des outils comme `curl` l'ignorent donc, et l'API doit quand même authentifier et autoriser chaque appel. CORS ne fait jamais qu'assouplir la politique de même origine ; il n'ajoute aucune protection. Et une requête simple atteint quand même le serveur et peut produire son effet : seule la réponse est retenue. Une origine générique est refusée pour les requêtes avec identifiants, et renvoyer tel quel n'importe quel `Origin` reçu annule tout l'intérêt : comparez-le donc à une liste d'autorisation.

La **falsification de requête intersites** (CSRF) exploite précisément cette faille. Une page d'un autre site amène le navigateur à envoyer une requête à la plateforme, le cookie du client peut l'accompagner, et le serveur voit ce qui ressemble au client en train de changer une adresse de livraison. Les principales défenses de l'OWASP sont un jeton imprévisible contrôlé à chaque requête qui change l'état (par session lorsque le serveur conserve un état, un cookie double-submit signé dans le cas contraire), ou un contrôle de l'en-tête `Sec-Fetch-Site`, avec repli sur la vérification d'`Origin`. L'attribut de cookie `SameSite` de la section 8.3 est une défense en profondeur utile, mais pas un remplacement : `Lax` envoie encore les cookies lors de navigations `GET` intersites de premier niveau ; un `GET` ne doit donc jamais changer l'état. Un jeton envoyé par script dans un en-tête `Authorization` n'est pas joint automatiquement et n'est donc pas exposé de cette façon, mais un backend for frontend réintroduit les cookies et a besoin d'une protection contre le CSRF. Un avertissement : le cross-site scripting met en échec toutes ces défenses, ce qui mène au contrôle suivant.

Content Security Policy et en-têtes de sécurité

Content Security Policy (CSP) est un en-tête de réponse qui indique au navigateur ce qu'une page peut charger et exécuter. Son principal usage est de limiter les dégâts du cross-site scripting : avec une politique en place, les scripts inline injectés, les gestionnaires d'événements inline et `eval()` sont bloqués, sauf si la politique les autorise explicitement. La spécification du W3C indique clairement que CSP s'utilise de préférence comme défense en profondeur : elle réduit les dégâts d'une injection, mais ne remplace pas la validation des entrées ni l'encodage des sorties. CSP Level 3 est encore un working draft : vérifiez donc la version actuelle et la prise en charge par les navigateurs.

Les listes d'hôtes autorisés sont souvent contournables et difficiles à maintenir ; les recommandations actuelles privilégient donc une **CSP stricte** fondée sur des nonces ou des hachages, par exemple `script-src 'nonce-RANDOM'; object-src 'none'; base-uri 'none'`. Un nonce doit être une nouvelle valeur imprévisible à chaque réponse, ce qui a une conséquence architecturale : les pages doivent être générées à chaque réponse, alors que les pages statiques en cache conviennent aux hachages, qui doivent être recalculés chaque fois qu'un script change. Chaque script tiers, comme un widget de paiement ou une balise d'analyse, doit être inventorié et autorisé. Le déploiement sûr passe par `Content-Security-`

`Policy-Report-Only` : la politique signale les violations à un endpoint sans rien bloquer, et elle est imposée une fois les rapports propres. La directive `frame-ancestors 'none'` empêche aussi d'autres sites d'encadrer la page de paiement, une défense contre le clickjacking.

Quelques autres en-têtes sont peu coûteux et méritent d'être définis partout. `Strict-Transport-Security` (HSTS, RFC 6797) indique au navigateur de n'utiliser que HTTPS pour le site ; comme la toute première visite n'est pas protégée, les sites peuvent aussi demander leur inscription sur la liste de préchargement des navigateurs. `X-Content-Type-Options: nosniff` empêche le navigateur de deviner les types de contenu. `Referrer-Policy: strict-origin-when-cross-origin` évite que les chemins et les références de commande figurent dans le référent envoyé à d'autres sites. `Permissions-Policy` désactive les fonctions du navigateur dont une page n'a pas besoin ; la page du livreur peut avoir besoin de la géolocalisation, pas la page de la carte. L'OWASP déconseille de compter sur l'ancien en-tête `X-XSS-Protection` et recommande de le mettre à `0` ; consultez sa cheat sheet pour les valeurs actuelles.

La préoccupation de l'architecte est l'endroit où tout cela réside. Si chaque équipe les définit à la main, un service oubliera. Définissez-les une seule fois en périphérie ou à la passerelle, ou dans un modèle partagé, et testez les réponses déployées dans le pipeline comme fonction d'aptitude (section 11.3).

VÉRIFIEZ VOTRE COMPRÉHENSION

▼ Pourquoi un TLS mutuel entre services, où chaque service peut appeler tous les autres, ne suffit-il pas au zero trust ?

Il apporte le chiffrement et l'identité, mais aucune autorisation. Un service compromis peut encore appeler n'importe quoi ; le zero trust exige une politique par appel qui n'autorise que les appels dont chaque identité a réellement besoin.

▼ Quelle est la différence entre le moteur de politique et le point d'application de politique ?

Le moteur de politique décide si une requête doit être autorisée, à partir de la politique et de données d'entrée comme l'identité et la posture ; le point d'application se trouve sur le chemin du trafic et applique cette décision en autorisant ou en bloquant la requête.

▼ Pourquoi les certificats à courte durée de vie sont-ils intéressants, et qu'exigent-ils ?

Une clé volée n'est utile que jusqu'à l'expiration du certificat, ce qui réduit la dépendance à la révocation. Ils exigent une émission et un renouvellement entièrement automatisés, ainsi qu'une surveillance des expirations.

▼ **Un client modifie le numéro de commande dans une URL et voit la commande d'un autre client. Quel composant aurait dû l'empêcher ?**

Le service qui possède les commandes, en vérifiant que le client authentifié est bien propriétaire de cette commande précise. La passerelle a correctement authentifié l'utilisateur, mais ne peut pas savoir à qui appartient chaque objet.

▼ **À vous : Une équipe dit : « La passerelle d'API valide chaque jeton, et nos services sont sur un réseau privé, donc les services n'ont besoin d'aucun autre contrôle. » Qu'est-ce qui pourrait mal tourner, et qu'ajouteriez-vous dans une logique zero trust ?**

Un service compromis ou un identifiant volé peut se déplacer latéralement et tout appeler, car l'emplacement est traité comme une preuve de confiance. Un jeton valide ne montre pas non plus que ce client a le droit de voir cette commande. Donnez à chaque service une identité vérifiée avec TLS mutuel, idéalement de courte durée et émise automatiquement, et n'autorisez que les appels dont il a besoin. Chaque service doit effectuer la vérification au niveau de l'objet sur les données dont il est propriétaire. La passerelle reste utile pour les jetons, la limitation de débit et la validation des entrées.

SOURCES

- NIST SP 800-207: Zero Trust Architecture (2020): https://csrc.nist.gov/pubs/sp/800/207/final
- Ward and Beyer: BeyondCorp, A New Approach to Enterprise Security (2014): https://research.google/pubs/beyondcorp-a-new-approach-to-enterprise-security/
- RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3: https://www.rfc-editor.org/rfc/rfc8446
- RFC 8555: Automatic Certificate Management Environment (ACME): https://www.rfc-editor.org/rfc/rfc8555
- SPIFFE: overview of the SPIFFE standards: https://spiffe.io/docs/latest/spiffe-about/overview/
- MDN: Same-origin policy: https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy
- MDN: Cross-Origin Resource Sharing (CORS): https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CORS
- OWASP Cross-Site Request Forgery Prevention Cheat Sheet: https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
- MDN: Content Security Policy (CSP): https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CSP
- W3C: Content Security Policy Level 3: https://www.w3.org/TR/CSP3/

- OWASP HTTP Headers Cheat Sheet: https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Headers_Cheat_Sheet.html
- RFC 6797: HTTP Strict Transport Security (HSTS): https://www.rfc-editor.org/rfc/rfc6797.html

Sécurité de la chaîne d'approvisionnement logicielle

AVANT DE LIRE

Cette section s'appuie sur :

- 0.3 Organiser le code
- 0.8 Comment les logiciels sont construits
- 8.1 Principes de sécurité

Termes déjà vus qu'elle utilise : Dépendance, Pipeline CI/CD, Image de conteneur, Surface d'attaque, Moindre privilège

L'IDÉE EN BREF

Un restaurant ne cultive pas sa propre farine et ne fabrique pas ses propres fours ; il fait confiance à des fournisseurs. Si la farine d'un fournisseur est contaminée, tous les restaurants qui l'ont achetée sont touchés, aussi propre que soit leur cuisine. Les bons restaurants tiennent la liste de chaque ingrédient et de sa provenance, achètent auprès de fournisseurs qui scellent et étiquettent leurs livraisons, et vérifient les scellés à la réception. Le logiciel se construit de la même façon, à partir de centaines de composants écrits par d'autres et assemblés par des machines de build. La sécurité de la chaîne d'approvisionnement consiste à savoir ce que contient votre logiciel, d'où cela vient, et que personne ne l'a altéré en chemin.

CE QU'IL FAUT SAVOIR

- L'essentiel du code d'un service moderne provient des dépendances, des outils de build, des actions de CI et des images de base ; chacun d'eux, et toute personne qui peut le modifier, fait partie de la surface d'attaque.
- event-stream (2018) a abusé de la confiance accordée à un mainteneur, SolarWinds (2020) d'un système de build compromis, Log4Shell (2021) d'une bibliothèque vulnérable enfouie au fond des arbres de dépendances, et xz utils (2024) d'une prise de contrôle patiente d'un processus de publication.
- Une SBOM au format SPDX ou CycloneDX liste les composants avec des identifiants exacts, de sorte que l'exposition à une nouvelle vulnérabilité se trouve par simple requête ; le décret présidentiel américain 14028 (2021) et le règlement européen sur la cyberrésilience ont généralisé les SBOM.

- SLSA (version 1.2) définit des niveaux Build L0 à L3, qui aboutissent à des builds renforcés et isolés, et des niveaux Source allant jusqu'à la revue par deux personnes ; la provenance indique qui a construit un artefact, et à partir de quoi.
- La signature sans clé de Sigstore associe un certificat à courte durée de vie à une identité de CI et inscrit les signatures dans un journal de transparence public ; des contrôles d'admission vérifient signatures et provenance avant le déploiement.
- Des lockfiles avec empreintes, l'épinglage des actions de CI et des images par digest, de petites mises à jour automatisées avec un court délai pour les nouvelles versions, une analyse priorisée et des builds éphémères et hermétiques réduisent le risque au quotidien.
- Les obligations de signalement du CRA s'appliquent à partir du 11 septembre 2026 (alerte précoce sous 24 heures, notification sous 72 heures), et la plupart des autres obligations à partir du 11 décembre 2027.

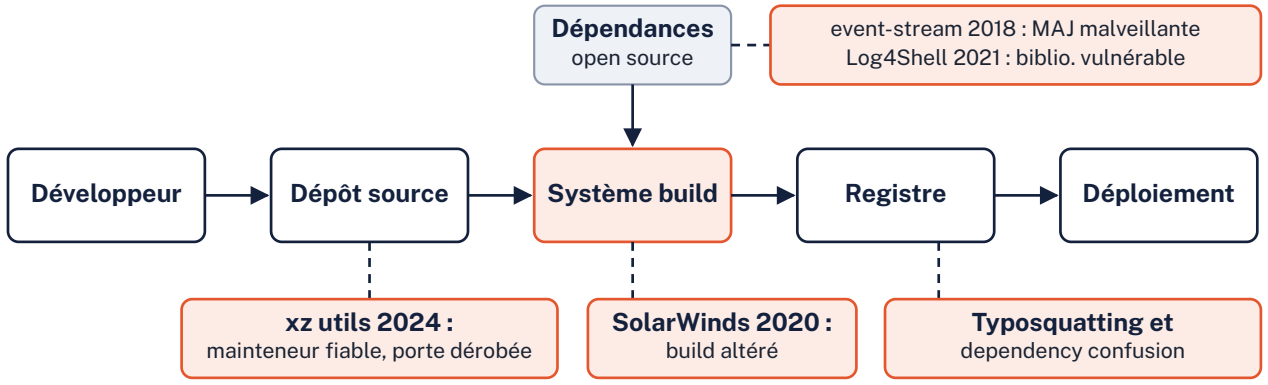
Pourquoi les dépendances sont une surface d'attaque

L'essentiel du code d'un service moderne n'a pas été écrit par l'équipe qui l'exploite. Un petit service web peut déclarer vingt dépendances directes et, par le jeu de leurs propres dépendances, installer plusieurs centaines de paquets ; le conteneur dans lequel il est livré ajoute l'espace utilisateur d'un système d'exploitation, et le pipeline qui le construit exécute des compilateurs, des plug-ins et des actions de CI écrits par d'autres personnes encore. La section 0.3 a présenté les dépendances comme une économie d'effort assortie d'un coût de maintenance. Cette section s'intéresse à leur coût en matière de sécurité : chacun de ces composants, et chaque personne ou système capable de les modifier, fait partie de votre surface d'attaque.

Une attaque de la chaîne d'approvisionnement logicielle compromet quelque chose que vous consommez afin de vous atteindre, généralement en touchant de nombreuses victimes à la fois. Elle peut arriver par quatre grandes voies. **Du code malveillant provenant d'une source de confiance :** le compte d'un mainteneur est piraté par hameçonnage, un mainteneur devient hostile, ou un projet est confié à un inconnu. **Des imitations malveillantes :** le typosquatting publie `reqeusts` à côté de `requests` ; la *dependency confusion* (confusion de dépendances), démontrée par Alex Birsan en février 2021, publie sur le registre public un paquet portant le même nom que le paquet interne d'une entreprise et un numéro de version plus élevé, que certains outils de build préfèrent alors. **Une altération dans le pipeline :** le code source est sain, mais le système de build, les clés de signature ou le registre sont compromis. **Des vulnérabilités involontaires :** aucun attaquant n'a participé à l'écriture du code, mais un bogue grave dans une dépendance transitive très utilisée expose tous ceux qui la livrent.

Les dernières années ont montré à quelle vitesse la première voie se propage. En mars 2025, les tags de version d'une GitHub Action populaire, tj-actions/changed-files, ont été déplacés vers un commit malveillant qui affichait les secrets de CI dans les journaux de build, touchant des milliers de dépôts qui référençaient l'action par tag. En septembre 2025, un mainteneur de paquets npm très utilisés comme `chalk` et `debug` a été piégé par hameçonnage au moyen d'un faux site de support npm, et des versions malveillantes ont été publiées puis retirées en quelques heures ; plus tard le même mois, un ver auto-propagateur, surnommé Shai-Hulud, a volé des jetons npm et s'en est servi pour publier des versions infectées d'autres paquets.

La chaîne d'approvisionnement et les relations de confiance qu'exploitent les attaquants



Chaque flèche est une relation de confiance ; chaque incident célèbre en a visé une autre

Le code source, le build, le registre et les dépendances sont des relations de confiance distinctes, et chaque incident célèbre a attaqué une relation différente.

Quatre incidents, racontés avec exactitude

event-stream (2018). event-stream était un paquet npm populaire, téléchargé environ deux millions de fois par semaine, dont l'auteur d'origine n'avait plus le temps de s'occuper. En septembre 2018, il a cédé les droits de publication à un bénévole qui avait proposé son aide. Le nouveau mainteneur a ajouté une dépendance, `flatmap-stream`, dont le paquet publié contenait une charge utile chiffrée qui ne s'activait que dans le build de Copay, une application de portefeuille bitcoin, et tentait de voler les portefeuilles aux soldes importants. Un développeur a remarqué le code suspect en novembre 2018. Rien dans le modèle de npm n'avait été cassé : un nom de confiance avait simplement changé de mains.

SolarWinds (2020). Des attaquants, attribués plus tard par le gouvernement américain au service de renseignement extérieur russe, ont obtenu l'accès à l'environnement de build de SolarWinds, l'éditeur du produit de gestion de réseau Orion. Un implant (baptisé SUNSPOT par les enquêteurs) surveillait les builds d'Orion et remplaçait pendant la compilation un fichier source par une version dotée d'une porte dérobée, puis remettait l'original en place. Les mises à jour qui en résultaient, publiées entre mars et juin 2020, étaient construites par le propre

pipeline de SolarWinds et signées avec son certificat authentique. Environ 18 000 clients les ont installées ; les attaquants ont ensuite choisi un nombre bien plus restreint d'agences gouvernementales et d'entreprises pour poursuivre l'intrusion. FireEye a découvert la campagne en décembre 2020 en enquêtant sur sa propre violation. La leçon pour les architectes : une signature prouve qui a construit quelque chose, pas que le build était sain.

Log4Shell (2021). La CVE-2021-44228, divulguée le 9 décembre 2021 et notée 10,0 selon le CVSS, était une faille de Log4j 2, la bibliothèque de journalisation Java dominante. Log4j résolvait les lookups écrits dans les messages de journal, y compris les lookups JNDI capables de charger une classe depuis un serveur distant ; un attaquant capable de faire journaliser une chaîne, comme un nom d'utilisateur ou un en-tête de requête, pouvait donc exécuter du code sur le serveur. Aucun attaquant n'avait touché à la chaîne d'approvisionnement ; les dégâts venaient de l'ampleur et de la profondeur avec lesquelles la bibliothèque était intégrée. Les organisations ont passé des semaines à trouver où elles l'utilisaient, plusieurs versions correctives successives ont été nécessaires pour combler des failles apparentées, et celles qui disposaient d'un inventaire exact ont répondu à la question en quelques heures.

xz utils (2024). xz est une bibliothèque de compression utilisée, par l'intermédiaire d'autres bibliothèques, par le serveur OpenSSH sur plusieurs distributions Linux. Pendant environ deux ans, un compte nommé Jia Tan a apporté des contributions utiles et, aidé par d'autres comptes qui pressaient le mainteneur d'origine, débordé, d'accepter de l'aide, est devenu co-mainteneur. Les versions 5.6.0 et 5.6.1, publiées en février et mars 2024, contenaient une porte dérobée qui aurait permis au détenteur d'une clé privée précise d'exécuter des commandes via SSH. Le code malveillant était caché dans des fichiers de test binaires et activé par un script de build modifié qui n'existait que dans les tarballs de publication, pas dans le dépôt public. Le 29 mars 2024, Andres Freund, développeur chez Microsoft, l'a révélé après avoir cherché pourquoi les connexions SSH sur un système de test consommaient un temps processeur inhabituel et prenaient environ une demi-seconde de plus ; la faille a été suivie sous la référence CVE-2024-3094. Comme elle a été détectée tôt, elle n'avait atteint pour l'essentiel que des distributions de développement et en publication continue (rolling release), et non des distributions stables.

