ca technologies

Cinq stratégies simples pour sécuriser les API

Par Scott Morrison, CA Technologies

Table des matières

Qu’est-ce qu’une API ? Vaut-elle la peine de prendre des risques ?

Comment rendre une entreprise flexible ?

Une API est une technologie émergente permettant d'intégrer des applications à l'aide de la technologie Web. Cette approche connaît un essor considérable, car elle utilise des techniques bien maîtrisées et s'appuie sur une infrastructure existante. Il ne faudrait toutefois pas pour autant penser qu'il est possible de sécuriser les API à l'aide des mêmes méthodes et technologies que celles utilisées pour la sécurisation du Web conventionnel, centré sur un navigateur. Même s'il est vrai qu'un grand nombre des menaces inhérentes au Web peuvent aussi frapper les API, ces dernières sont fondamentalement différentes et possèdent un profil de risque qu'il est nécessaire de gérer.

Cette brochure électronique présente un aperçu de ces nouveaux risques et propose cinq solutions simples pour contrer les menaces courantes. En mettant en œuvre une architecture API sécurisée dès le départ, les organisations peuvent poursuivre leur stratégie dans ce domaine, de manière sûre et sécurisée, et profiter de tous les avantages de l'intégration agile promise par cette formidable technologie.

Section suivante
Table des matières

Qu’est-ce qu’une API ?

Depuis l’avènement de l’informatique, les développeurs s’efforcent de faire communiquer les applications. Des protocoles, tels que COM+, CORBA ou même SOAP, ont vu le jour au fil des ans, mais aucun d’eux n’était suffisamment puissant pour répondre aux besoins d’évolutivité, de simplicité et de fonctionnalités de compatibilité entre les différents langages.

Pourtant, la réponse a toujours été sous leurs yeux. Le Web a été le premier système véritablement évolutif et distribué, capable de relier des plates-formes disparates à l’aide d’un protocole simple à comprendre et beaucoup plus puissant qu’il n’y paraissait. L’idée de base était de s’appuyer sur ce succès pour intégrer des applications autres que la combinaison navigateur/serveur Web pour lequel le protocole avait été conçu.

L’API est la technologie qui sous-tend cette approche. Les API permettent aux développeurs de créer une architecture ouverte pour le partage de fonctionnalités et de données entre les applications. **Les API sont comme les fenêtres d’une application**, un passage qui mène directement aux principales fonctionnalités et données résidant au cœur de l’application.