Ce que chaque incident enseigne

Incident	Ce qui s'est passé	Maillon attaqué	Contrôles qui auraient aidé
event-stream (npm), 2018	Un nouveau mainteneur bénévole, à qui l'auteur d'origine, épuisé, avait confié le paquet, a ajouté une dépendance (flatmap-stream) contenant une charge utile chiffrée visant une application de portefeuille bitcoin précise.	Confiance envers les mainteneurs et dépendances transitives	Lockfiles et délai avant d'adopter les nouvelles versions ; revue des nouvelles dépendances transitives ; financement et succession pour les paquets critiques.
SolarWinds Orion, 2020	Des attaquants présents dans le réseau de l'éditeur ont implanté du code qui remplaçait un fichier source pendant le build, si bien que des mises à jour légitimement signées contenaient la porte dérobée SUNBURST.	Système build	Builds isolés, éphémères et surveillés ; provenance ; builds reproductibles pour comparer les résultats ; moindre privilège pour l'outil de gestion de réseau lui-même.
Log4Shell (Log4j 2), 2021	Une fonctionnalité ancienne qui résolvait les lookups JNDI dans le texte journalisé permettait à tout attaquant capable de faire journaliser une chaîne d'exécuter du code sur le serveur.	Une dépendance vulnérable, profondément imbriquée	Une SBOM pour trouver en quelques minutes chaque système touché ; filtrage du trafic sortant ; application rapide et répétée des correctifs.
xz utils, 2024	Après environ deux ans de contributions patientes, un co-mainteneur a livré des tarballs de publication contenant une porte dérobée cachée qui visait OpenSSH sur certaines distributions Linux.	Processus de source et de publication	Construire à partir du dépôt relu, et non de tarballs faits à la main ; builds reproductibles ; examen attentif des fichiers de test binaires et des scripts de build.

Les nomenclatures logicielles

Une nomenclature logicielle (SBOM, software bill of materials) est une liste lisible par machine des composants d'un logiciel, avec leurs versions, leurs identifiants et les relations entre eux. Elle ne rend pas le logiciel plus sûr à elle seule ; elle permet de répondre aux questions. Quand le prochain Log4Shell est annoncé, un stockage interrogeable des SBOM de chaque artefact déployé transforme «sommes-nous touchés, et où ?» en une simple requête.

Deux formats dominent. SPDX, de la Linux Foundation, est né comme format de conformité des licences ; sa version 2.2.1 est la norme internationale ISO/IEC 5962:2021, et SPDX 3.0 (2024) ajoute des profils pour la sécurité, les builds et l'IA. CycloneDX, un projet OWASP conçu pour les cas d'usage de sécurité, est normalisé par Ecma International sous la référence ECMA-424 ; la version 1.7 est sortie en octobre 2025. Les deux peuvent identifier les composants par une package URL (purl), comme `pkg:npm/express@4.21.2`, ce sur quoi s'appuient les bases de données de vulnérabilités pour leurs correspondances. L'idée complémentaire de VEX (Vulnerability Exploitability eXchange) permet à un fournisseur d'indiquer qu'un composant vulnérable listé n'est pas exploitable dans son produit, par exemple parce que la fonction vulnérable n'est jamais appelée, ce qui réduit le bruit pour les clients.

Les pouvoirs publics ont fait entrer les SBOM dans les pratiques courantes. Le décret présidentiel américain 14028, *Improving the Nation's Cybersecurity*, signé le 12 mai 2021, a défini des mesures relatives à la chaîne d'approvisionnement logicielle pour les logiciels vendus à l'administration fédérale, dont la fourniture d'une SBOM aux acheteurs ; la National Telecommunications and Information Administration a ensuite publié les éléments minimaux d'une SBOM (fournisseur, nom du composant, version, autres identifiants uniques, relations de dépendance, auteur des données de la SBOM et horodatage). La politique américaine s'est depuis éloignée des obligations uniformes : en janvier 2026, le mémorandum M-26-05 de l'Office of Management and Budget a abrogé l'obligation antérieure faite aux fournisseurs d'attester de pratiques de développement sécurisé, laissant les agences fixer des exigences fondées sur les risques et demander des SBOM lorsqu'elles le jugent utile. Dans l'UE, le règlement sur la cyberrésilience, décrit plus loin, fait de la nomenclature des logiciels (SBOM) un élément de la documentation technique des produits comportant des éléments numériques.

Générez les SBOM pendant le build, à partir de ce qui a réellement été résolu et empaqueté, plutôt qu'à partir du manifeste écrit par un développeur ; stockez-les avec l'artefact, indexées par son digest ; et transmettez-les à un service qui revérifie chaque SBOM stockée lorsqu'une nouvelle vulnérabilité est publiée.

```
{
  "bomFormat": "CycloneDX",
  "specVersion": "1.7",
  "metadata": {"timestamp": "2026-10-20T09:14:02Z",
               "component": {"bom-ref": "orders-service", "type": "application",
                             "name": "orders-service", "version": "4.12.0"}},
  "components": [
    {"bom-ref": "log4j-core", "type": "library", "name": "log4j-core",
     "group": "org.apache.logging.log4j", "version": "2.24.3",
     "purl": "pkg:maven/org.apache.logging.log4j/log4j-core@2.24.3",
     "hashes": [{"alg": "SHA-256", "content": "9f2c..."}]},
    {"bom-ref": "jackson-databind", "type": "library", "name": "jackson-databind",
     "version": "2.18.2",
     "purl": "pkg:maven/com.fasterxml.jackson.core/jackson-databind@2.18.2"}
  ],
  "dependencies": [
    {"ref": "orders-service", "dependsOn": ["log4j-core", "jackson-databind"]}
  ]
}
```

Une SBOM CycloneDX abrégée pour le service des commandes de la plateforme de livraison de repas. La purl et l'empreinte identifient chaque composant avec exactitude, ce qui permet à un scanner de le confronter aux bases de données de vulnérabilités.

SLSA : des niveaux pour l'intégrité du build

SLSA (Supply-chain Levels for Software Artifacts, prononcé *salsa*) est un cadre, aujourd'hui maintenu au sein de l'Open Source Security Foundation, qui décrit des niveaux progressifs de protection contre les altérations. La version 1.0 (2023) définissait une piste Build ; la version 1.2 a restructuré la spécification en plusieurs pistes et ajouté une piste Source. L'élément central est la provenance : une déclaration signée, généralement au format d'attestation in-toto, indiquant quelle plateforme de build a produit un artefact, à partir de quelle révision du code source, avec quelle définition de build et quels paramètres. Un consommateur vérifie la provenance par rapport à des attentes comme *construit par notre CI à partir de la branche main de github.com/fooddelivery/orders*.

La piste Source comporte quatre niveaux : gestion de versions (L1), historique continu et immuable avec une provenance pour chaque révision (L2), contrôles techniques appliqués en continu, comme les branches protégées (L3), et revue par deux personnes de chaque modification des branches protégées (L4). Pour la plupart des organisations, la cible réaliste est Build L3 sur une plateforme de build partagée et bien gérée, plus Source L3 ou L4 pour les dépôts importants ; les niveaux disent ce qu'une plateforme doit garantir, et une équipe plateforme peut les atteindre une fois pour tous les services.

Les niveaux de la piste Build de SLSA

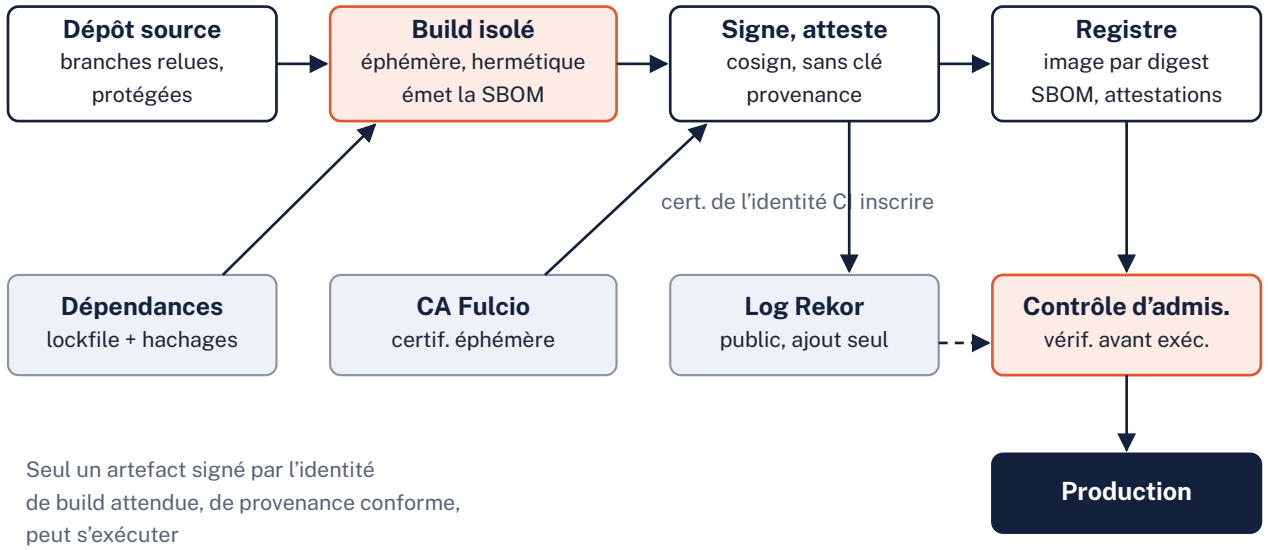
Niveau	Nom	Ce qu'il garantit
Build L0	Aucune garantie	Aucune exigence : l'absence de SLSA.
Build L1	La provenance existe	Le build produit une provenance qui décrit comment l'artefact a été construit. Elle détecte les erreurs, mais est facile à falsifier.
Build L2	Plateforme de build hébergée	Un service de build hébergé génère et signe la provenance, ce qui protège contre les altérations après le build.
Build L3	Builds renforcés	La plateforme de build isole les builds les uns des autres et garde le matériel de signature hors de portée des étapes de build définies par l'utilisateur, ce qui protège contre les altérations pendant le build.

Signature et provenance avec Sigstore

Signer les artefacts est possible depuis longtemps, mais la gestion de clés de signature à longue durée de vie était si contraignante que peu de projets open source le faisaient, et les clés volées constituaient elles-mêmes un risque. Sigstore, lancé en 2021 et désormais rattaché à l'Open Source Security Foundation, rend la signature peu coûteuse grâce à la signature *sans clé* (keyless). Un job de CI prouve son identité à Fulcio, l'autorité de certification de Sigstore, avec un jeton OpenID Connect ; Fulcio émet un certificat valable quelques minutes qui nomme cette identité, par exemple un workflow précis dans un dépôt précis ; l'outil signe l'artefact et inscrit la signature dans Rekor, un journal de transparence public en ajout seul ; la clé privée éphémère est ensuite détruite. Les vérificateurs contrôlent la signature, l'identité figurant dans le certificat et l'entrée du journal, au lieu de faire confiance à une clé qui aurait pu fuiter.

L'outil cosign applique ce principe aux images de conteneur et à d'autres artefacts, et les écosystèmes de paquets ont adopté les mêmes idées : npm peut publier des paquets avec des déclarations de provenance qui les relient au dépôt source et au build (2023), et PyPI prend en charge la publication de confiance (trusted publishing) depuis la CI sans jetons à longue durée de vie (2023) et des attestations fondées sur Sigstore (2024). Au déploiement, un contrôleur d'admission dans Kubernetes, comme le policy controller de Sigstore ou Kyverno, peut refuser toute image qui n'est pas signée par l'identité attendue ou qui n'a pas la provenance attendue. Souvenez-vous de SolarWinds : la signature et la provenance prouvent d'où vient quelque chose ; seuls l'isolation du build, la revue et la détection rendent fiable ce qui en sort.

Un pipeline renforcé, du commit à la production



Chaque étape produit des preuves que la suivante peut vérifier, si bien que la production n'exécute que ce que le pipeline attendu a construit à partir d'un code source relu.

Lockfiles, épinglage et mises à jour automatisées

Un manifeste comme `package.json` ou `pyproject.toml` indique des plages de versions acceptables ; un lockfile (fichier de verrouillage : `package-lock.json`, `poetry.lock`, `uv.lock`, `go.sum`, `Cargo.lock`) enregistre les versions exactes résolues, généralement avec une empreinte cryptographique de chacune. Commiter le lockfile et installer à partir de lui en CI (`npm ci`, `pip install --require-hashes`) garantit que chaque build utilise les mêmes octets relus, et qu'un paquet republié avec un contenu différent échoue à la vérification d'empreinte. L'épinglage s'applique aussi à l'outillage de build : référencez les actions de CI par leur empreinte de commit complète, et non par un tag que leur propriétaire, ou un attaquant, peut déplacer, ce qui est précisément ce qui a permis à l'incident tj-actions de se propager ; référencez les images de base par digest.

Épingler sans mettre à jour crée le risque inverse : les anciennes versions accumulent les vulnérabilités connues et les montées de version deviennent trop lourdes pour être tentées. Des bots de mise à jour comme Dependabot et Renovate ouvrent de petites pull requests fréquentes que la CI teste automatiquement. Un raffinement utile est un âge minimal de publication : attendre quelques jours avant d'adopter une nouvelle version de routine, car la plupart des versions malveillantes sont détectées et retirées en quelques heures ou quelques jours ; les correctifs de sécurité en sont exemptés et passent en priorité.

Les outils d'analyse de la composition logicielle (SCA), comme OWASP Dependency-Check, Trivy, Grype ou des services commerciaux, confrontent vos dépendances aux bases de données de vulnérabilités. Leur faiblesse est le bruit : des centaines de résultats, pour la plupart dans des chemins de code que vous n'appelez jamais. Priorisez à l'aide des données d'exploitation (la vulnérabilité figure-t-elle dans le catalogue Known Exploited Vulnerabilities de la CISA, quel est son score EPSS ?), de l'analyse d'atteignabilité quand vos outils la proposent, et de l'exposition (le service est-il exposé à internet ?). La section 8.8 traite de la gestion des vulnérabilités en tant que processus.

SI VOUS DÉBUTEZ

Pas à pas : même code, build différent

Le point qui fait le plus souvent trébucher ici est que le même code source peut produire un logiciel différent selon les jours si les dépendances exactes ne sont pas verrouillées. Suivez une bibliothèque au fil de deux jours de builds du service des commandes.

- Le manifeste du service des commandes demande une bibliothèque en « version 2.18 ou toute version 2.x ultérieure ». Lundi, le build récupère la correspondance la plus récente, 2.18.2, et les tests passent.
- Mardi, la version 2.18.3 apparaît, publiée par le mainteneur ou peut-être par un attaquant qui a piraté le compte du mainteneur par hameçonnage. Personne dans l'équipe ne modifie une ligne, et pourtant le build de mardi récupère 2.18.3, car elle correspond elle aussi à la plage.
- La même chose peut se produire dans les propres dépendances transitives de la bibliothèque, que l'équipe n'a jamais choisies.
- Un lockfile empêche cela. Lundi, le gestionnaire de paquets enregistre la version exacte de chaque paquet qu'il a résolu, paquets transitifs compris, généralement avec une empreinte cryptographique : une empreinte du contenu exact du paquet. L'équipe commite le fichier, et la CI n'installe que ce qu'il liste.
- Si la 2.18.2 est un jour republiée avec un contenu différent, l'empreinte ne correspond plus et le build échoue. Les outils de build exigent le même soin : une action de CI désignée par un tag peut être déplacée vers un autre code, comme lors de l'incident tj-actions, alors qu'une empreinte de commit complète désigne toujours le même code.
- Le verrouillage n'est pas une fin en soi. Les anciennes versions accumulent les vulnérabilités connues ; un bot de mise à jour propose donc de petites montées de version que la CI teste. Attendre quelques jours avant d'adopter les versions de routine permet de repérer d'abord la plupart des versions malveillantes ; les correctifs de sécurité passent rapidement.
- **La leçon :** décidez délibérément quand vos dépendances changent. Verrouillez les versions exactes pour que chaque build utilise les mêmes octets relus, puis mettez à jour par petites étapes testées.

Conteneurs et intégrité du build

Une image de conteneur embarque l'espace utilisateur d'un système d'exploitation en plus de votre code, et chaque paquet qu'elle contient fait partie de la chaîne d'approvisionnement. Partez d'images de base minimales (distroless, Alpine ou images de distribution allégées), qui contiennent moins de paquets susceptibles d'être vulnérables ; reconstruisez les images

régulièrement pour que les correctifs de l'image de base arrivent même quand votre code n'a pas changé ; analysez les images en CI, puis en continu dans le registre, car de nouvelles vulnérabilités visant d'anciennes images sont publiées chaque jour ; et déployez les images par digest, pour que ce qui a été analysé soit bien ce qui s'exécute.

Le système de build est la cible la plus précieuse de la chaîne, car il touche à tout, comme l'a montré SolarWinds. Exécutez les builds sur des runners éphémères, créés pour un seul job et détruits ensuite, afin qu'un attaquant ne puisse pas persister d'un build à l'autre. Rendez les builds hermétiques, avec toutes leurs entrées déclarées et récupérées par empreinte, et sans accès réseau pendant la compilation, pour qu'aucune entrée non déclarée ne puisse s'y glisser. Tenez les secrets à longue durée de vie à l'écart de la CI en utilisant la fédération OIDC pour obtenir des identifiants cloud à courte durée de vie pour chaque job, et donnez à chaque pipeline le moindre privilège dont il a besoin, afin qu'une pull request issue d'un fork ne puisse pas atteindre les identifiants de déploiement. Protégez le chemin de mise en production par la protection des branches et une revue par deux personnes. Enfin, les builds reproductibles, où le même code source produit toujours un résultat identique au bit près, permettent à des parties indépendantes de reconstruire et de comparer ; le projet Reproducible Builds et des distributions comme Debian ont rendu reproductibles la plupart de leurs paquets, et une divergence est un indice sérieux d'altération.

Le règlement européen sur la cyberrésilience

Le règlement sur la cyberrésilience (Cyber Resilience Act, CRA), règlement (UE) 2024/2847, fixe des exigences horizontales de cybersécurité pour les *produits comportant des éléments numériques* mis à disposition sur le marché de l'UE : produits matériels et logiciels, y compris leurs solutions de traitement de données à distance. Il est entré en vigueur le 10 décembre 2024 et s'applique par étapes. Les règles relatives à la notification des organismes d'évaluation de la conformité s'appliquent à partir du 11 juin 2026. Les obligations de signalement de l'article 14 s'appliquent à partir du 11 septembre 2026 : un fabricant doit signaler toute vulnérabilité activement exploitée ou tout incident grave ayant des répercussions sur la sécurité de son produit, par une alerte précoce au plus tard 24 heures après en avoir pris connaissance, une notification plus complète au plus tard 72 heures après, puis un rapport final (pour une vulnérabilité, au plus tard 14 jours après la mise à disposition d'une mesure de correction ; pour un incident grave, dans un délai d'un mois), via la plateforme unique de signalement gérée par l'ENISA, l'agence de l'Union européenne pour la cybersécurité. La plupart des autres obligations s'appliquent à partir du 11 décembre 2027.

Les exigences essentielles de l'annexe I relèvent autant de l'architecture que de la procédure : les produits doivent être conçus avec un niveau de sécurité approprié, mis à disposition avec une configuration sécurisée par défaut, limiter les surfaces d'attaque, protéger la confidentialité et l'intégrité des données, minimiser les données, et pouvoir enregistrer les

événements de sécurité et recevoir des mises à jour de sécurité. Les fabricants doivent recenser et documenter les composants, notamment en établissant une nomenclature des logiciels (SBOM) couvrant au moins les dépendances de premier niveau, traiter les vulnérabilités au moyen d'une politique de divulgation coordonnée des vulnérabilités, et fournir des mises à jour de sécurité pendant une période d'assistance qui devrait normalement être d'au moins cinq ans, sauf si le produit est destiné à être utilisé moins longtemps. Les amendes pour manquement aux exigences essentielles ou aux obligations des fabricants peuvent atteindre 15 millions d'euros ou 2,5 % du chiffre d'affaires annuel mondial total, le montant le plus élevé étant retenu.

Le champ d'application est déterminant. Les logiciels libres et ouverts développés ou fournis en dehors d'une activité commerciale ne relèvent pas du CRA, et un régime allégé s'applique aux *intendants de logiciels ouverts* qui soutiennent de tels logiciels destinés à un usage commercial. Le logiciel purement en tant que service (SaaS) échappe généralement au CRA, sauf s'il constitue la solution de traitement de données à distance d'un produit ; les fournisseurs de services cloud et d'autres services numériques peuvent relever plutôt de la directive SRI 2 (NIS2). Savoir si un produit donné entre dans le champ d'application, et quelle procédure d'évaluation de la conformité s'applique, est une question juridique propre à chaque organisation ; la conséquence pour l'architecture est que les SBOM, le traitement des vulnérabilités, les mécanismes de mise à jour et la journalisation de sécurité deviennent des preuves qui doivent exister, et non de bonnes intentions.

> **REMARQUE**
>
> ### Information, et non conseil juridique
>
> Les descriptions du décret présidentiel, du mémorandum de l'OMB et du règlement sur la cyberrésilience résument la façon dont ces mesures touchent l'architecture logicielle. Elles ne constituent pas un conseil juridique ; les obligations dépendent du produit, du marché et du rôle de l'organisation, et doivent être confirmées au moyen des textes officiels et auprès de conseillers qualifiés.

Liste de contrôle : la sécurité de la chaîne d'approvisionnement pour une équipe de service

La plupart de ces points sont d'abord des fonctions de la plateforme, puis des habitudes.

- [ ] **Tenir un inventaire**
  Générez une SBOM pour chaque build et stockez-la avec l'artefact, pour que « sommes-nous touchés ? » soit une requête, et non un projet.
- [ ] **Verrouiller et vérifier les dépendances**
  Commitez les lockfiles, installez avec vérification des empreintes en CI, et faites échouer le build si le lockfile et le manifeste divergent.
- [ ] **Choisir les dépendances délibérément**
  Avant d'en ajouter une, vérifiez sa maintenance, le nombre de mainteneurs, les résultats OpenSSF Scorecard, la licence et la quantité de code qu'elle entraîne de manière transitive.
- [ ] **Mettre à jour en continu, mais pas instantanément**
  Laissez un bot proposer de petites mises à jour chaque semaine ; appliquez un court âge minimal de publication aux mises à jour de routine ; faites passer les correctifs de sécurité en priorité.
- [ ] **Analyser, puis trier**
  Exécutez l'analyse de la composition logicielle et l'analyse des images en CI et en continu dans le registre ; priorisez selon l'exploitabilité et l'atteignabilité, pas selon le nombre.
- [ ] **Épingler les outils de build autant que les bibliothèques**
  Épinglez les actions de CI et les plug-ins sur des empreintes de commit complètes et les images de base sur des digests ; revoyez leurs modifications comme du code.
- [ ] **Isoler et renforcer le build**
  Runners éphémères, aucun secret à longue durée de vie (utilisez la fédération OIDC), jetons au moindre privilège, branches protégées et revue par deux personnes pour les chemins de mise en production.
- [ ] **Signer et enregistrer la provenance**
  Signez les artefacts avec Sigstore ou un équivalent, publiez une provenance SLSA, et visez Build L3 sur la plateforme.
- [ ] **Vérifier au déploiement**
  Des politiques d'admission rejettent les images non signées, signées par une mauvaise identité ou dépourvues de provenance issue du dépôt attendu.

- [ ] **Préparer le prochain Log4Shell**
  Entraînez-vous à trouver et à corriger une bibliothèque dans tous les services en une journée, et connaissez vos obligations de signalement au titre du CRA, des contrats et envers vos clients.

Questions fréquentes

Faut-il vendoriser (copier) toutes les dépendances dans notre dépôt ?
Le vendoring donne du contrôle et permet des builds hors ligne, mais complique les mises à jour et masque les correctifs en amont. Un lockfile avec empreintes, plus un miroir de registre privé qui met en cache les paquets approuvés, donne l'essentiel du contrôle avec moins de maintenance.

Un registre privé suffit-il à arrêter la dependency confusion ?
Seulement si les builds sont configurés pour résoudre les noms internes exclusivement à partir de lui. Utilisez des noms de paquets à portée (scope) ou à espace de noms, enregistrez vos noms internes sur les registres publics, et configurez le gestionnaire de paquets pour qu'un paquet public ne puisse jamais remplacer un paquet interne.

Qui, dans l'organisation, est responsable de la sécurité de la chaîne d'approvisionnement ?
En général, c'est une responsabilité partagée : une équipe plateforme ou sécurité fournit une fois pour toutes le pipeline renforcé, la signature, l'analyse et les politiques ; chaque équipe de service garde ses dépendances peu nombreuses, à jour et justifiées.

VÉRIFIEZ VOTRE COMPRÉHENSION

▼ **Pourquoi les vérifications de signature des clients de SolarWinds ne les ont-elles pas protégés ?**

La porte dérobée a été insérée pendant le build de SolarWinds lui-même, si bien que la mise à jour piégée était authentiquement signée par SolarWinds. Une signature montre qui a produit un artefact, pas que le build était sain.

▼ **Qu'apporte un lockfile par rapport à un manifeste ?**

Le manifeste indique des plages de versions acceptables ; le lockfile enregistre les versions exactes résolues et généralement leurs empreintes, de sorte que chaque build installe les mêmes octets et qu'un paquet modifié échoue à la vérification.

▼ **Pourquoi référencer une action de CI par tag est-il plus risqué que par empreinte de commit ?**

Un tag peut être déplacé par quiconque contrôle le dépôt, y compris un attaquant, si bien que la même référence peut exécuter silencieusement un code différent ; une empreinte de commit complète identifie un contenu immuable.

▼ **À quelle question une SBOM répond-elle, à laquelle un scanner de vulnérabilités seul ne répond pas rapidement sur l'ensemble d'un parc ?**

Quels artefacts déployés, dans quels services et quels environnements, contiennent un composant et une version donnés, de sorte que l'exposition à une vulnérabilité nouvellement annoncée puisse être repérée partout à la fois.

▼ **À vous : Un pipeline de CI utilise build-action@main, une image de base étiquetée latest, et installe des paquets sans lockfile. Personne ne peut dire quels services utilisent une bibliothèque nouvellement vulnérable. Classez les correctifs que vous apporteriez, en donnant une raison pour chacun.**

D'abord, produisez une SBOM par build, afin qu'une nouvelle vulnérabilité puisse être retrouvée par requête en quelques minutes. Ensuite, versionnez les lockfiles avec leurs empreintes et épinglez l'action de CI et l'image de base par digest, car les tags peuvent bouger et quiconque peut les changer peut atteindre votre build. Puis automatisez les petites mises à jour de dépendances, avec un court délai pour les nouvelles versions. Enfin, signez les artefacts avec leur provenance et vérifiez-les au déploiement. Chaque étape limite ce qu'un composant amont compromis peut faire.

SOURCES

- SLSA specification v1.2: https://slsa.dev/spec/v1.2/
- Sigstore documentation: https://docs.sigstore.dev/
- Regulation (EU) 2024/2847 (Cyber Resilience Act), EUR-Lex: https://eur-lex.europa.eu/eli/reg/2024/2847/oj
- NVD: CVE-2024-3094 (xz utils backdoor): https://nvd.nist.gov/vuln/detail/CVE-2024-3094
- CycloneDX specification: https://cyclonedx.org/

Protection de la vie privée dès la conception et RGPD

AVANT DE LIRE

Cette section s'appuie sur :

- 7.5 CQRS et event sourcing
- 8.2 Modélisation des menaces
- 8.4 Protéger les données

Termes déjà vus qu'elle utilise : Crypto-shredding (effacement cryptographique), Résidence des données, Diagramme de flux de données (DFD), Durée de vie (TTL)

L'IDÉE EN BREF

Imaginez que vous prêtiez votre journal intime à un ami. Vous attendez de lui qu'il ne lise que la page que vous lui avez montrée, qu'il ne le photocopie pas, qu'il ne le garde pas indéfiniment et qu'il vous le rende quand vous le demandez. La protection de la vie privée dès la conception intègre ces attentes dans le logiciel dès le départ : ne collecter que ce qui est nécessaire, ne l'utiliser que pour la raison annoncée, ne le conserver que le temps nécessaire, et permettre de montrer aux personnes ce qui est détenu à leur sujet et de le supprimer à leur demande. Des lois comme le RGPD transforment ces attentes en obligations, et beaucoup d'entre elles ne peuvent être respectées que si le système a été conçu pour cela.

CE QU'IL FAUT SAVOIR

- La protection de la vie privée demande s'il faut collecter et utiliser les données ou non, et quel contrôle les personnes conservent ; un système peut être sécurisé et pourtant porter atteinte à la vie privée.
- Le RGPD (règlement (UE) 2016/679) et le RGPD britannique, avec le Data Protection Act 2018 modifié par le Data (Use and Access) Act 2025, partagent les principes de l'article 5 : licéité, loyauté et transparence, limitation des finalités, minimisation, exactitude, limitation de la conservation, intégrité et confidentialité, et responsabilité.
- L'article 25 exige la protection des données dès la conception (mesures appropriées telles que la pseudonymisation) et par défaut (seules les données nécessaires sont traitées, conservées et rendues accessibles), en s'inspirant des sept principes d'Ann Cavoukian.
- Les droits d'accès, de rectification, d'effacement, de limitation, de portabilité et d'opposition exigent une cartographie des données, des gestionnaires d'export et d'effacement par service, des indicateurs de limitation et des listes d'exclusion, avec des

réponses normalement sous un mois.

- Une AIPD est obligatoire avant tout traitement susceptible d'engendrer un risque élevé (article 35) et donne les meilleurs résultats lorsqu'elle est menée en parallèle du modèle de menaces, sur le même diagramme de flux de données.
- Les données pseudonymisées restent des données à caractère personnel pour quiconque peut les réidentifier ; l'anonymisation doit résister à tous les moyens raisonnablement susceptibles d'être utilisés, et elle est difficile à obtenir.
- L'effacement doit atteindre les caches, les index de recherche, l'analytique, les journaux d'événements (souvent par crypto-shredding), les sous-traitants et les sauvegardes (mises hors d'usage jusqu'à leur expiration) ; cette section informe sur la façon dont le droit influe sur l'architecture et ne constitue pas un conseil juridique.

REMARQUE

Information, et non conseil juridique

Cette section explique comment le droit de la protection des données influe sur l'architecture logicielle. Elle résume fidèlement le droit à la date de rédaction mais ne constitue pas un conseil juridique : l'application d'une règle, et la manière dont elle s'applique, dépendent des faits, et les organisations doivent prendre conseil auprès d'avocats qualifiés en protection des données et de leur délégué à la protection des données.

La vie privée est une propriété de conception

La sécurité et la protection de la vie privée se recoupent mais ne sont pas la même chose. La sécurité demande si les données sont protégées contre les personnes qui ne devraient pas y avoir accès. La protection de la vie privée demande aussi si l'organisation elle-même devrait collecter ces données ou non, ce qu'elle a le droit d'en faire, pendant combien de temps, et quel contrôle conserve la personne qu'elles décrivent. Un système peut n'avoir subi aucune violation et pourtant porter atteinte à la vie privée, par exemple en conservant indéfiniment l'historique de positions de chaque livreur ou en réutilisant discrètement les historiques de commandes pour entraîner un modèle de tarification.

La plateforme de livraison de repas montre bien pourquoi la protection de la vie privée relève de l'architecture. Elle détient des noms, des numéros de téléphone et des adresses personnelles ; des historiques de commandes qui peuvent révéler la religion ou l'état de santé à travers les choix alimentaires ; des références de paiement ; et un flux continu de positions des livreurs. Ces données circulent depuis les applications à travers des dizaines de services, d'événements, de caches, de journaux, de stockages analytiques et de sauvegardes, et jusqu'à

des sous-traitants comme le prestataire de paiement, la passerelle SMS et le SaaS du service d'assistance. Les décisions sur l'endroit où sont stockées les données à caractère personnel, sur la clé qui les identifie et sur la façon de les retrouver et de les supprimer coûtent cher à changer plus tard, ce qui est exactement le critère de ce qui est architecturalement significatif.

Le droit en bref : le RGPD et le RGPD britannique

Le règlement général sur la protection des données, règlement (UE) 2016/679, s'applique depuis le 25 mai 2018. Il protège les données à caractère personnel, que l'article 4, point 1, définit comme toute information se rapportant à une personne physique identifiée ou identifiable, directement ou indirectement, notamment par référence à un nom, à un numéro d'identification, à des données de localisation ou à un identifiant en ligne. Le responsable du traitement détermine les finalités et les moyens du traitement et en porte la responsabilité première ; un sous-traitant traite des données pour le compte du responsable du traitement en vertu d'un contrat (article 28), comme le fait la passerelle SMS de la plateforme. Le RGPD s'applique aux organisations établies dans l'Union et, en vertu de l'article 3, aux autres organisations qui offrent des biens ou des services à des personnes se trouvant dans l'Union ou qui suivent leur comportement.

Au Royaume-Uni, le RGPD a été maintenu après le Brexit sous le nom de RGPD britannique (UK GDPR), qui s'applique conjointement avec le Data Protection Act 2018. Le Data (Use and Access) Act 2025 a modifié l'un et l'autre ; la plupart de ses modifications relatives à la protection des données sont entrées en vigueur le 5 février 2026, notamment celles qui touchent les fondements juridiques, la prise de décision automatisée et les transferts internationaux, et l'obligation pour les responsables du traitement de traiter les plaintes relatives à la protection des données s'applique à partir du 19 juin 2026. La Commission européenne a renouvelé ses décisions d'adéquation concernant le Royaume-Uni en décembre 2025, de sorte que les données à caractère personnel peuvent continuer à circuler de l'UE vers le Royaume-Uni sans garanties supplémentaires. Pour les architectes, les deux régimes partagent les mêmes principes, les mêmes droits et les mêmes obligations de conception ; les différences sont des questions de détail pour les juristes.

Les amendes maximales pour les manquements les plus graves s'élèvent à 20 millions d'euros ou 4 % du chiffre d'affaires annuel mondial au titre du RGPD (17,5 millions de livres sterling ou 4 % au titre du RGPD britannique), le montant le plus élevé étant retenu, mais les conséquences les plus fréquentes sont des injonctions de cesser le traitement, une atteinte à la réputation et le coût d'une refonte technique. En novembre 2025, la Commission européenne a proposé des modifications du RGPD dans le cadre d'un paquet *omnibus numérique* ; à la date de rédaction, elles suivent encore la procédure législative et ne changent encore rien.

Les principes de l'article 5 et ce qu'ils impliquent pour les systèmes

Principe	En bref	Conséquence architecturale
Licéité, loyauté et transparence (art. 5, paragraphe 1, point a))	Ne traiter les données que sur un fondement juridique, d'une manière à laquelle les personnes peuvent raisonnablement s'attendre, et leur dire ce que vous faites.	Consigner un fondement juridique par finalité ; générer les mentions d'information à partir de la cartographie des données qu'utilisent les systèmes ; aucun pistage caché.
Limitation des finalités (art. 5, paragraphe 1, point b))	Collecter pour des finalités déterminées et explicites, et ne pas réutiliser les données pour des finalités incompatibles.	Étiqueter les données avec leurs finalités ; vérifier l'étiquette avant d'autoriser un nouveau consommateur (comme un pipeline d'apprentissage automatique) à les lire.
Minimisation des données (art. 5, paragraphe 1, point c))	Adéquates, pertinentes et limitées à ce qui est nécessaire.	Ne pas collecter la date de naissance si un indicateur « plus de 18 ans » suffit ; ne stocker l'historique de positions du livreur que tant qu'il est nécessaire.
Exactitude (art. 5, paragraphe 1, point d))	Exactes et, si nécessaire, tenues à jour.	Un seul propriétaire par fait ; permettre aux personnes de corriger leurs données et propager les corrections aux copies.
Limitation de la conservation (art. 5, paragraphe 1, point e))	Conservées sous une forme permettant l'identification pendant une durée n'excédant pas celle nécessaire.	Une durée de conservation pour chaque jeu de données, appliquée automatiquement par des TTL, des suppressions de partitions et des tâches d'expiration, y compris dans les journaux et les sauvegardes.
Intégrité et confidentialité (art. 5, paragraphe 1, point f))	Une sécurité appropriée, y compris contre le traitement non autorisé, la perte et les dégâts.	Tout ce que couvrent les sections 8.1 à 8.6 : contrôle d'accès, chiffrement, gestion des clés, zero trust, contrôles de la chaîne d'approvisionnement.
Responsabilité (art. 5, paragraphe 2)	Le responsable du traitement est responsable du respect des principes et doit être en mesure de le démontrer.	Les registres des activités de traitement, les AIPD, les enregistrements des consentements, les registres d'effacement et les journaux d'audit sont des preuves que l'architecture doit produire.

Les fondements juridiques, en bref

L'article 6 n'autorise le traitement que sur l'un des six fondements juridiques : le consentement, l'exécution d'un contrat avec la personne, une obligation légale, la sauvegarde des intérêts vitaux, une mission d'intérêt public, ou des intérêts légitimes sur lesquels ne prévalent pas les intérêts et les droits de la personne. Les catégories particulières de données, comme la santé

ou les convictions religieuses, exigent une condition supplémentaire au titre de l'article 9. Le fondement se choisit par finalité, non par système, et il change ce que l'architecture doit faire. Sur la plateforme de livraison de repas, l'adresse de livraison est traitée pour exécuter le contrat ; les factures sont conservées pour respecter une obligation légale découlant du droit fiscal ; la détection de la fraude repose généralement sur les intérêts légitimes ; et les e-mails marketing nécessitent normalement le consentement, que les règles sur la prospection électronique encadrent également.