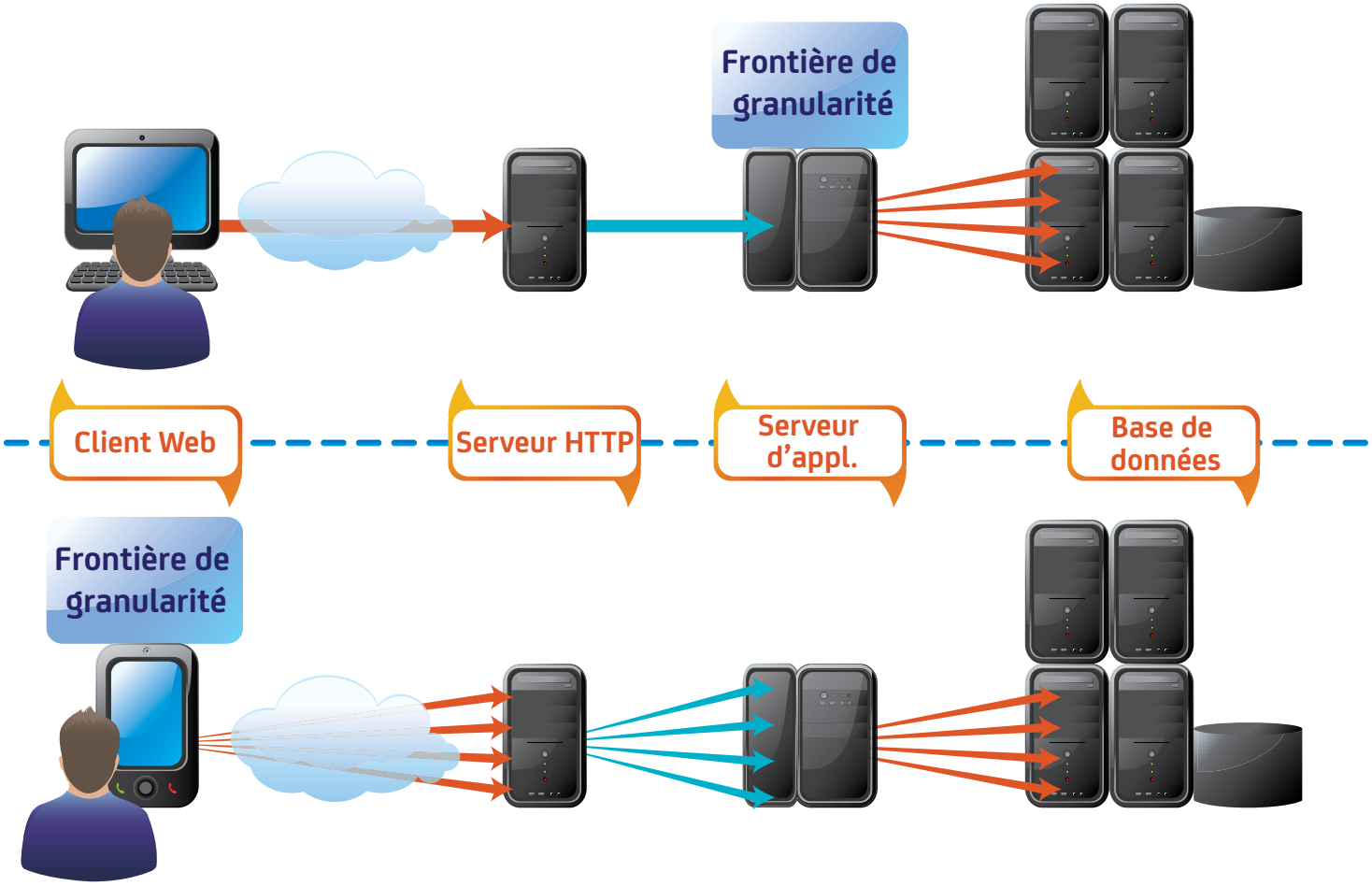
Les API donnent aux développeurs côté-client (qu’ils soient légitimes ou malveillants), un accès beaucoup plus détaillé à l’application qu’une application Web traditionnelle. Ceci s’explique par le fait que la frontière de granularité des appels vers les niveaux d’arrière-plan dépasse les niveaux internes relativement sécurisés (ceux qui résident dans la zone démilitarisée), pour s’étendre vers l’extérieur, jusqu’au niveau de l’application client résidant sur Internet.

Cette explication peut sembler un peu nébuleuse. En fait, la frontière de granularité décrit simplement la portion des systèmes d’arrière-plan à laquelle une application appelante peut accéder. Malheureusement, c’est précisément tout ce qui fait l’attrait des API, qui en fait aussi la cible idéale des pirates informatiques.

Pourquoi les API augmentent-elles les risques encourus par une organisation ?

Le problème avec les API est qu'elles donnent souvent des informations détaillées décrivant l'implémentation sous-jacente d'une application ; des détails qui, dans un contexte Web, seraient dissimulés sous des couches de fonctionnalités d'applications. Ces informations peuvent être exploitées par des pirates informatiques et leur offrir des vecteurs d'attaque qu'ils auraient autrement ignorés. Les API tendent en effet à être extrêmement claires et bien documentées, mais aussi à donner des informations précises sur les objets internes et même sur la structure de la base de données interne, autant d'informations exploitables par les pirates.

Mais cette visibilité accrue n'est pas le seul risque que comportent les API. L'augmentation du nombre d'appels potentiels a également pour effet d'élargir la surface d'attaque et ainsi de doter les pirates d'un terrain plus vaste à exploiter. Les risques augmentent avec les occasions de nuire.



Les vieilles habitudes ont la vie dure

Bien que les API représentent un risque pour l'entreprise, les bénéfices potentiels qu'elles peuvent lui apporter l'emportent largement sur les dangers inhérents. La plus grande menace réside sans doute dans la façon dont elles sont implémentées.