Le consentement obéit aux mécanismes les plus stricts. En vertu de l'article 7, le responsable du traitement doit être en mesure de démontrer que le consentement a été donné, celui-ci doit être spécifique à une finalité, et il doit être aussi simple de le retirer que de le donner. Le consentement ne peut donc pas être une simple colonne booléenne : le système a besoin d'un enregistrement en ajout seul (append-only) de ce à quoi la personne a consenti, quand, par quel canal et en réponse à quelle formulation, et chaque consommateur doit honorer rapidement un retrait.

```
{
  "event": "ConsentGiven",
  "subjectRef": "cust_4411",
  "purpose": "marketing_email",
  "noticeVersion": "privacy-notice-2026-09",
  "wording": "Send me offers from restaurants near me",
  "channel": "ios-app 7.3.0",
  "method": "unticked checkbox, user ticked",
  "at": "2026-10-03T18:22:41Z"
}
{
  "event": "ConsentWithdrawn",
  "subjectRef": "cust_4411",
  "purpose": "marketing_email",
  "channel": "email unsubscribe link",
  "at": "2026-10-19T07:05:12Z"
}
```

Le consentement sous forme d'enregistrement d'événements en ajout seul : l'état courant est déduit du dernier événement par finalité, et l'historique constitue la preuve qu'exige l'article 7.

Protection des données dès la conception et par défaut

L'article 25 fait de la protection de la vie privée dès la conception une obligation légale. L'article 25, paragraphe 1, la protection des données dès la conception, impose au responsable du traitement, tant au moment de déterminer les moyens du traitement qu'au moment du traitement lui-même, de mettre en œuvre des mesures techniques et organisationnelles appropriées, telles que la pseudonymisation, destinées à appliquer les principes, compte tenu de l'état des connaissances, des coûts, de la nature et des finalités du traitement et des

risques. L'article 25, paragraphe 2, la protection par défaut, exige que seules les données nécessaires au regard de chaque finalité spécifique soient traitées par défaut, ce qui s'applique à la quantité de données collectées, à l'étendue de leur traitement, à leur durée de conservation et à leur accessibilité ; en particulier, les données ne doivent pas être rendues accessibles à un nombre indéterminé de personnes sans l'intervention de la personne concernée. Les lignes directrices 4/2019 du comité européen de la protection des données relatives à l'article 25 donnent des exemples pratiques.

Le droit a emprunté l'idée à Ann Cavoukian, commissaire à l'information et à la protection de la vie privée de l'Ontario, qui a élaboré la protection de la vie privée dès la conception dans les années 1990 et en a publié les sept principes fondamentaux en 2009 ; la conférence internationale des commissaires à la protection des données les a adoptés en 2010. Ils restent une liste de contrôle utile pour les revues de conception.

Les sept principes fondamentaux de la protection de la vie privée dès la conception selon Cavoukian

Principe	Ce qu'il exige d'une conception
1. Mesures proactives et non réactives ; préventives et non correctives	Anticiper les atteintes à la vie privée dès la conception au lieu de les corriger après un incident.
2. Protection de la vie privée par défaut	La personne n'a rien à faire pour être protégée ; le partage facultatif est désactivé jusqu'à ce qu'elle l'active.
3. Protection de la vie privée intégrée dans la conception	La protection de la vie privée fait partie de l'architecture et des processus métier ; ce n'est pas un ajout plaqué après coup.
4. Fonctionnalité intégrale : somme positive, et non somme nulle	Rechercher des conceptions qui atteignent à la fois la protection de la vie privée et l'objectif métier, plutôt que de sacrifier l'une à l'autre.
5. Sécurité de bout en bout : protection pendant tout le cycle de vie	Protéger les données depuis leur collecte jusqu'à leur destruction sécurisée.
6. Visibilité et transparence	Les pratiques sont ouvertes et vérifiables par les utilisateurs et par des examinateurs indépendants.
7. Respect de la vie privée des utilisateurs	Rester centré sur l'utilisateur : des paramètres par défaut protecteurs, une information claire et des choix faciles à exercer.

Les droits que l’architecture doit prendre en charge

C’est par les droits des personnes concernées que le droit de la protection de la vie privée se traduit le plus directement en exigences pour les systèmes. Un responsable du traitement doit normalement répondre dans un délai d’un mois à compter de la réception d’une demande, délai qui peut être prolongé de deux mois au plus pour les demandes complexes ou nombreuses (article 12, paragraphe 3), et doit être capable de retrouver les données de la personne où qu’elles se trouvent. Dans un système de vingt services, cinq sous-traitants SaaS et un entrepôt de données, c’est impossible sans une cartographie des données tenue à jour : le registre des activités de traitement qu’exige déjà l’article 30, conservé assez près du code pour rester exact. Avec les modifications britanniques de 2025, un responsable du traitement qui répond à une demande d’accès n’est tenu d’effectuer que des recherches raisonnables et proportionnées, ce qu’une bonne cartographie des données rend rapide.

Les droits des personnes concernées et ce que chacun exige des systèmes

Droit	Ce que la personne peut demander	Ce dont les systèmes ont besoin
Accès (art. 15)	La confirmation que des données sont traitées, une copie, et des informations comme les finalités, les destinataires, la durée de conservation et la source.	Une cartographie de tous les stockages détenant des données à caractère personnel ; une API d'export par service, indexée par un identifiant client stable ; la vérification de l'identité ; un moyen d'assembler et de transmettre le résultat dans le délai.
Rectification (art. 16)	La correction des données inexactes et le complément des données incomplètes.	Un seul propriétaire faisant autorité par champ ; corrections propagées aux copies et aux caches par des événements ; destinataires informés (art. 19).
Effacement (art. 17)	La suppression lorsque, entre autres motifs, les données ne sont plus nécessaires ou que le consentement est retiré, sous réserve d'exceptions comme les obligations légales.	Un orchestrateur d'effacement, des règles d'effacement par stockage (supprimer, anonymiser, conserver avec un motif), le crypto-shredding pour les stockages immuables, et un registre prouvant l'achèvement.
Limitation (art. 18)	Données conservées mais pas utilisées autrement, par exemple pendant que leur exactitude est contestée.	Un indicateur de limitation que chaque consommateur vérifie, de sorte que les données restent stockées mais exclues des traitements comme le marketing ou l'entraînement de modèles.
Portabilité (art. 20)	Les données fournies par la personne, traitées à l'aide de procédés automatisés sur la base du consentement ou d'un contrat, dans un format structuré, couramment utilisé et lisible par machine.	Un export en JSON ou CSV avec des champs documentés ; éventuellement, la transmission directe à un autre responsable du traitement lorsque c'est techniquement possible.
Opposition (art. 21)	L'arrêt d'un traitement fondé sur les intérêts légitimes ou sur une mission d'intérêt public, sauf motifs impérieux ; absolu pour la prospection.	Des interrupteurs par finalité ; une liste d'exclusion (ne gardant que le minimum nécessaire, comme une adresse e-mail hachée) pour que la personne ne soit pas réinscrite par un import ultérieur.
Décisions automatisées (art. 22)	Des garanties contre les décisions fondées exclusivement sur un traitement automatisé produisant des effets juridiques ou des effets significatifs similaires.	Par exemple, la désactivation automatique d'un livreur : une voie vers un examen humain, une explication et un moyen de la contester.

Les analyses d'impact relatives à la protection des données

L'article 35 exige une analyse d'impact relative à la protection des données (AIPD) avant tout traitement susceptible d'engendrer un risque élevé pour les droits et libertés des personnes, en particulier l'évaluation systématique et approfondie, y compris le profilage, produisant des effets significatifs, le traitement à grande échelle de catégories particulières de données ou de données relatives aux condamnations pénales et aux infractions, et la surveillance systématique à grande échelle d'une zone accessible au public. Les autorités de contrôle publient des listes d'autres cas ; le suivi continu de la position des livreurs et la gestion algorithmique de leur travail en sont des exemples typiques. Une AIPD décrit le traitement et ses finalités, évalue sa nécessité et sa proportionnalité, évalue les risques pour les personnes et consigne les mesures envisagées pour y faire face ; si un risque élevé subsiste, le responsable du traitement doit consulter l'autorité de contrôle avant de commencer (article 36).

Les architectes devraient traiter l'AIPD comme le pendant, pour la vie privée, du modèle de menaces de la section 8.2, réalisée au même moment et sur le même diagramme de flux de données. Le cadre LINDDUN, développé à la KU Leuven, fait pour les menaces sur la vie privée ce que STRIDE fait pour les menaces de sécurité, avec des catégories comme l'association (linking), l'identification, la détection, la divulgation de données et la méconnaissance (unawareness). Les conclusions de l'AIPD deviennent des exigences architecturales, comme *les historiques de positions sont supprimés 30 jours après la livraison*, qui peuvent être testées.

Transferts internationaux et résidence des données

Le chapitre V du RGPD encadre les transferts de données à caractère personnel vers des pays situés hors de l'Espace économique européen. Un transfert est licite si la Commission a constaté que le pays assure un niveau de protection adéquat (article 45), si des garanties appropriées comme les clauses contractuelles types de la Commission sont en place (article 46), ou au titre de dérogations limitées (article 49). Depuis que la Cour de justice a invalidé le bouclier de protection des données UE–États-Unis (Privacy Shield) par l'arrêt Schrems II de juillet 2020, les transferts fondés sur des clauses contractuelles nécessitent une évaluation de la législation du pays de destination ; le cadre de protection des données UE–États-Unis (Data Privacy Framework), jugé adéquat en juillet 2023, couvre les transferts vers les entreprises américaines qui y adhèrent par certification, et a résisté à un premier recours devant le Tribunal de l'Union européenne en septembre 2025 (*Latombe/Commission*), même si d'autres contentieux, y compris un éventuel pourvoi devant la Cour de justice, pourraient encore changer la situation.

Deux points d'architecture en découlent. Premièrement, un transfert n'est pas seulement une base de données dans une autre région : l'accès à distance par une équipe d'assistance à l'étranger, les sous-traitants ultérieurs d'un sous-traitant SaaS et un service de journalisation hébergé ailleurs comptent tous, si bien que la cartographie des données doit indiquer où les données sont stockées et d'où on y accède. Deuxièmement, les exigences de résidence des données, qu'elles soient légales, contractuelles ou choisies, sont plus faciles à satisfaire avec des cellules de déploiement régionales : chaque région détient les données à caractère personnel et les clés de ses clients, les services globaux ne voient que des références pseudonymes, et la réplication entre régions est explicite et revue. Ajouter après coup la résidence des données à une base de données mondiale unique est l'un des changements les plus coûteux qu'une plateforme puisse affronter, c'est pourquoi elle fait partie des contraintes à recueillir tôt.

La conformité hors de l'UE et du Royaume-Uni : SOC 2, ISO 27001, HIPAA et le CCPA

À mesure que la plateforme vend à de grandes chaînes de restaurants ou s'implante dans de nouveaux pays, d'autres noms apparaissent. Ce passage est une information, pas un conseil juridique : l'application d'une loi dépend des faits, et les règles changent ; vérifiez donc la situation actuelle auprès de conseillers qualifiés. Ces noms se répartissent en deux groupes : les dispositifs d'assurance que réclament les clients et les lois qui vous obligent.

SOC 2 est un rapport d'attestation, ni une certification ni une loi. Selon les normes de l'American Institute of CPAs (AICPA), un auditeur indépendant examine les contrôles d'une organisation de services au regard de critères répartis en cinq catégories : sécurité, disponibilité, intégrité du traitement, confidentialité et vie privée. La sécurité est obligatoire dans chaque rapport, l'organisation ajoute les autres catégories adaptées à ses engagements, et aucune liste fixe de contrôles n'est imposée. Un rapport de **type 1** examine la conception des contrôles à un instant donné ; un rapport de **type 2** teste aussi leur fonctionnement effectif sur une période, couramment de six à douze mois, et c'est le plus répandu des deux. **ISO/IEC 27001:2022** (modifiée en 2024 au titre de l'action pour le climat) est une norme internationale d'exigences pour un système de management de la sécurité de l'information : une façon de conduire la sécurité fondée sur les risques, avec 93 mesures de référence à l'annexe A. Contrairement à SOC 2, elle débouche sur un certificat, dont l'utilité ne dépasse pas celle du périmètre qu'il indique.

HIPAA est une loi fédérale américaine, et sa Security Rule (45 CFR, parties 160 et 164) exige des garanties administratives, physiques et techniques pour les informations de santé protégées sous forme électronique. Elle s'impose aux entités couvertes (covered entities : régimes d'assurance maladie, chambres de compensation et de nombreux prestataires de

soins) et à leurs partenaires commerciaux (business associates), mais pas à tous ceux qui détiennent des données de santé. La plateforme se situerait normalement en dehors de son champ, mais livrer des ordonnances pour des pharmacies serait une raison de demander conseil pour savoir si elle est devenue un partenaire commercial. Le HHS a proposé des règles plus sévères fin 2024, notamment le chiffrement et l'authentification multifacteur obligatoires ; vérifiez si elles ont été adoptées définitivement.

Le **California Consumer Privacy Act** (CCPA), tel que modifié par le California Privacy Rights Act, couvre les entreprises à but lucratif qui exercent une activité en Californie et atteignent un seuil : en gros, un chiffre d'affaires annuel brut supérieur à un montant fixé à 25 millions de dollars et ajusté à l'inflation chaque année impaire (26 625 000 dollars depuis janvier 2025 ; vérifiez le chiffre actuel), ou le fait d'acheter, de vendre ou de partager les informations personnelles de 100 000 résidents ou foyers californiens ou plus, ou de tirer au moins la moitié de ses revenus de leur vente ou de leur partage. Les résidents peuvent connaître, supprimer et rectifier leurs données, s'opposer à leur vente ou à leur partage à des fins de publicité intercontextuelle, et limiter l'utilisation des informations sensibles. Des règlements en vigueur depuis janvier 2026 ajoutent des évaluations des risques et, par phases, des audits de cybersécurité. D'autres États américains ont leurs propres lois sur la vie privée.

La leçon d'architecture est que les exigences se recoupent : moindre privilège, chiffrement, journalisation, reprise testée et inventaire des données reviennent dans chacune. Construisez un seul ensemble de contrôles et mettez-le en correspondance avec chaque dispositif, plutôt qu'un programme distinct pour chacun, et réduisez le périmètre par la classification des données et la segmentation, comme la section 8.4 l'a montré pour PCI DSS.

Les preuves de conformité en continu

Un rapport de type 2 demande si un contrôle a fonctionné toute l'année ; des captures d'écran réunies la semaine précédant l'audit sont donc faibles et coûteuses. Mieux vaut collecter les preuves comme un sous-produit du travail normal : journaux de pipeline montrant que chaque mise en production a passé la revue et l'analyse, définitions d'infrastructure, revues d'accès, piste d'audit (section 8.8). Chaque élément est étiqueté avec les contrôles qu'il appuie, et un contrôle en échec déclenche une alerte au moment où il échoue, pas des mois plus tard. La section 11.4 décrit le volet politique en tant que code. L'Open Security Controls Assessment Language (OSCAL) du NIST offre des formats XML, JSON et YAML lisibles par machine pour les catalogues de contrôles, les détails d'implémentation et les résultats d'évaluation, en remplaçant les documents par des données structurées. L'automatisation réunit les preuves ; elle ne remplace ni le jugement de l'auditeur ni les personnes responsables des contrôles.

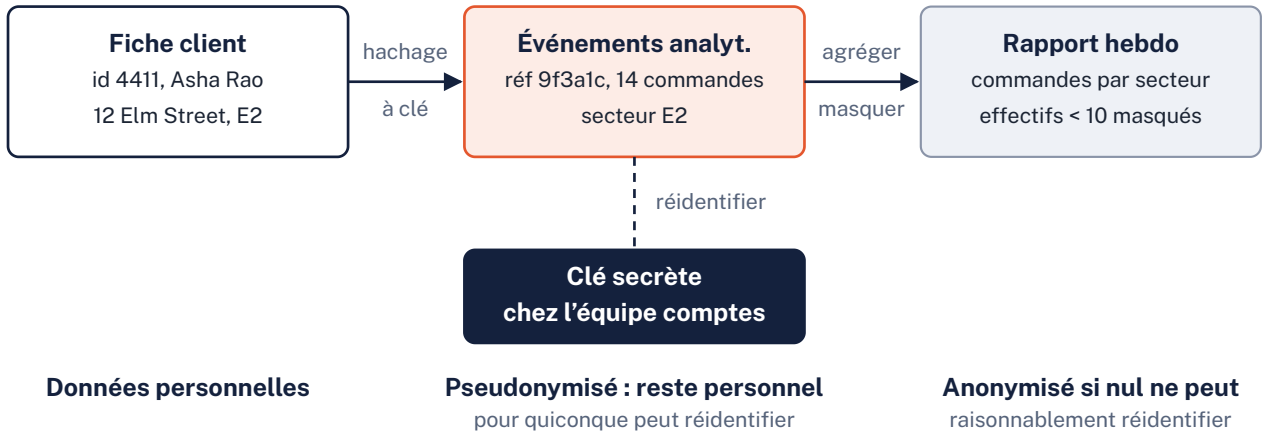
Minimisation, pseudonymisation et anonymisation

Les données à caractère personnel les moins chères à protéger sont celles que vous ne collectez jamais. La minimisation se manifeste dans de petites décisions : un indicateur « plus de 18 ans » au lieu d'une date de naissance, un secteur postal au lieu de l'adresse complète pour l'analytique, un numéro de téléphone masqué relayé par la plateforme au lieu de révéler le numéro du client au livreur, et un échantillonnage de la position seulement pendant qu'une livraison est en cours.

La pseudonymisation (article 4, point 5) traite les données de façon qu'elles ne puissent plus être attribuées à une personne sans recourir à des informations supplémentaires, comme une clé, conservées séparément et protégées. Un hachage à clé (HMAC) de l'identifiant client, dont la clé est détenue par l'équipe chargée des comptes, permet à l'équipe analytique de compter les commandes par client sans voir qui sont les clients. C'est une garantie précieuse que l'article 25 cite expressément, mais le considérant 26 indique clairement que des données pseudonymisées qui pourraient être attribuées à une personne par le recours à des informations supplémentaires restent des données à caractère personnel. Dans l'affaire *CEPD/CRU* (en anglais EDPS v SRB, C-413/23 P, 4 septembre 2025), la Cour de justice a apporté une nuance : le caractère personnel des données s'apprécie du point de vue de celui qui les détient, si bien que des données pseudonymisées peuvent ne pas être des données à caractère personnel pour un destinataire qui n'a aucun moyen raisonnable de réidentifier qui que ce soit, bien qu'elles restent des données à caractère personnel pour le responsable du traitement qui détient la clé, et que celui-ci doive toujours informer les personnes de l'existence de tels destinataires. Les lignes directrices 01/2025 du comité européen de la protection des données sur la pseudonymisation décrivent comment bien la mettre en œuvre.

L'anonymisation va plus loin : les données ne se rapportent plus à une personne identifiable, compte tenu de l'ensemble des moyens raisonnablement susceptibles d'être utilisés pour identifier quelqu'un, et le RGPD ne s'applique alors plus. Une véritable anonymisation est difficile. Retirer les noms ne suffit pas quand la combinaison du code postal, de la date de naissance et du sexe identifie la plupart des personnes, ou quand l'historique de positions d'un seul livreur révèle où il habite. L'agrégation avec masquage des petits effectifs, la généralisation et, pour les statistiques, la confidentialité différentielle sont les techniques habituelles, et le résultat devrait être testé face à des tentatives réalistes de réidentification.

Données à caractère personnel, pseudonymisées et anonymisées



La pseudonymisation réduit le risque mais laisse les données personnelles pour quiconque peut l'inverser ; l'anonymisation doit résister à tout moyen de réidentification raisonnablement susceptible d'être utilisé.

SI VOUS DÉBUTEZ

Pas à pas : un nom caché ne rend pas anonyme

L'idée la plus difficile de cette section est la différence entre données pseudonymisées et données anonymisées. Les deux cachent les noms, mais, comme l'explique la section, le RGPD les traite très différemment. Suivez une étude sur la sécurité des livreurs.

- Une fiche de livreur contient un identifiant, un nom et un numéro de téléphone : ce sont manifestement des données à caractère personnel.
- Pour une étude sur la sécurité routière, la plateforme remplace l'identifiant du livreur par un hachage à clé : un code de référence calculé avec une clé secrète que l'équipe des opérations livreurs conserve séparément. Les chercheurs voient le code, un nombre de livraisons et la zone de la ville, mais ni les noms ni les numéros de téléphone.
- Quiconque détient la clé peut transformer un identifiant de livreur en ce même code et retrouver les lignes de ce livreur. Les données sont pseudonymisées : les noms sont cachés, mais on peut encore distinguer les personnes. Comme la plateforme détient la clé, la règle de la section s'applique : les données restent des données à caractère personnel pour la plateforme, chercheurs compris.
- Cela vaut tout de même la peine : les chercheurs peuvent comparer les charges de travail sans savoir qui est qui.
- Ensuite, la plateforme établit un rapport mensuel des livraisons par zone de la ville, en omettant les zones comptant moins de dix livreurs. Si personne ne pouvait identifier qui que ce soit à partir de ce rapport par un moyen raisonnablement susceptible d'être utilisé, le rapport est anonymisé et, comme l'explique la section, le RGPD ne s'y applique alors plus.
- Retirer les noms ne suffit souvent pas. Le code postal, la date de naissance et le sexe, ensemble, identifient la plupart des personnes, et l'historique de positions d'un livreur peut révéler son domicile.
- **La leçon :** un nom caché ne fait pas une personne anonyme. La pseudonymisation réduit le risque, mais les données restent personnelles pour quiconque peut réidentifier les personnes ; ne qualifiez des données d'anonymes que si personne ne pourrait raisonnablement réidentifier qui que ce soit à partir d'elles, et vérifiez-le par des tentatives réalistes.

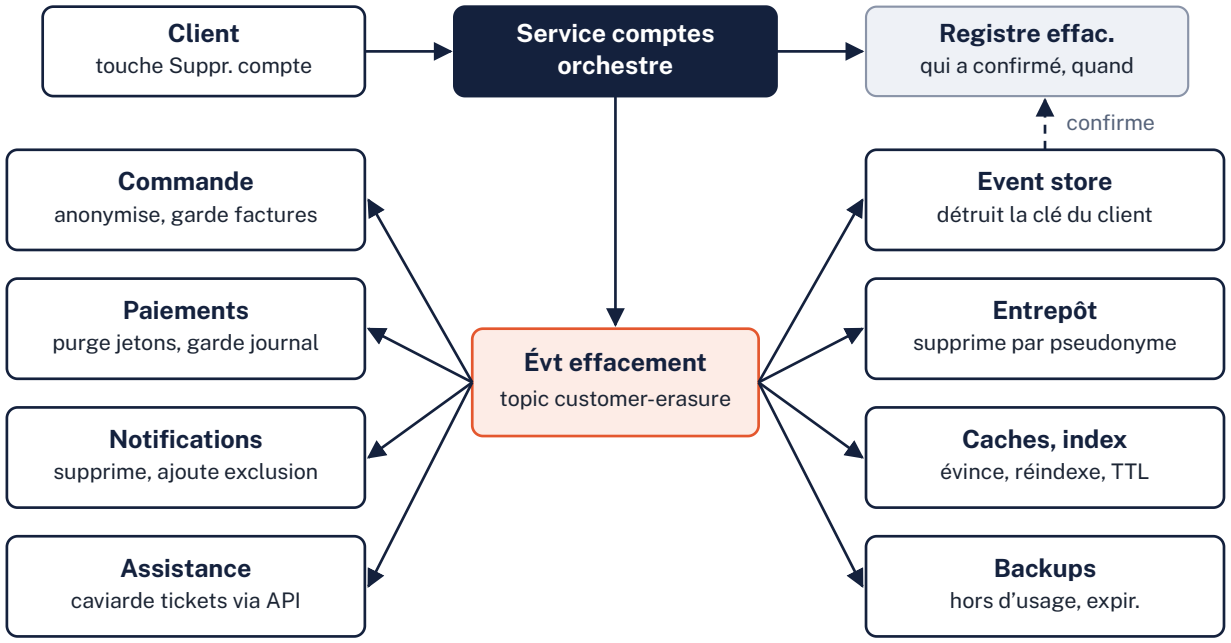
Conservation et effacement dans chaque copie

La limitation de la conservation exige un calendrier de conservation : pour chaque catégorie de données, combien de temps elles sont conservées, pourquoi, et ce qui se passe à l'échéance (suppression, anonymisation ou archivage). Un calendrier ne fonctionne que si les systèmes

l'appliquent automatiquement : TTL sur les entrées de cache et de base clé-valeur, tables partitionnées par période dont les anciennes partitions sont supprimées, règles de cycle de vie du stockage objet, paramètres de conservation des journaux et tâches planifiées dont la réussite est surveillée.

L'effacement est le plus difficile là où les données à caractère personnel s'infiltrent. Les **caches et index de recherche** exigent une éviction et une réindexation explicites, avec des TTL comme filet de sécurité. Les stockages d'**analytique** devraient contenir des références pseudonymes afin que les lignes puissent être retrouvées et supprimées, ou bien les données devraient être anonymisées dès leur entrée. Les **journaux d'événements et les stockages en event sourcing** sont immuables par conception ; la technique habituelle est le crypto-shredding (effacement cryptographique), qui consiste à chiffrer les champs de chaque personne avec une clé propre à cette personne puis à détruire la clé, comme le décrit la section 7.5, ou à garder les données à caractère personnel hors des événements en ne stockant que des références. Les **journaux applicatifs** ne devraient pas contenir de données à caractère personnel inutiles, point sur lequel revient la section 8.8. Les **sauvegardes** ne peuvent généralement pas être modifiées ; l'Information Commissioner's Office (ICO) britannique admet que des données subsistent dans les sauvegardes jusqu'à leur écrasement selon un calendrier établi, à condition qu'elles soient mises hors d'usage entre-temps et que la personne soit clairement informée de ce qui se passera, et si une sauvegarde est un jour restaurée, les effacements doivent être réappliqués, ce qui exige de conserver la liste des identifiants effacés. Enfin, les **sous-traitants** doivent recevoir l'instruction de supprimer, puisqu'ils n'agissent que sur instruction du responsable du traitement (article 28), et les autres **destinataires** doivent être informés (article 19) : l'effacement doit donc atteindre la passerelle SMS, le service d'assistance et la plateforme marketing par leurs API.

L’effacement en éventail sur la plateforme de livraison de repas



Chaque stockage détenant des données du client doit agir, et dire qu’il l’a fait

Une demande, de nombreux stockages : chaque propriétaire applique sa propre règle (supprimer, anonymiser, crypto-shredding ou conserver avec un motif) et confirme l’achèvement.

EXEMPLE COMMENTÉ

Exemple commenté : concevoir « supprimer mon compte » pour la plateforme de livraison de repas

Un client à Londres demande la suppression de son compte. La conception doit effacer ce qui peut l'être, conserver ce que la loi exige avec un motif déclaré, terminer bien avant un mois, résister aux défaillances survenant en cours de route et produire des preuves. Les durées de conservation ci-dessous sont des choix illustratifs pour cet exemple, et non des exigences légales pour une entreprise particulière.

1. Vérifier et confirmer	La demande arrive dans l'application depuis une session connectée ; le service des comptes exige une nouvelle authentification, explique ce qui sera supprimé et ce qui sera conservé, et propose d'abord un export des données. Un court délai de réflexion (par exemple 14 jours) permet au client d'annuler ; les commandes et remboursements en cours doivent d'abord aboutir.
2. Enregistrer la demande	Le service des comptes écrit une entrée dans le registre des effacements avec un identifiant de demande, l'identifiant du client et la liste des stockages dont une réponse est attendue, tirée de la cartographie des données. Le registre ne stocke pas plus de données à caractère personnel que l'identifiant nécessaire au traitement de la demande.
3. Diffuser en éventail	Il publie `CustomerErasureRequested` via l'outbox dans un topic. Chaque consommateur est idempotent, car la livraison se fait au moins une fois, et traite l'événement même si le client a déjà été partiellement supprimé.
4. Appliquer les règles de chaque stockage	Commandes remplace le nom, le téléphone et l'adresse par une valeur de remplacement (tombstone) mais conserve les lignes de commande et les factures pendant la durée de conservation fiscale, avec l'obligation légale consignée comme motif. Paiements demande au prestataire de paiement de supprimer les jetons de carte stockés et conserve le journal des transactions. Notifications supprime les coordonnées et ajoute un e-mail haché à la liste d'exclusion. L'event store détruit la clé de données du client. L'entrepôt supprime les lignes par pseudonyme lors de son prochain traitement quotidien. Les caches sont purgés et l'index de recherche mis à jour.
5. Atteindre les sous-traitants	Des connecteurs appellent les API de suppression du SaaS d'assistance, de la passerelle SMS et de la plateforme marketing, en enregistrant chaque réponse.
6. Confirmer et relancer	Chaque consommateur écrit une confirmation dans le registre. Une tâche quotidienne déclenche une alerte pour tout stockage qui n'a pas confirmé sous sept jours, bien en deçà de la limite d'un mois.

7. Gérer les sauvegardes	Les sauvegardes de base de données sont conservées 35 jours et ne servent à rien d'autre qu'à la reprise après sinistre. Les identifiants des clients effacés sont conservés dans une liste, de sorte qu'une restauration réapplique tous les effacements avant la réouverture du système.
8. Informer le client	Un dernier message confirme l'achèvement, indique ce qui a été conservé, pourquoi et jusqu'à quand, et précise que les copies de sauvegarde expireront sous 35 jours.

La cartographie des données est l'artefact essentiel : un stockage qui en est absent est un stockage qui garde les données. Une fonction d'aptitude (fitness function) peut faire échouer le build si un nouveau service déclare un champ de données à caractère personnel sans enregistrer de gestionnaire d'effacement.

Conserver avec un motif n'est pas un échec de l'effacement. L'article 17, paragraphe 3, énumère des exceptions, dont le respect d'une obligation légale ; la conception conserve le minimum de données nécessaire à cette obligation et les supprime lorsque l'obligation prend fin.

Questions fréquentes

Chiffrer les données à caractère personnel suffit-il pour respecter le RGPD ?
Non. Le chiffrement est une mesure de sécurité importante, et une fuite de données chiffrées peut ne pas devoir être communiquée aux personnes, mais les principes, les fondements juridiques, les droits et les règles de conservation s'appliquent aux données chiffrées comme à toutes les autres.

Pouvons-nous tout conserver sous forme anonymisée au lieu de le supprimer ?
Seulement si les données sont vraiment anonymes, c'est-à-dire si personne ne peut raisonnablement réidentifier les personnes à partir d'elles. Hacher un identifiant ou retirer les noms produit généralement des données pseudonymisées, qui restent des données à caractère personnel pour quiconque détient les moyens de réidentifier.

Qui fixe les durées de conservation : les architectes ou les juristes ?
Les juristes et le métier fixent les durées et les motifs ; les architectes s'assurent que chaque stockage a une durée, qu'elle est appliquée automatiquement et que le système peut le prouver.

VÉRIFIEZ VOTRE COMPRÉHENSION

▼ Pourquoi une simple colonne booléenne `marketing_consent` est-elle une conception fragile ?

Elle ne peut pas montrer quand, comment et en réponse à quelle formulation le consentement a été donné, ce que l'article 7 impose au responsable du traitement de démontrer, et elle perd l'historique des retraits. Un enregistrement d'événements de consentement en ajout seul fournit cette preuve.

▼ L'équipe analytique reçoit des données de commandes indexées par un HMAC de l'identifiant client. S'agit-il de données à caractère personnel ?

Pour la plateforme, qui détient la clé, oui : ce sont des données pseudonymisées. Pour un destinataire qui n'a aucun moyen raisonnable de réidentification, elles peuvent ne pas l'être, selon l'arrêt CEPD/CRU (EDPS v SRB), mais les obligations du responsable du traitement demeurent.

▼ Pourquoi une sauvegarde restaurée menace-t-elle les effacements, et qu'est-ce qui l'empêche ?

La restauration ramène des données effacées après la prise de la sauvegarde. Conserver une liste des identifiants effacés et réappliquer les effacements avant d'utiliser le système restauré l'empêche.

▼ Qu'entend l'article 25, paragraphe 2, par protection des données par défaut ?

Sans aucune action de la personne, seules les données nécessaires à chaque finalité sont collectées, traitées, stockées et rendues accessibles ; par exemple, le partage facultatif du profil est désactivé par défaut.

▼ À vous : Le marketing veut conserver pour toujours la trace GPS complète de chaque livreur « pour l'analytique », avec le nom des livreurs. Appliquez la protection de la vie privée dès la conception : que contesteriez-vous et quelle conception proposeriez-vous ? Il s'agit ici d'architecture, et non de conseil juridique.

Remettez d'abord en cause la finalité : quelle question a besoin de ces données ? La limitation de la conservation et la minimisation plaident contre une conservation sans fin. Ne gardez les tracés précis que le temps nécessaire à l'affectation et aux litiges, puis supprimez-les automatiquement. Pour l'analytique, utilisez des données agrégées, comme les vitesses par zone, sans les noms. Remplacer les noms par des ID n'est qu'une pseudonymisation, et il s'agit toujours de données à caractère personnel. Restreignez l'accès par défaut, et réalisez une AIPD parallèlement au modèle de menaces si le suivi est susceptible d'engendrer un risque élevé.

SOURCES

- Regulation (EU) 2016/679 (GDPR), EUR-Lex: https://eur-lex.europa.eu/eli/reg/2016/679/oj
- EDPB Guidelines 4/2019 on Article 25 Data Protection by Design and by Default: https://www.edpb.europa.eu/our-work-tools/our-documents/guidelines/guidelines-42019-article-25-data-protection-design-and_en
- ICO: Right to erasure: https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/individual-rights/individual-rights/right-to-erasure/
- SOC 2 services and report types (Schellman): https://www.schellman.com/soc-2
- An Introduction to SOC 2 (Accounting Today): https://www.accountingtoday.com/news/an-introduction-to-soc-2
- AICPA & CIMA: SOC suite of services: https://www.aicpa-cima.com/resources/landing/system-and-organization-controls-soc-suite-of-services
- ISO/IEC 27001:2022/Amd 1:2024: https://www.iso.org/standard/88435.html
- KPMG: ISO/IEC 27001:2022 overview: https://assets.kpmg.com/content/dam/kpmg/cn/pdf/en/2023/02/iso-iec-27001-2022.pdf
- HHS: Summary of the HIPAA Security Rule: https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html
- HHS: HIPAA Security Rule NPRM fact sheet: https://www.hhs.gov/hipaa/for-professionals/security/hipaa-security-rule-nprm/factsheet/index.html
- California Attorney General: CCPA: https://oag.ca.gov/privacy/ccpa
- CPPA: Updated monetary thresholds in CCPA: https://www.cppa.ca.gov/regulations/cpi_adjustment.html
- CPPA: California finalizes regulations (23 September 2025): https://www.cppa.ca.gov/announcements/2025/20250923.html
- NIST OSCAL: https://pages.nist.gov/OSCAL/

SECTION 8.8 SUR 8.8 · INTERMÉDIAIRE · 32 MIN DE LECTURE · DERNIÈRE RELECTURE : NOVEMBRE 2026

Sécurité en exploitation : journalisation, détection et réponse

AVANT DE LIRE

Cette section s'appuie sur :

- 2.7 Sécurité, utilisabilité et autres qualités
- 8.2 Modélisation des menaces
- 8.7 Protection de la vie privée dès la conception

Termes déjà vus qu'elle utilise : Données à caractère personnel, Objectif de temps de reprise (RTO), Objectif de point de reprise (RPO), Outbox transactionnelle (transactional outbox), Nomenclature logicielle (SBOM)

L'IDÉE EN BREF

Une banque ne compte pas seulement sur des portes solides. Elle a aussi des caméras qui enregistrent qui est entré, des alarmes qui sonnent quand quelque chose d'inhabituel se produit, un plan qui dit au personnel quoi faire pendant un braquage, et des liquidités de réserve mises sous clé ailleurs au cas où la chambre forte principale serait perdue. Exploiter un logiciel de manière sécurisée fonctionne de la même façon. Les systèmes consignent soigneusement les actions importantes, surveillent ces traces pour y déceler des signes d'attaque, corrigent rapidement les faiblesses nouvellement découvertes, suivent un plan répété quand quelque chose tourne mal, conservent des sauvegardes hors de portée des attaquants, et informent à temps les régulateurs et les clients quand des données à caractère personnel sont touchées.

CE QU'IL FAUT SAVOIR

- Partez du principe que des violations se produiront : concevez pour détecter, contenir et rétablir, avec des scénarios de sécurité qui fixent des délais de réponse mesurables.
- Journalisez les événements de sécurité (authentification, échecs d'autorisation, actions d'administration et à haut risque) avec qui, quoi, quand et où ; ne journalisez jamais les mots de passe, tokens, clés, données de carte ni données personnelles inutiles ; protégez les journaux par un stockage séparé, à écriture unique.
- Une piste d'audit est un enregistrement complet, immuable et attribuable d'actions définies, émis par le service propriétaire au moment où l'action est validée.

- Un SIEM corrèle la télémétrie et déclenche des alertes selon des règles tirées du modèle de menaces et de MITRE ATT&CK ; les règles doivent être versionnées, testées et ajustées pour éviter la fatigue d'alerte.
- Priorisez les vulnérabilités selon l'exploitation avérée (KEV), la probabilité (EPSS), l'atteignabilité et l'exposition, et non selon le seul score de base CVSS, et convenez à l'avance des délais de correction.
- La publication NIST SP 800-61 révision 3 (avril 2025) organise la réponse aux incidents selon les fonctions du CSF 2.0 ; la résilience face aux rançongiciels dépend de sauvegardes immuables, protégées par des identifiants distincts, et de restaurations testées.
- En vertu de l'article 33 du RGPD, les violations de données à caractère personnel doivent être notifiées à l'autorité de contrôle dans les meilleurs délais et, si possible, 72 heures au plus tard après en avoir pris connaissance, et communiquées aux personnes concernées en vertu de l'article 34 lorsque le risque est élevé.