Bien conçue, une API permet aux organisations de fournir de puissants outils Web, directement à leurs employés et leurs clients. Un bon développeur d'API comprend le profil de risque de ce qu'il conçoit. Cela dit, il se trouve que de nombreux développeurs d'API sont d'anciens concepteurs de sites Web qui, malheureusement, ont gardé de mauvaises habitudes. Il est important de réaliser que, bien qu'elles partagent une infrastructure et des racines communes, la conception Web et la conception d'API poursuivent des objectifs distincts et requièrent donc des approches différentes.

Les pirates utilisent principalement trois vecteurs pour s'attaquer aux API. La connaissance de ces vecteurs d'attaque vous permettra de concevoir des API plus sûres.

Les trois vecteurs d'attaque à surveiller

Vecteurs : explications détaillées

Les vecteurs d'attaque les plus courants peuvent être divisés en trois catégories.

Paramètres

Les attaques par manipulation de paramètres exploitent les données envoyées dans une API, notamment les URL, les paramètres de requête, les en-têtes HTTP et/ou les contenus publiés.

Identité

Les attaques liées à l'identité exploitent les failles au niveau de l'authentification, des autorisations et des suivis de session. Un grand nombre de ces failles résultent de mauvaises pratiques en provenance du Web et qui sont appliquées au développement des API.

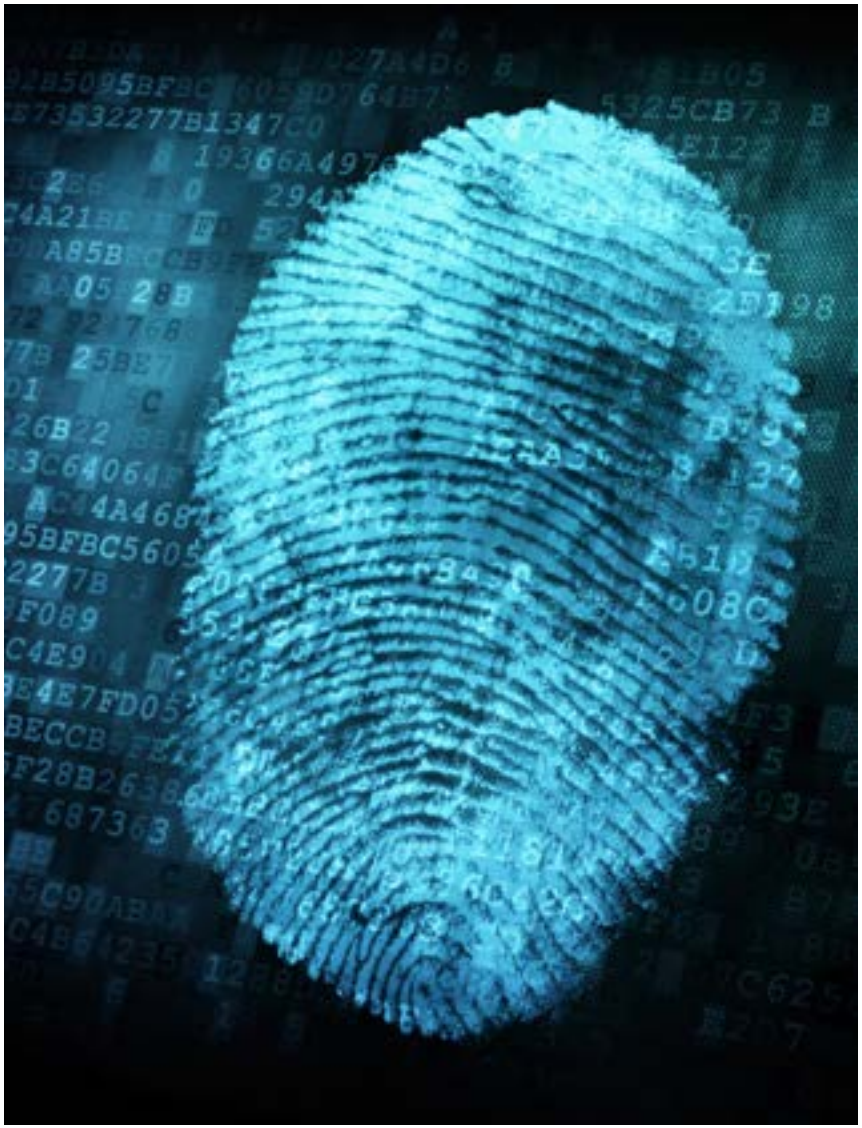
Attaques par interception (Man in the Middle)

Ces attaques interceptent les transactions légitimes et exploitent les données non signées et/ou non chiffrées échangées entre le client et le serveur. Elles peuvent dévoiler des informations confidentielles (telles que des données personnelles), altérer une transaction en cours ou même reproduire des transactions légitimes.