La sécurité ne s'arrête pas au déploiement

Les sections précédentes de ce chapitre visent à empêcher les attaques. Aucune prévention n'est parfaite ; les organisations matures conçoivent donc aussi en partant du principe qu'une violation se produira : elles veulent la remarquer rapidement, en limiter les dégâts, s'en remettre proprement et en tirer des leçons. Le chapitre 2 a décrit les tactiques de sécurité comme le fait de résister aux attaques, de les détecter, d'y réagir et de s'en remettre, et a montré comment écrire des scénarios de sécurité avec des réponses mesurables comme *un identifiant divulgué est détecté et révoqué en moins de 15 minutes*. Cette section porte sur l'architecture et les pratiques qui rendent de telles réponses atteignables.

Une grande partie s'appuie sur les fondations opérationnelles de l'observabilité et de la gestion des incidents du chapitre 9. La différence, c'est l'adversaire : un disque défaillant ne masque pas ses traces, ne supprime pas ses journaux et ne revient pas la semaine suivante par une porte qu'il a laissée ouverte.

La journalisation de sécurité : ce qu'il faut journaliser et ce qu'il ne faut jamais journaliser

Les journaux de sécurité servent à établir, après coup, qui a fait quoi, sur quelle ressource, quand, d'où et avec quel résultat. L'OWASP Logging Cheat Sheet est la référence standard des praticiens. Elle recommande d'enregistrer des événements comme les réussites et les échecs d'authentification, les échecs d'autorisation, les échecs de gestion de session, les échecs de validation des entrées, les erreurs applicatives, les fonctions à haut risque comme les actions d'administration et les exports de données, ainsi que l'utilisation des contrôles de sécurité eux-

mêmes. Chaque événement devrait indiquer quand (un horodatage synchronisé), où (application, service, hôte, environnement), qui (l'identité authentifiée, l'adresse source) et quoi (l'action, l'objet, le résultat et la raison).

Il est tout aussi important de savoir ce qui doit en rester exclu. Les journaux sont largement copiés, conservés longtemps et lus par de nombreuses personnes et de nombreux outils : c'est donc un endroit privilégié pour les fuites de secrets et de données à caractère personnel. Ne journalisez jamais les mots de passe, les identifiants de session ou les jetons d'accès (journalisez un hachage ou un identifiant de clé si vous devez corréler), les clés de chiffrement et les chaînes de connexion, les données de carte de paiement ni les catégories particulières de données à caractère personnel. Traitez les données de journal comme des données à caractère personnel lorsqu'elles identifient des personnes, et appliquez les règles de minimisation et de conservation de la section 8.7. Le masquage est mieux appliqué de façon centralisée dans le pipeline de journaux, en plus du code, car une seule instruction de débogage oubliée peut l'annuler.

À journaliser, à ne jamais journaliser (d'après l'OWASP Logging Cheat Sheet)

Enregistrer	Exclure, masquer ou hacher
Réussites et échecs d'authentification, défis MFA, réinitialisations de mot de passe	Mots de passe, même erronés (un mot de passe mal saisi est souvent presque correct)
Échecs d'autorisation (accès refusé), surtout lorsqu'ils se répètent	Identifiants de session, jetons d'accès, jetons de rafraîchissement, clés d'API (journaliser plutôt un hachage ou l'identifiant de la clé)
Échecs de validation des entrées et requêtes rejetées par la passerelle	Clés de chiffrement, chaînes de connexion aux bases de données et autres secrets
Actions d'administration : changements de rôle, changements de politique, création d'utilisateurs, changements de configuration	Numéros complets de carte de paiement, cryptogrammes visuels, coordonnées bancaires
Actions métier à haut risque : remboursements, versements, exports de données, suppression de compte	Données de santé, identifiants officiels et autres catégories particulières de données à caractère personnel
Erreurs et exceptions pertinentes pour la sécurité ; démarrage et arrêt des contrôles de sécurité	Données à caractère personnel au-delà de ce dont l'enquête a besoin, et tout ce qu'un utilisateur a refusé
Utilisation d'un accès d'urgence (bris de glace)	Code source, et données d'un niveau de classification supérieur à celui que le système de journalisation est habilité à détenir

```
{
  "ts": "2026-10-21T19:42:07.318Z",
  "event": "authz.denied",
  "service": "orders",
  "env": "prod",
  "actor": {"type": "user", "id": "cust_4411", "session": "sha256:7c1e..."},
  "client": {"ip": "203.0.113.24", "ua_family": "iOS app 7.3"},
  "action": "GET /orders/81723",
  "resource_owner": "cust_9902",
  "reason": "not_owner",
  "trace_id": "4bf92f3577b34da6a3ce929d0e0e4736"
}
```

Un événement de sécurité structuré émis par le service Commandes. Il montre qui a tenté quoi et pourquoi cela a été refusé, avec un identifiant de trace pour la corrélation, mais sans jeton ni nom de client.

Les journaux doivent aussi être protégés contre les attaquants, dont le premier geste après avoir obtenu un accès est souvent d'effacer les preuves. Envoyez rapidement les journaux de sécurité vers un compte ou un service distinct que les identifiants propres à l'application ne peuvent pas modifier ; stockez-les avec une conservation à écriture unique ; restreignez et journalisez l'accès à ces journaux ; et gardez les horloges synchronisées pour que les événements de systèmes différents concordent. Traitez le contenu des journaux comme une entrée non fiable : un attaquant capable d'insérer un saut de ligne dans un champ journalisé peut forger de fausses entrées (injection dans les journaux, ou log injection) ; encodez ou rejetez donc les caractères de contrôle, et souvenez-vous que Log4Shell était déclenché par la journalisation d'un texte contrôlé par l'attaquant.

Les pistes d'audit

Une piste d'audit est une forme de journal plus étroite et plus forte : un enregistrement complet, chronologique et dont toute altération est détectable, portant sur des actions définies et conservé pour établir les responsabilités. Les journaux de débogage sont échantillonnés, soumis à rotation et modifiés au gré des développeurs ; une piste d'audit doit être complète dans son périmètre (chaque remboursement, chaque changement de rôle, chaque accès au dossier d'assistance d'un client), immuable, attribuable à une identité authentifiée et conservée pendant une durée définie. Les réglementations et les normes en exigent souvent une ; PCI DSS, par exemple, exige que l'historique des journaux d'audit soit conservé au moins douze mois, les trois derniers mois étant immédiatement disponibles pour analyse.

Sur le plan architectural, concevez la piste d'audit comme un produit du domaine, et non comme un sous-produit de la journalisation. Émettez les événements d'audit depuis le service qui possède l'action, au moment où l'action est validée (l'outbox transactionnelle garantit que les deux se produisent ou qu'aucun ne se produit), vers un stockage en ajout seul dont l'accès

en écriture est restreint. La détection des altérations peut s'obtenir par chaînage de hachages, où chaque enregistrement inclut un hachage du précédent, ou en ancrant périodiquement ailleurs un condensé signé. Les systèmes en event sourcing en obtiennent une grande partie gratuitement, comme l'a montré la section 7.5. Les journaux d'audit du plan de contrôle des fournisseurs de cloud, qui enregistrent chaque appel d'API effectué sur le compte cloud, font partie du même tableau et doivent être activés dans chaque compte et chaque région.

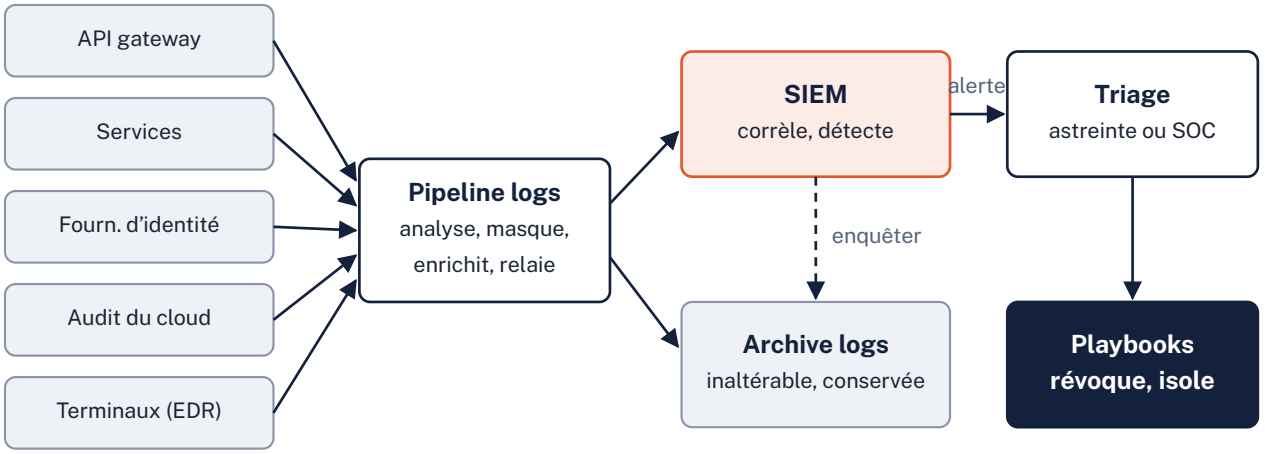
La détection : SIEM, règles et automatisation de la réponse

Un système de gestion des informations et des événements de sécurité (SIEM) collecte la télémétrie de sécurité de l'ensemble du parc, la normalise, corrèle les événements et déclenche des alertes quand des motifs correspondent à des règles de détection : de nombreuses connexions échouées suivies d'une réussite, un compte de service utilisé depuis un nouveau pays, un rôle qui reçoit des droits d'administrateur hors d'une fenêtre de changement, un gros export à trois heures du matin. Les sources typiques sont l'API gateway, les événements de sécurité applicatifs, le fournisseur d'identité, les journaux d'audit du cloud, les journaux d'audit de Kubernetes et les agents de détection et de réponse sur les terminaux (EDR) installés sur les ordinateurs portables du personnel et sur les serveurs.

Une bonne détection se construit ; elle ne s'achète pas. Tirez les règles du modèle de menaces et de MITRE ATT&CK, une base de connaissances publique des tactiques et techniques d'attaquants réels, tenue par MITRE depuis 2015 ; écrivez-les sous forme de code dans un format partagé comme Sigma, versionnez-les, et testez-les en simulant l'attaque qu'elles doivent détecter. Mesurez la qualité des alertes : une règle qui se déclenche cinquante fois par jour et se révèle toujours bénigne apprend aux gens à l'ignorer, et c'est par la fatigue d'alerte que de vraies attaques passent inaperçues. Des pièges peu coûteux et très parlants aident : les honeytokens (jetons leurres), comme une fausse clé d'accès ou une fausse fiche client qu'aucun processus légitime ne touche jamais, ne produisent presque aucune fausse alerte. Les outils d'orchestration, d'automatisation et de réponse de sécurité (SOAR) exécutent ensuite des playbooks pour les alertes bien comprises, comme désactiver un utilisateur, révoquer une clé ou isoler un conteneur, afin que les humains se consacrent aux décisions qui demandent du jugement.

L'ingestion dans un SIEM est souvent facturée au volume, ce qui incite les équipes à abandonner des journaux. Une conception à plusieurs niveaux aide : les événements de sécurité à forte valeur vont au SIEM pour une détection en temps réel ; les journaux volumineux vont vers un stockage à écriture unique moins cher, où ils peuvent être fouillés pendant une enquête.

Un pipeline de détection de sécurité



Règles de détection = code : versionnées, testées sur des attaques types, réglées pour que chaque alerte mérite un humain

La télémétrie est collectée et masquée une seule fois, puis sert à la fois à la détection en temps réel et, depuis une archive à écriture unique, aux enquêtes.

La gestion des vulnérabilités

De nouvelles vulnérabilités sont publiées chaque jour dans les logiciels que vous exécutez. Un identifiant CVE (Common Vulnerabilities and Exposures), comme CVE-2021-44228, donne à chaque vulnérabilité publiquement connue un nom unique ; le programme, lancé par MITRE en 1999, attribue les identifiants par l'intermédiaire de centaines de CVE Numbering Authorities, parmi lesquelles de grands éditeurs et des fondations open source. La National Vulnerability Database américaine ajoute des analyses comme les identifiants des produits concernés et des scores, mais un arriéré dans cet enrichissement à partir de 2024, et une brève alerte en avril 2025 lorsque le contrat finançant le programme CVE a failli expirer, ont montré le risque de dépendre d'une source unique ; la plupart des organisations combinent désormais plusieurs sources, dont la base d'avis de sécurité de GitHub et la base open source OSV.

CVSS (le Common Vulnerability Scoring System, maintenu par le FIRST ; la version 4.0 a été publiée en novembre 2023) évalue la gravité de 0 à 10, avec les niveaux faible (0,1 à 3,9), moyen (4,0 à 6,9), élevé (7,0 à 8,9) et critique (9,0 à 10,0). Le score de base publié décrit la vulnérabilité en général ; les métriques de menace et d'environnement de la version 4.0 l'ajustent en fonction de l'activité d'exploitation et de votre propre environnement. Un score de base ne mesure pas votre risque. Deux signaux améliorent considérablement la priorisation : l'Exploit Prediction Scoring System (EPSS, également du FIRST), qui estime la probabilité qu'une vulnérabilité soit exploitée dans les 30 prochains jours, et le catalogue Known Exploited Vulnerabilities (KEV) de la Cybersecurity and Infrastructure Security Agency américaine, qui recense les vulnérabilités dont l'exploitation réelle est avérée. Combinez-les avec l'atteignabilité (la fonction vulnérable

est-elle utilisée ?) et l'exposition (le service est-il exposé à Internet ?), et la plupart des centaines de constats d'un scanner rejoignent la file courante, tandis qu'une poignée exige une action dès aujourd'hui.

Convenez à l'avance des délais de correction, pour que chaque constat ne devienne pas une négociation, et mesurez le délai de correction. Les chiffres ci-dessous sont des illustrations typiques, pas une norme ; les environnements réglementés peuvent imposer les leurs : PCI DSS, par exemple, exige que les correctifs de sécurité critiques soient installés dans le mois qui suit leur publication.

Exemples de délais de correction

Constat	Délai visé
Exploitation avérée (par exemple dans le catalogue KEV de la CISA), exposé à Internet	Atténuer sous 24 à 72 heures ; corriger dès qu'un correctif existe
Gravité critique, atteignable, aucune exploitation connue	Sous 7 à 14 jours
Gravité élevée	Sous 30 jours
Gravité moyenne	Sous 90 jours, ou à la prochaine mise à jour courante
Gravité faible, ou non atteignable	Mises à jour courantes ; consigner la raison en cas de report

GUIDE DE DÉCISION

Guide de décision : trier une vulnérabilité nouvellement publiée

Utilisez-le quand un scanner, un avis de sécurité ou un article de presse signale une vulnérabilité dans un élément que vous pourriez utiliser.

Q1 **Un artefact déployé contient-il le composant vulnérable dans la version concernée (vérifiez le stockage des SBOM) ?**

- **Oui** Passez à la question suivante, et consignez chaque service concerné et son propriétaire.
- **Non** Consignez « non concerné » avec les preuves ; continuez à surveiller l'apparition de nouvelles versions concernées.

Q2 **Est-elle connue pour être exploitée dans la nature (par exemple inscrite au catalogue KEV de la CISA), ou un code d'exploitation fonctionnel est-il public ?**

- **Oui** Voie d'urgence : atténuez en quelques heures à quelques jours (correctif, désactivation de la fonctionnalité ou blocage au niveau de la passerelle ou du WAF), puis corrigez proprement.
- **Non** Passez à la question suivante.

Q3 **Le code vulnérable est-il atteignable dans votre usage, et le service est-il exposé à Internet ou traite-t-il des données sensibles ?**

- **Oui** Corrigez dans le délai du SLA correspondant à sa gravité, ajusté avec les métriques d'environnement de CVSS et l'EPSS.
- **Non** Passez à la question suivante.

Q4 **Existe-t-il une mesure compensatoire qui bloque entièrement l'exploitation (fonctionnalité désactivée, aucun chemin réseau) ?**

- **Oui** Documentez-la (une déclaration VEX convient à cela) et corrigez lors du cycle de mise à jour courant.
- **Non** Corrigez lors du cycle courant et réexaminez si une exploitation est signalée.

SI VOUS DÉBUTEZ

Pas à pas : le score le plus élevé ne passe pas toujours en premier

L'idée qui demande le plus d'attention ici est de classer les vulnérabilités selon votre propre risque plutôt que selon leur score. La section 8.2 a présenté CVSS : son score de base décrit une faille en général, pas le risque qu'elle représente pour vous. Suivez deux constats qui arrivent le même matin.

- Le constat A est une faille critique, de score de base 9,1, dans une bibliothèque intégrée au service Commandes. Le constat B est une faille élevée, de score de base 7,5, dans un logiciel qu'exécute l'API gateway. Triés par score, A passe en premier.
- La première vérification consiste à savoir si le composant est réellement déployé. Le stockage des SBOM montre que les deux le sont, et l'équipe consigne chaque service concerné et son propriétaire.
- Vient ensuite l'exploitation avérée. B figure dans le catalogue Known Exploited Vulnerabilities tenu par la CISA, une agence gouvernementale américaine : des éléments montrent que des attaquants l'utilisent. Rien n'indique que quiconque exploite A.
- Puis l'atteignabilité et l'exposition. La passerelle est exposée à Internet, donc les attaquants peuvent atteindre B directement. A se trouve dans une partie de la bibliothèque que le service Commandes n'appelle jamais.
- B prend donc la voie d'urgence : atténuer en quelques heures à quelques jours, par exemple en désactivant la fonctionnalité concernée ou en bloquant l'attaque en périphérie, puis corriger proprement. A est consigné comme non atteignable, avec les preuves, et corrigé lors du cycle de mise à jour courant, à réexaminer si une exploitation est signalée.
- Pour les nombreux constats qui ne sont ni répertoriés ni clairement inatteignables, l'EPSS aide à les classer : il estime la probabilité que chacun soit exploité dans les 30 prochains jours.
- **La leçon :** classez les vulnérabilités selon l'exploitation avérée, la probabilité, l'atteignabilité et l'exposition, et non selon le seul score de base, et convenez à l'avance des délais de correction pour que chaque constat ne donne pas lieu à une nouvelle négociation.

Tests d'intrusion, bug bounties et divulgation

Un test d'intrusion est une tentative autorisée et limitée dans le temps, menée par des testeurs compétents, de pénétrer dans un système, idéalement en connaissant son architecture et avec un périmètre convenu par écrit. Il trouve des classes de problèmes que les scanners manquent, comme les défauts d'autorisation au niveau des objets et l'abus de la logique métier (utiliser

deux fois le même bon de réduction grâce à une situation de compétition, ou race condition). Commandez-en un avant un lancement majeur, après un changement architectural important et périodiquement ; un exercice de red team va plus loin, en testant la détection et la réponse par la simulation d'un véritable adversaire pendant des semaines.

Des chercheurs extérieurs trouveront eux aussi des problèmes, et ils ont besoin d'un moyen sûr de les signaler. Une politique de divulgation des vulnérabilités indique comment signaler, quels tests sont autorisés et que la recherche de bonne foi ne fera pas l'objet de menaces juridiques ; un fichier `security.txt` placé à l'adresse `/.well-known/security.txt` (RFC 9116, 2022) indique aux chercheurs où envoyer leurs signalements. Un bug bounty (programme de prime aux bogues) ajoute une rémunération pour les constats valides, souvent via une plateforme qui trie les soumissions. Les normes ISO/IEC 29147 (divulgation) et ISO/IEC 30111 (traitement) décrivent les processus, et le règlement sur la cyberrésilience impose aux fabricants des produits relevant de son champ d'application d'avoir une politique de divulgation coordonnée des vulnérabilités. Un bug bounty ne remplace pas le travail de sécurité interne : lancé sur un système non testé, il tend à produire un afflux de signalements que l'équipe ne peut pas absorber.

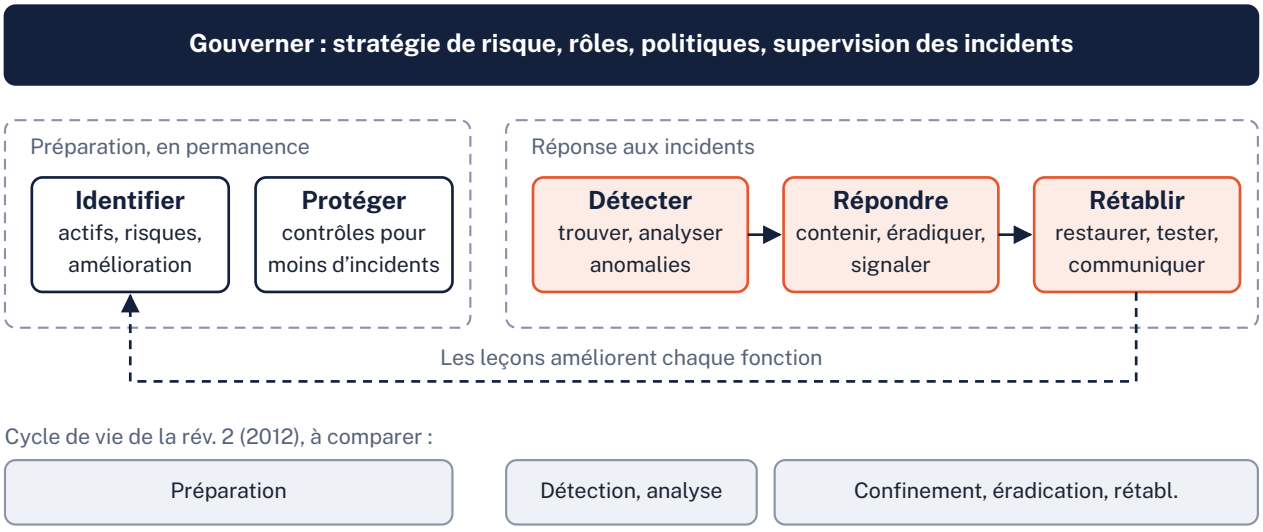
Réponse aux incidents

La réponse aux incidents est le traitement organisé d'un incident de sécurité, de sa détection jusqu'au rétablissement et aux enseignements. Pendant plus de dix ans, la référence standard a été la publication NIST SP 800-61 révision 2 (2012), avec son cycle de vie en quatre phases : préparation ; détection et analyse ; confinement, éradication et rétablissement ; et activité post-incident. La révision 3, publiée en avril 2025 sous le titre *Incident Response Recommendations and Considerations for Cybersecurity Risk Management: A CSF 2.0 Community Profile*, remplace ce cycle de vie par les six fonctions du NIST Cybersecurity Framework 2.0. Gouverner, Identifier et Protéger constituent une préparation continue qui rend les incidents moins probables et la réponse plus efficace ; Détecter, Répondre et Rétablir constituent la réponse elle-même ; et les enseignements tirés alimentent l'amélioration de toutes ces fonctions au lieu de constituer une phase finale.

En pratique, une bonne réponse exige des rôles nommés (un coordinateur d'incident (incident commander) qui coordonne et décide, des responsables techniques qui enquêtent et agissent, un responsable de la communication, un scribe qui tient la chronologie, et un accès rapide aux conseillers juridiques et au délégué à la protection des données), des niveaux de gravité qui déterminent qui est réveillé, et un canal hors bande, car un attaquant présent dans votre messagerie ou votre système de chat peut lire la réponse. Le confinement est un compromis : couper rapidement l'accès à un attaquant limite les dégâts mais peut détruire des preuves et l'alerter ; préservez donc les preuves (snapshots, journaux, captures mémoire) avant de

reconstruire, et assurez-vous d'avoir trouvé toutes ses positions d'appui avant d'annoncer qu'il est parti. Ensuite, une revue sans reproche examine comment le système et ses processus ont permis l'incident, et produit des actions architecturales.

La réponse aux incidents dans la publication NIST SP 800-61 révision 3



La 4e phase de la rév. 2, l'activité post-incident, devient l'amélioration continue de la rév. 3

La révision 3 projette la réponse aux incidents sur les fonctions du CSF 2.0 au lieu d'un cycle de vie en quatre phases.

La résilience face aux rançongiciels

Un rançongiciel chiffre les systèmes d'une organisation et exige un paiement contre la clé ; la plupart des groupes volent désormais aussi les données au préalable et menacent de les publier. Les défenses architecturales sont celles de tout ce chapitre (MFA, moindre privilège, segmentation, correctifs, détection), mais le rétablissement dépend avant tout d'une chose : des sauvegardes que l'attaquant ne peut pas atteindre. Les attaquants dotés de droits d'administrateur cherchent d'abord les sauvegardes et les suppriment ou les chiffrent.

Conservez au moins une copie des données critiques immuable ou hors ligne : un stockage objet avec des verrous de conservation à écriture unique que même un administrateur ne peut pas raccourcir, ou des supports physiquement déconnectés du réseau. Placez les sauvegardes dans un compte ou un tenant distinct, avec des identifiants et une MFA distincts, afin que la compromission des identités de production ne les atteigne pas. La règle traditionnelle 3-2-1 (trois copies, sur deux types de supports, dont une hors site) est une base raisonnable. Surtout, testez régulièrement les restaurations par rapport au RTO et au RPO convenus avec le métier, y compris l'ordre de reprise : les services d'identité et le DNS passent en premier, car rien d'autre ne peut être reconstruit sans eux. En 2017, la compagnie maritime Maersk a reconstruit son

réseau mondial après que le logiciel malveillant NotPetya eut détruit ses contrôleurs de domaine Windows ; la reprise a reposé sur le seul contrôleur de domaine qui avait survécu, au Ghana, parce qu'une coupure de courant l'avait mis hors ligne pendant l'attaque.

Violations de données à caractère personnel et délais de notification

Une violation de données à caractère personnel est, au sens de l'article 4, point 12, du RGPD, une violation de la sécurité entraînant, de manière accidentelle ou illicite, la destruction, la perte, l'altération, la divulgation non autorisée de données à caractère personnel, ou l'accès non autorisé à de telles données. La notion est plus large que le vol : une attaque par rançongiciel qui rend les données indisponibles, ou un e-mail envoyé aux mauvais destinataires, peut en relever. En vertu de l'article 33, le responsable du traitement doit notifier la violation à l'autorité de contrôle dans les meilleurs délais et, si possible, 72 heures au plus tard après en avoir pris connaissance, à moins qu'elle ne soit pas susceptible d'engendrer un risque pour les droits et libertés des personnes ; une notification plus tardive doit être accompagnée des motifs du retard. En vertu de l'article 34, le responsable du traitement doit aussi informer les personnes concernées dans les meilleurs délais si la violation est susceptible d'engendrer un risque élevé pour elles. L'article 33, paragraphe 5, exige que toute violation, notifiée ou non, soit documentée. Le RGPD britannique prévoit les mêmes règles, les notifications étant adressées à l'Information Commissioner's Office.

D'autres régimes peuvent s'appliquer en parallèle, avec leurs propres délais, destinataires et seuils, et les contrats ajoutent souvent des obligations envers les clients et les partenaires. Le tableau résume les principaux délais de l'UE et du Royaume-Uni. La proposition d'omnibus numérique de la Commission européenne de novembre 2025 porterait le délai du RGPD à 96 heures et limiterait la notification à l'autorité aux violations à haut risque, mais, à la date de rédaction, elle n'a pas été adoptée et la règle des 72 heures s'applique.

La leçon d'architecture est qu'un délai compté en heures ne peut être tenu que si les systèmes peuvent établir, rapidement et preuves à l'appui, quelles données ont été touchées, à qui elles appartenaient et si elles ont réellement été consultées. Cela exige la cartographie des données de la section 8.7, des journaux d'accès aux données sur les stockages de données à caractère personnel, et un chiffrement susceptible de rendre incompréhensible une copie divulguée. Comme pour le reste de ce chapitre, il s'agit d'informations sur la façon dont le droit influe sur les systèmes, et non d'un conseil juridique ; la décision de notifier appartient au responsable du traitement, conseillé par son DPD et ses avocats.

Délais de notification couramment applicables

Règle	Qui informe qui	Délai
RGPD / RGPD britannique, art. 33	Responsable du traitement à l'autorité de contrôle (au Royaume-Uni, l'ICO)	Dans les meilleurs délais et, si possible, 72 heures au plus tard après en avoir pris connaissance, sauf si la violation n'est pas susceptible d'engendrer un risque pour les personnes. Les informations peuvent être communiquées de manière échelonnée.
RGPD / RGPD britannique, art. 33, paragraphe 2	Sous-traitant au responsable du traitement	Dans les meilleurs délais après en avoir pris connaissance.
RGPD / RGPD britannique, art. 34	Responsable du traitement aux personnes concernées	Dans les meilleurs délais lorsque la violation est susceptible d'engendrer un risque élevé, sauf, par exemple, si les données étaient protégées par un chiffrement qui les rend incompréhensibles.
Directive SRI 2 (NIS2), directive (UE) 2022/2555	Entités essentielles et importantes au CSIRT ou à l'autorité	Alerte précoce dans les 24 heures après avoir pris connaissance d'un incident important, notification d'incident dans les 72 heures, rapport final dans un délai d'un mois.
Règlement sur la cyberrésilience, art. 14	Fabricants, via la plateforme unique de déclaration de l'ENISA	Pour les vulnérabilités activement exploitées et les incidents graves : alerte précoce dans les 24 heures, notification dans les 72 heures (applicable à partir du 11 septembre 2026).

Les runbooks

Un runbook (procédure d'exploitation) est une procédure écrite et testée pour une situation précise : révoquer et renouveler un identifiant divulgué, isoler un conteneur compromis, restaurer la base de données des commandes à partir d'une sauvegarde immuable, réagir à une suspicion de violation de données à caractère personnel. Un bon runbook indique son déclencheur, les personnes à impliquer, les étapes et commandes exactes, comment vérifier que chaque étape a fonctionné, comment revenir en arrière et quelles preuves conserver. Les runbooks sont du code dans l'esprit : conservez-les sous gestion de versions à côté des systèmes qu'ils décrivent, automatisez les parties mécaniques, et répétez-les lors de game days et d'exercices sur table, car une procédure qui n'a jamais été exécutée échoue généralement à l'étape trois. Chaque règle de détection devrait renvoyer au runbook de son alerte.

Exemple commenté : les 24 premières heures d'un incident de fuite d'identifiants sur la plateforme de livraison de repas

Durée	Ce qui se passe	Pourquoi c'est important
09:12	Un ingénieur pousse dans un dépôt public un commit qui contient une clé d'accès cloud à longue durée de vie pour le rôle de déploiement en production.	La cause habituelle : un script de débogage, pas de la malveillance. Des scanners automatisés surveillent les commits publics : il faut donc considérer immédiatement la clé comme exposée.
09:14	L'alerte de détection de secrets parvient à l'ingénieur sécurité d'astreinte ; à 09:16, le SIEM montre des appels d'API effectués avec la clé depuis une adresse inconnue.	Une détection en quelques minutes suppose que la détection de secrets et les journaux d'audit du cloud alimentent le SIEM avant l'incident.
09:20	Incident déclaré au niveau de gravité le plus élevé. Un coordinateur d'incident, un scribe et un responsable technique sont désignés ; un canal dédié est ouvert.	Des rôles clairs évitent que cinq personnes fassent la même chose et que personne ne fasse la chose urgente.
09:25	Confinement : la clé est désactivée, les sessions émises à partir d'elle sont révoquées et une politique de refus total est attachée au rôle jusqu'à sa reconstruction.	Désactiver plutôt que supprimer conserve l'identifiant de la clé pour les recherches dans les journaux. Retirer le commit n'est pas un confinement : la clé a déjà été copiée.
09:45	Périmètre : l'équipe interroge le journal d'audit du cloud pour retrouver chaque appel effectué avec l'identifiant de la clé, depuis le moment de sa création, et pas seulement depuis le push.	La clé a pu fuiter plus tôt par une autre voie ; le journal indique ce qui a été fait, d'où et quand.
10:30	Constatations : l'attaquant a listé les buckets de stockage, lancé 40 grosses instances de calcul dans une région inutilisée et créé un nouvel utilisateur doté de sa propre clé d'accès.	Les attaquants établissent d'abord leur persistance ; ne nettoyer que la clé d'origine les laisserait à l'intérieur.
11:00	Éradication : instances arrêtées, nouvel utilisateur et nouvelle clé supprimés, politiques récemment modifiées examinées et rétablies ; snapshots pris au préalable pour les preuves.	Préservez les preuves avant de les détruire ; elles sont nécessaires pour la revue, pour les assureurs et éventuellement pour les régulateurs.

Durée	Ce qui se passe	Pourquoi c’est important
11:45	Question des données : les journaux d’accès au niveau des objets montrent des téléchargements réussis depuis le bucket contenant les exports clients. Le DPD et le conseiller juridique rejoignent l’équipe.	Sans journaux au niveau des objets, l’équipe ne pourrait pas prouver que les données n’ont pas été lues et devrait supposer qu’elles l’ont été.
12:30	Le DPD conclut à une violation de données à caractère personnel susceptible d’engendrer un risque ; le délai de 72 heures est considéré comme courant depuis le moment où l’organisation en a pris connaissance. Une première notification est rédigée.	L’article 33 permet de communiquer les informations de manière échelonnée ; une notification tardive doit être motivée. L’obligation d’informer les personnes dépend du niveau de risque.
15:00	Rétablissement : rôle de déploiement reconstruit avec des identifiants à courte durée de vie via une fédération OIDC depuis la CI ; alarmes de facturation et restrictions de région ajoutées ; le fournisseur est contacté au sujet des frais frauduleux.	Corrigez la classe de problème, pas seulement le cas : sans clés à longue durée de vie, il n’y a rien à divulguer.
18:00	Protection des push contre les secrets activée pour chaque dépôt ; tous les autres secrets lisibles depuis la machine du même ingénieur renouvelés par précaution.	Supposez que des identifiants liés peuvent aussi être exposés jusqu’à preuve du contraire.
Lendemain 09:00	Point de situation à la direction ; notification à l’autorité de contrôle soumise dans les 72 heures avec les éléments connus ; revue sans reproche programmée dans les cinq jours ouvrés.	La revue demande pourquoi une clé à longue durée de vie existait et a pu être commitée, pas qui l’a commitée.

La chronologie montre pourquoi l’exploitation fait partie de l’architecture. Chaque étape qui s’est bien passée dépendait d’une décision de conception prise des mois plus tôt : la détection de secrets et les journaux d’audit alimentant le SIEM, les journaux d’accès au niveau des objets sur le bucket d’exports, une cartographie des données qui a indiqué au DPD à qui appartenaient les données qu’il contenait, et une conception des identités qui a permis de reconstruire le rôle de déploiement sans clés à longue durée de vie. Chaque étape qui s’est mal passée (l’existence même d’une clé à longue durée de vie, un bucket d’exports clients dont plus personne n’avait besoin) relevait aussi d’un choix architectural.

Liste de contrôle : le système est-il prêt à être attaqué ?

- [ ] **Les événements de sécurité sont journalisés, structurés et centralisés**
  Authentification, échecs d'autorisation, actions d'administration et à haut risque, avec qui, quoi, quand, où et un identifiant de trace ; aucun secret ni donnée à caractère personnel inutile.
- [ ] **Les journaux sont protégés et conservés**
  Compte ou stockage distinct avec conservation à écriture unique, accès restreint, horloges synchronisées et durée de conservation définie.
- [ ] **Les journaux d'audit du cloud et d'accès aux données sont activés**
  Y compris l'accès au niveau des objets pour les stockages de données à caractère personnel ou de paiement, afin que le périmètre puisse être prouvé.
- [ ] **Des détections existent pour vos principales menaces**
  Règles tirées du modèle de menaces, testées par des attaques simulées, avec un propriétaire et un runbook pour chaque alerte.
- [ ] **Les SLA de correction des vulnérabilités sont convenus et mesurés**
  Priorisation selon l'exploitation, l'atteignabilité et l'exposition ; le stockage des SBOM répond à « où est-ce ? » en quelques minutes.
- [ ] **Une voie de divulgation existe**
  Une politique de divulgation des vulnérabilités et un fichier security.txt ; des tests d'intrusion avant les lancements majeurs et après les changements importants.
- [ ] **Le plan de réponse aux incidents est répété**
  Rôles, niveaux de gravité, canal de communication hors bande et contacts pour le juridique, le DPD, les assureurs et les fournisseurs ; exercices au moins une fois par an.
- [ ] **Les sauvegardes résistent à un attaquant doté de droits d'administrateur**
  Copies immuables ou hors ligne dans un compte distinct avec des identifiants distincts ; restaurations testées par rapport au RTO et au RPO.
- [ ] **Les obligations de notification sont recensées**
  Quels régulateurs, clients et partenaires doivent être informés, par qui et dans quels délais, avec des modèles préparés.

VÉRIFIEZ VOTRE COMPRÉHENSION

▼ Pourquoi un journal devrait-il enregistrer un hachage ou un identifiant de clé plutôt que le jeton d'accès lui-même ?

Les journaux sont largement copiés et lus : un jeton journalisé peut donc être volé et rejoué. Un hachage ou un identifiant de clé permet tout de même aux enquêteurs de corréler les événements sans exposer d'identifiant de connexion utilisable.

▼ Un scanner signale une vulnérabilité de score CVSS 9,8 dans une bibliothèque dont vous n'appelez jamais la fonction vulnérable, et une autre de score CVSS 7,5, inscrite au catalogue KEV, dans votre passerelle exposée à Internet. Laquelle traiter en premier ?

La vulnérabilité de la passerelle inscrite au catalogue KEV : son exploitation est avérée et elle est exposée. La vulnérabilité à 9,8 non atteignable peut suivre le cycle de correctifs courant, avec le raisonnement consigné.

▼ Pourquoi les sauvegardes doivent-elles se trouver dans un compte distinct avec des identifiants distincts ?

Les attaquants qui obtiennent des droits d'administrateur en production cherchent les sauvegardes et les détruisent ; la séparation fait que la compromission des identités de production n'atteint pas la copie de reprise.