Vecteur d'attaque : *Paramètres*

Les attaques par manipulation de paramètres, dont la plus courante est l'injection SQL, tentent de manipuler un système en lui fournissant des informations qui exploitent le comportement des applications et l'infrastructure qui les supporte (telle que les bases de données). Elles résultent souvent de la négligence des développeurs qui omettent de contrôler de façon suffisamment rigoureuse les informations entrées dans l'application. En outre, contrairement à ce qui est d'usage sur de nombreuses applications Web, le nom de la fonction sous-jacente des paramètres des API est souvent explicite, offrant un indice bien tentant pour un pirate.

Les attaques par manipulation de paramètres ne sont certainement pas nouvelles, ce vecteur est exploité sur le Web depuis des années. Elles sont cependant en recrudescence dans le monde des API, parce que de nombreux développeurs omettent de nettoyer régulièrement les entrées, car ils sont habitués à ce que ce processus soit effectué automatiquement par de nombreux protocoles Web. Pourtant les API sont tout autant exposées à ces menaces et des précautions s'imposent pour limiter les risques.



Vecteur d'attaque : *Identité*

Le concept d'identité de l'utilisateur nous est familier. Les API introduisent également le concept d'identité de l'application. Il s'agit une clé qui identifie de manière unique l'application qui appelle l'API. Cette clé est communément appelée « clé API », pour le meilleur et pour le pire. Cette clé API est répliquée au niveau de chaque instance de l'application. Elle entend supporter une gestion basique du client, comme la limitation du débit, de manière à ce qu'une application devenue très populaire ne monopolise pas l'API au détriment d'applications ne jouissant pas de la même popularité.

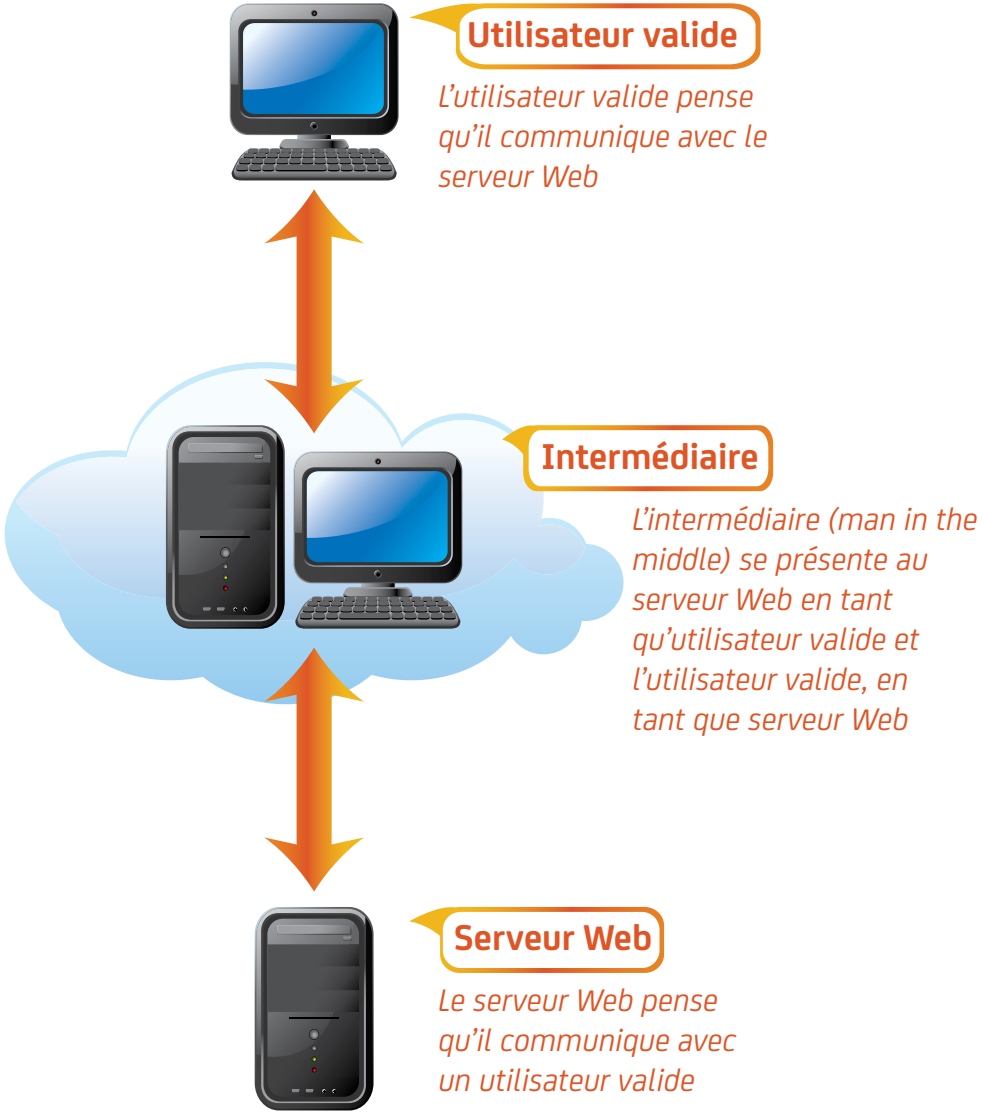
Malheureusement, les clés d'API sont souvent prises pour ce qu'elles ne sont pas, à savoir des informations d'identification autoritaires. Les clés d'API sont généralement cachées à l'intérieur du code d'une application client appelante, et malgré tous les efforts des développeurs pour les dissimuler, elles sont généralement faciles à trouver et à exploiter. La sécurisation de l'authentification des applications a toujours été, et demeure, un problème épineux à résoudre. Les API l'exacerbent encore, mais il existe malheureusement peu de façons de contrer ce problème. En réalité, il convient de ne jamais traiter une clé d'API comme si elle était secrète.

Section suivante
Table des matières

Vecteur d'attaque par interception : *Man in the middle*

Une attaque par interception décrit une situation dans laquelle un attaquant vient s'interposer entre l'expéditeur et le destinataire des informations. Il peut le faire de manière invisible, ou se faire explicitement passer pour une partie ou l'autre, mais dans tous les cas, il utilise sa position pour exploiter les échanges de données non signées ou non codées.

Les API qui ne sont pas correctement configurées à l'aide de SSL/TLS sont extrêmement vulnérables à ce type d'attaque. Malheureusement, en matière de conception Web, l'usage est de ne pas coder la plupart des communications, notamment en raison de problèmes passés liés au dimensionnement des charges SSL. Et même lorsque SSL/TLS est appliqué, il est souvent mal configuré ou vulnérable aux attaques de type « downgrade » (destruction du chiffrement sur une connexion) qui le rende inefficace. Dans le monde des API, les enjeux priment et la protection de l'acheminement est indispensable pour sécuriser les données, les sessions et l'accès aux fonctionnalités.



Cinq stratégies simples pour publier les API de manière plus sécurisée

Même si les API peuvent faire l'objet de nombreuses attaques, la mise en œuvre de cinq stratégies très simples peut permettre à une organisation de publier des API en toute sécurité.

Section suivante
Table des matières

Stratégie 1 : *Valider les paramètres*

La première étape de l'implémentation d'une API résistante est d'assainir toutes les données entrantes, afin de confirmer qu'elles sont valides et sans danger. La manière la plus efficace de se défendre contre les attaques par manipulation de paramètres et par injection, est de valider toutes les données entrantes suivant un schéma strict, qui consiste en une description des entrées considérées comme admissibles dans le système. La validation par schéma doit être la plus restrictive possible et utiliser des saisies, des plages, des ensembles et même des listes explicites d'utilisateurs certifiés, chaque fois que c'est possible. Il convient également de prendre en compte le fait que les schémas générés automatiquement par de nombreux outils de développement, réduisent souvent tous les paramètres à des modèles qui sont beaucoup trop vastes pour permettre l'identification efficace des menaces potentielles. Il convient de privilégier les schémas et listes d'utilisateurs certifiés établis manuellement car les développeurs peuvent limiter le nombre d'entrées en fonction de leur propre compréhension du modèle de données requis par l'application.