▼ Quand le délai de 72 heures du RGPD commence-t-il à courir ?

Lorsque le responsable du traitement prend connaissance de la violation de données à caractère personnel, c'est-à-dire lorsqu'il a un degré raisonnable de certitude qu'un incident de sécurité a compromis des données à caractère personnel, et non lorsque l'enquête est terminée.

▼ À vous : La vulnérabilité A a un score CVSS de 9,8, dans une bibliothèque sur un serveur de traitement par lots interne, et sa fonction vulnérable n'est jamais appelée. La vulnérabilité B a un score CVSS de 7,5, sur votre service de connexion public, et elle est connue pour être activement exploitée. Laquelle d'abord ?

B d'abord. Le score de base CVSS seul est un mauvais guide. B est accessible depuis internet et figure sur la liste des vulnérabilités activement exploitées : une attaque est donc probable et pourrait toucher un service critique. Corrigez-le ou protégez-le maintenant, et cherchez dans les journaux des signes d'exploitation passée. Le score de A est plus élevé, mais il est interne et son code n'est pas atteignable : planifiez-le donc dans le délai convenu. Consignez le raisonnement, puis revérifiez si l'exposition change.

SOURCES

- OWASP Logging Cheat Sheet: https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html
- NIST SP 800-61 Rev. 3: Incident Response Recommendations and Considerations (2025): https://csrc.nist.gov/pubs/sp/800/61/r3/final
- FIRST: Common Vulnerability Scoring System v4.0: https://www.first.org/cvss/v4-0/
- FIRST: Exploit Prediction Scoring System (EPSS): https://www.first.org/epss/
- RFC 9116: A File Format to Aid in Security Vulnerability Disclosure: https://www.rfc-editor.org/rfc/rfc9116

Lectures complémentaires

- *Threat Modeling: Designing for Security*, Adam Shostack (2014), Wiley. Le livre pratique de référence sur la modélisation des menaces : les quatre questions, les diagrammes de flux de données, STRIDE, les arbres d'attaque et la façon d'intégrer la modélisation des menaces au quotidien de l'ingénierie.
- *Security Engineering: A Guide to Building Dependable Distributed Systems (3rd edition)*, Ross Anderson (2020), Wiley. Un panorama vaste et très documenté de la façon dont les systèmes réels échouent, des protocoles et du contrôle d'accès à la banque, aux paiements et à l'économie de la sécurité.
- *Designing Secure Software: A Guide for Developers*, Loren Kohnfelder (2021), No Starch Press. Les principes de sécurité dès la conception, la modélisation des menaces et les revues de conception sécurisée, par l'un des co-inventeurs de STRIDE, à l'intention des développeurs et des architectes en activité.
- *API Security in Action*, Neil Madden (2020), Manning. L'authentification, les tokens, OAuth 2.0, la sécurité fondée sur les capacités et la sécurisation des appels de service à service, avec du code fonctionnel.
- *OAuth 2 in Action*, Justin Richer and Antonio Sanso (2017), Manning. Construit pas à pas des clients, des serveurs d'autorisation et des serveurs de ressources, ce qui rend concrets les flux et leurs attaques.
- *The Protection of Information in Computer Systems*, Jerome H. Saltzer and Michael D. Schroeder (1975), Proceedings of the IEEE, vol. 63, no. 9. L'origine du moindre privilège, des valeurs par défaut sûres, de la médiation complète et des autres principes de conception encore cités aujourd'hui.
  https://doi.org/10.1109/PROC.1975.9939
- *SP 800-207: Zero Trust Architecture*, Scott Rose, Oliver Borchert, Stu Mitchell and Sean Connelly (NIST) (2020), National Institute of Standards and Technology. La définition de référence du zero trust, de ses principes fondamentaux et de ses composants logiques (moteur de politique, administrateur de politique et point d'application).
  https://csrc.nist.gov/pubs/sp/800/207/final
- *Regulation (EU) 2016/679 (General Data Protection Regulation)*, European Parliament and Council of the European Union (2016), Official Journal of the European Union (EUR-Lex). Le texte principal sur lequel s'appuie la section sur la protection de la vie privée ; les articles 5, 25 et 32 à 35 sont les plus importants pour les architectes.
  https://eur-lex.europa.eu/eli/reg/2016/679/oj

Glossaire

Analyse d’impact relative à la protection des données (AIPD)
Analyse exigée par l’article 35 du RGPD avant tout traitement susceptible d’engendrer un risque élevé pour les personnes, qui décrit le traitement, sa nécessité, les risques et les mesures envisagées pour y faire face.

Analyse de la composition logicielle (SCA)
Outils et pratiques qui identifient les composants open source d’un logiciel et les confrontent à des bases de données de vulnérabilités et de licences connues.

Anonymisation
Traitement des données de sorte qu’elles ne se rapportent plus à une personne identifiable par aucun moyen raisonnablement susceptible d’être utilisé ; les données réellement anonymes sortent du champ du RGPD.

Arbre d’attaque
Arbre dont la racine est l’objectif d’un attaquant et dont les enfants sont les moyens alternatifs (OU) ou combinés (ET) de l’atteindre, popularisé par Bruce Schneier en 1999.

Attaque de la chaîne d’approvisionnement logicielle
Attaque qui compromet quelque chose qu’une organisation consomme, comme une dépendance, un outil de build, une mise à jour ou un registre, afin d’atteindre cette organisation et d’autres en aval.

Authentification
Vérification qu’un utilisateur ou un système est bien celui ou ce qu’il prétend être, par exemple au moyen d’un mot de passe, d’une passkey ou d’un certificat.

Authentification multifacteur (MFA)
Authentification qui exige deux types de facteurs différents ou plus : quelque chose que vous savez, quelque chose que vous possédez et quelque chose que vous êtes.

Autorisation
Décision indiquant si un utilisateur ou un système authentifié a le droit d’effectuer une action particulière sur une ressource particulière.

Autorité de certification (CA)
Organisation ou système qui émet des certificats numériques en les signant, garantissant qu’une clé publique appartient à un serveur, une personne ou une charge de travail nommés.

Bug bounty (programme de prime aux bogues)
Programme qui rémunère des chercheurs externes pour le signalement de vulnérabilités de sécurité avérées selon un processus convenu.

Build reproductible (reproducible build)
Build qui produit une sortie identique bit pour bit à partir du même code source et des mêmes entrées, afin que chacun puisse reconstruire un artefact publié et vérifier qu'il n'a pas été altéré.

Calendrier de conservation
Liste documentée indiquant combien de temps chaque catégorie de données est conservée, pourquoi, et ce qu'il en advient à la fin de cette période.

Chiffrement d'enveloppe (envelope encryption)
Chiffrement des données avec une clé de données, puis chiffrement (encapsulation) de cette clé de données avec une clé de chiffrement de clés détenue dans un KMS, de sorte que les données volumineuses ne transitent jamais par le KMS et que les clés peuvent être renouvelées à peu de frais.

Chiffrement symétrique
Chiffrement dans lequel la même clé secrète sert à chiffrer et à déchiffrer, comme AES ; rapide, il est donc utilisé pour les gros volumes de données.

Contrôle d'accès basé sur les attributs (ABAC)
Décision d'accès prise en évaluant une politique portant sur des attributs de l'utilisateur, de la ressource, de l'action et du contexte, comme le pays, la classification des données ou l'heure de la journée.

Contrôle d'accès basé sur les relations (ReBAC)
Décider des accès à partir des relations entre utilisateurs et objets (propriétaire de, membre de, travaille chez), stockées sous forme de graphe ; Zanzibar de Google (2019) en est l'exemple le plus connu.

Contrôle d'accès basé sur les rôles (RBAC)
Attribuer des permissions à des rôles et des rôles à des utilisateurs, de sorte que les accès se gèrent en affectant les personnes à des rôles comme agent du support ou gérant de restaurant.

Cryptographie asymétrique (à clé publique)
Cryptographie fondée sur une paire de clés : une clé publique que tout le monde peut détenir et une clé privée gardée secrète ; utilisée pour l'échange de clés, les signatures numériques et les certificats.

CVE
Common Vulnerabilities and Exposures : programme qui attribue à chaque vulnérabilité publiquement connue un identifiant unique, comme CVE-2021-44228.

CVSS
Common Vulnerability Scoring System, maintenu par FIRST, qui évalue la gravité d'une vulnérabilité connue de 0 à 10 ; la version 4.0 a été publiée en 2023.

Défense en profondeur
Superposition de plusieurs contrôles indépendants de sorte qu'un attaquant qui en déjoue un se heurte encore aux autres, par exemple une vérification au niveau de la passerelle, une vérification au niveau du service et le chiffrement des données elles-mêmes.

Dépendance transitive
Paquet que votre logiciel utilise indirectement parce que l'une de vos dépendances, ou l'une des leurs, en dépend.

Diagramme de flux de données (DFD)
Diagramme des entités externes, des processus, des stockages de données et des flux de données entre eux, avec des frontières de confiance tracées en travers des flux ; le modèle habituel pour la modélisation des menaces.

Données à caractère personnel
Au sens du RGPD, toute information se rapportant à une personne physique identifiée ou identifiable (une personne vivante), comme un nom, un numéro d'identification, des données de localisation ou un identifiant en ligne.

EPSS
Exploit Prediction Scoring System de FIRST, qui estime la probabilité qu'une vulnérabilité soit exploitée dans les 30 prochains jours.

Fondement juridique (base légale)
L'un des six fondements prévus à l'article 6 du RGPD (consentement, contrat, obligation légale, intérêts vitaux, mission d'intérêt public, intérêts légitimes) sur lequel tout traitement de données à caractère personnel doit reposer.

Frontière de confiance
Ligne dans un système où des données ou le contrôle passent entre des parties ayant des niveaux de confiance ou de privilège différents, par exemple d'Internet vers une API ; les entrées doivent donc y être vérifiées.

Gestion des secrets
Stocker, distribuer, renouveler et auditer des identifiants comme les mots de passe, les clés d'API et les clés privées dans un coffre-fort dédié plutôt que dans le code ou les fichiers de configuration.

Hachage des mots de passe
Stocker un mot de passe sous la forme du résultat d'une fonction délibérément lente et salée, comme Argon2id, scrypt ou bcrypt, afin qu'une base de données volée ne puisse pas être reconvertie à peu de frais en mots de passe.

Identité de charge de travail (workload identity)
Identité vérifiable attribuée à un service en fonctionnement plutôt qu'à une personne, délivrée par la plateforme (par exemple un SPIFFE ID dans un certificat de courte durée), utilisée à la place de secrets partagés.

Infrastructure à clés publiques (PKI)
Les autorités de certification, les certificats, les magasins de confiance, les politiques et les mécanismes de révocation qui, ensemble, permettent aux parties de vérifier mutuellement leurs clés publiques.

JSON Web Token (JWT)
Jeton compact, signé (et éventuellement chiffré), qui porte des déclarations (claims) telles que l'émetteur, le sujet, l'audience et l'expiration, défini par la RFC 7519 ; très utilisé pour les jetons d'accès et les jetons d'identité (ID tokens).

Lockfile (fichier de verrouillage)
Fichier généré par un gestionnaire de paquets qui enregistre les versions exactes, et généralement les empreintes (hashes), de chaque dépendance résolue, afin que chaque installation utilise les mêmes paquets.

Microsegmentation
Diviser un réseau en zones très petites, jusqu'à la charge de travail individuelle, afin que chacune n'accepte le trafic que des appelants dont elle a besoin.

Modélisation des menaces
Manière structurée de se demander, à propos d'une conception, ce que l'on construit, ce qui peut mal tourner, ce que l'on va faire à ce sujet et si l'on a fait un travail suffisamment bon.

Module de sécurité matériel (HSM)
Matériel inviolable qui génère et utilise des clés en son sein et ne les divulgue jamais en clair ; les services KMS reposent généralement sur des HSM.

Moindre privilège
Principe selon lequel chaque utilisateur, programme et service ne reçoit que les permissions dont il a besoin pour sa tâche, et seulement pour la durée nécessaire, afin qu'une erreur ou une compromission cause des dégâts limités.

Mouvement latéral
La progression d'un attaquant depuis la première machine ou le premier compte compromis vers d'autres systèmes du même réseau ou de la même organisation.

Nomenclature logicielle (SBOM)
Inventaire lisible par machine des composants et des dépendances d'un logiciel, dans un format standard comme SPDX ou CycloneDX.

OAuth 2.0
Le cadre d'autorisation déléguée de la RFC 6749, dans lequel un serveur d'autorisation délivre des jetons d'accès qui permettent à un client d'appeler un serveur de ressources pour le compte d'un propriétaire de ressources, sans voir son mot de passe.

OpenID Connect (OIDC)
Couche d'identité au-dessus d'OAuth 2.0 qui ajoute un jeton d'identité (ID token), un JWT signé indiquant au client qui est l'utilisateur, ainsi que comment et quand il s'est authentifié.

Passkey (clé d'accès)
Identifiant FIDO2/WebAuthn : une paire de clés créée pour un site web donné, dont la clé privée reste sur l'appareil ou dans le gestionnaire de mots de passe de l'utilisateur, ce qui rend la connexion résistante à l'hameçonnage et ne laisse aucun secret partagé sur le serveur.

Piste d'audit
Enregistrement complet, chronologique et infalsifiable sans trace des actions définies, montrant qui a fait quoi et quand, conservé pour établir les responsabilités.

PKCE
Proof Key for Code Exchange (RFC 7636) : le client envoie l'empreinte d'un secret à usage unique lorsqu'il démarre le flux de code d'autorisation, puis le secret lui-même lorsqu'il échange le code, de sorte qu'un code volé est inutilisable.

Point d'application de politique (PEP)
Le composant situé sur le chemin du trafic, comme un proxy, une passerelle ou un sidecar, qui autorise ou bloque chaque requête selon une décision de politique.

Point de décision de politique (PDP)
Le composant qui évalue la politique d'accès et répond « autoriser » ou « refuser » ; un point d'application de politique (PEP) l'interroge, puis applique la réponse.

Protection des données dès la conception et par défaut
Obligation de l'article 25 du RGPD d'intégrer les principes de protection des données dans les systèmes au moyen de mesures appropriées, et de ne traiter par défaut que les données nécessaires à chaque finalité.

Provenance
Informations vérifiables sur l'origine d'un artefact : quelle plateforme de build l'a construit, à partir de quelle révision du code source, avec quelles entrées et quels paramètres.

Pseudonymisation
Traiter des données à caractère personnel de façon qu'elles ne puissent plus être attribuées à une personne sans recourir à des informations supplémentaires conservées séparément ; au sens du RGPD, les données pseudonymisées restent des données à caractère personnel.

Règlement sur la cyberrésilience (Cyber Resilience Act, CRA)
Règlement (UE) 2024/2847, qui fixe des exigences de cybersécurité ainsi que des obligations de gestion et de signalement des vulnérabilités pour les produits matériels et logiciels vendus dans l'UE.

Réponse aux incidents
Traitement organisé d'un incident de sécurité : le détecter et l'analyser, le contenir, éliminer la cause, rétablir le service et en tirer des enseignements.

Responsable du traitement (controller)
Organisation qui détermine les finalités et les moyens du traitement des données à caractère personnel et qui porte la responsabilité première du respect du droit de la protection des données.

Runbook (procédure d'exploitation)
Procédure écrite et testée pour traiter pas à pas une situation d'exploitation ou de sécurité précise, y compris la manière de vérifier chaque étape et de revenir en arrière.

Sauvegarde immuable
Copie de sauvegarde qui ne peut être ni modifiée ni supprimée avant la fin de sa durée de conservation, même par un administrateur, ce qui la protège contre les rançongiciels et les personnes malveillantes en interne.

Secure Software Development Framework (SSDF)
NIST SP 800-218 : ensemble de pratiques de développement sécurisé regroupées sous les intitulés « préparer l'organisation », « protéger le logiciel », « produire un logiciel bien sécurisé » et « répondre aux vulnérabilités ».

Sécurisé par défaut
Livrer un produit dont la configuration d'origine est la configuration sûre, de sorte que les utilisateurs doivent choisir délibérément d'affaiblir la protection plutôt que de penser à l'activer.

Sécurité périmétrique
Le modèle traditionnel qui protège un réseau interne de confiance par des pare-feu et des VPN et considère comme amical tout ce qui se trouve déjà à l'intérieur ; on parle parfois de sécurité « château fort ».

Service de gestion de clés (KMS)
Service managé qui crée, stocke, renouvelle et contrôle l'usage des clés cryptographiques, en effectuant lui-même les opérations avec elles afin que les applications ne voient jamais les clés maîtres.

Shift left
Avancer les activités de sécurité (ou de test) dans le cycle de développement, dans les exigences, la conception et le codage, là où les problèmes coûtent moins cher à corriger qu'en production.

SIEM
Security information and event management (gestion des informations et des événements de sécurité) : système qui collecte les journaux de sécurité de nombreuses sources, les corrèle et déclenche des alertes lorsqu'ils correspondent à des règles de détection.

Sigstore
Projet open source pour signer et vérifier des artefacts logiciels au moyen de certificats de courte durée liés à une identité et d'un journal de transparence public, ce qui évite les clés de signature de longue durée.

SLSA
Supply-chain Levels for Software Artifacts : cadre de niveaux d'assurance croissants qu'un artefact a été construit à partir du code source attendu par un build digne de confiance et résistant aux altérations.

Sous-traitant (processor)
Organisme qui traite des données à caractère personnel pour le compte d'un responsable du traitement, selon ses instructions et dans le cadre d'un contrat, par exemple un fournisseur cloud d'envoi de SMS ou d'e-mails.

STRIDE
Moyen mnémotechnique pour six types de menaces (usurpation d'identité ou spoofing, falsification ou tampering, répudiation, divulgation d'informations, déni de service et élévation de privilèges), créé chez Microsoft par Loren Kohnfelder et Praerit Garg en 1999.

Surface d'attaque
Ensemble des points par lesquels un attaquant peut tenter d'introduire des données dans un système, d'en extraire ou d'interagir autrement avec lui : endpoints, ports, fichiers, files, outils d'administration, ainsi que les personnes et le code auxquels il fait confiance.

TLS mutuel (mTLS)
TLS dans lequel les deux parties présentent et vérifient des certificats, de sorte qu'un service sait quel service l'appelle, en plus de l'appelant qui connaît le serveur.

Tokenisation
Remplacer une valeur sensible, comme un numéro de carte, par un jeton dépourvu de signification, tandis que la valeur réelle est conservée dans un coffre-fort distinct et étroitement protégé, souvent exploité par un prestataire de paiement.

Violation de données à caractère personnel
Au sens de l'article 4, point 12, du RGPD, une violation de la sécurité entraînant, de manière accidentelle ou illicite, la destruction, la perte, l'altération, la divulgation non autorisée de données à caractère personnel, ou l'accès non autorisé à de telles données.

Zero trust (confiance zéro)
Modèle de sécurité qui n'accorde aucune confiance implicite fondée sur l'emplacement réseau : chaque requête est authentifiée, autorisée et chiffrée, par ressource et par session (NIST SP 800-207).