Pour les types de contenu basés sur XML, l'une des possibilités est d'utiliser le langage du schéma XML, particulièrement efficace pour la création de modèles de contenu restreints et de structure hautement contraignantes. Pour les types de données JSON, de plus en plus répandues, il existe différents langages de description du schéma JSON. Même s'il n'est pas aussi riche que XML, JSON est beaucoup plus simple à composer et à appréhender, offrant une transparence qui le rend en fait plus simple à sécuriser.

Section suivante
Table des matières

Stratégie 2 : *Effectuer une détection systématique des menaces*

Une validation adaptée par le biais de schémas peut protéger contre de nombreuses attaques par injection, mais il convient également d'envisager la mise en place d'analyses explicites pour détecter certaines attaques courantes. Les attaques par injection SQL ou injection de scripts se trahissent souvent elles-mêmes en suivant des modèles communs qui sont faciles à identifier via l'analyse d'entrées brutes.

Ne perdez pas de vue non plus que les attaques peuvent prendre d'autres formes, comme le déni de service (DoS). Tirez parti de l'infrastructure réseau pour identifier et limiter les attaques DoS au niveau du réseau, mais recherchez également les attaques DoS qui utilisent ces paramètres. Des messages volumineux, des structures de données lourdement imbriquées ou excessivement complexes peuvent favoriser la réussite d'une attaque par déni de service, consommant inutilement les ressources du serveurs d'API affecté.

Procédez aussi à une détection des virus sur les contenus codés présentant des risques potentiels. Les API impliquées dans le transfert de fichiers doivent pouvoir décoder les pièces jointes en base64 et les soumettre à une analyse de virus au niveau du serveur, avant de les placer dans un système de fichiers où elles pourraient être activées par inadvertance.

Stratégie 3 : *Activer systématiquement SSL*

Faites de SSL/TLS la règle pour toutes les API. Au 21e siècle, SSL n'est pas un luxe ; c'est une exigence de base. L'ajout systématique de SSL/TLS, et son application correcte, sont un moyen efficace de se défendre contre les attaques par interception.

SSL/TLS protège l'intégrité de toutes les données échangées entre un client et un serveur, notamment à l'aide d'importants jetons d'accès, tels que ceux utilisés dans OAuth. Il offre également la possibilité d'effectuer une authentification côté client à l'aide de certificats, ce qui peut être important dans de nombreux environnements.

Section suivante
Table des matières

Stratégie 4 : *Mettre en œuvre un système rigoureux d'authentification et d'autorisation*

L'identité de l'utilisateur et de l'application est un concept qui doit être implémenté et géré séparément. L'autorisation doit être envisagée dans un contexte large d'identification, en prenant notamment en compte des facteurs pratiques tels que les adresses IP entrantes (si elles sont fixes ou se situent entre des limites particulières), les fenêtres de temps d'accès, l'identification des appareils (utile pour les applications mobiles), la géolocalisation, etc.

OAuth est rapidement en train de s'imposer en tant que ressource privilégiée pour les autorisations liées aux API centrées sur les utilisateurs, mais elle demeure une technologie complexe et difficile qui évolue rapidement. Les développeurs devraient s'en remettre aux cas d'utilisation de base, bien maîtrisés du protocole OAuth et toujours utiliser les bibliothèques existantes plutôt que d'essayer d'en créer une eux-mêmes.

Section suivante
Table des matières

Stratégie 5 : *Utiliser des solutions éprouvées*

La première règle en matière de sécurité est de ne pas inventer votre propre solution. Il n'est nullement nécessaire de créer votre infrastructure de sécurité pour les API, car d'excellentes solutions existent déjà. Le seul défi consiste à les utiliser de manière appropriée.

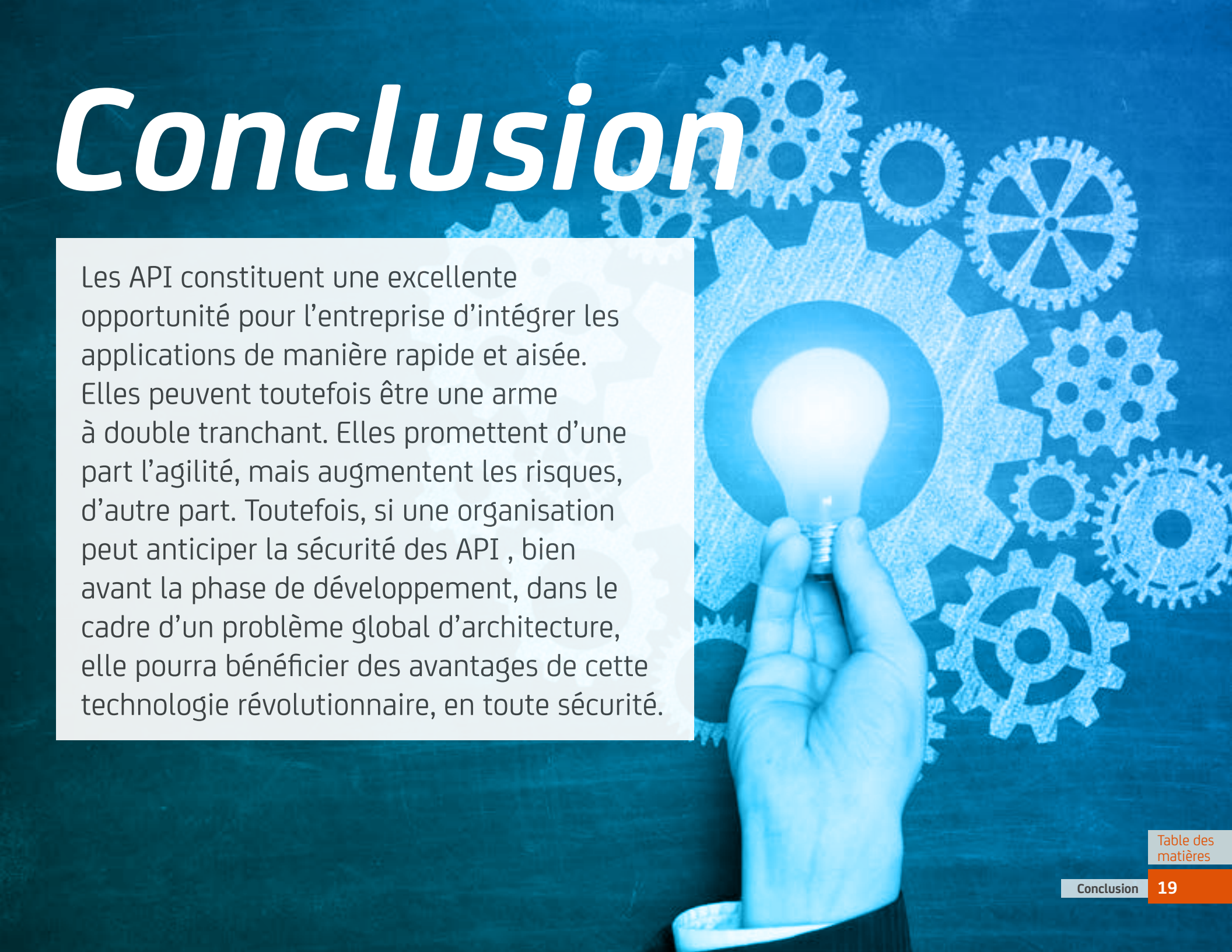
Sécurisation des architectures d'API

Le meilleur moyen de protéger vos API de tout type d'intrusion est de séparer leur implémentation de leur sécurisation en deux niveaux distincts. Cette séparation très logique des préoccupations vise à ce que l'expertise se concentre sur le bon problème, au bon moment.

Cette approche permet de ne plus monopoliser le développeur d'API qui pourra dès lors se consacrer entièrement à l'application en veillant à ce que chaque API soit bien conçue et qu'elle facilite l'intégration entre différentes applications. La sécurité relève alors des compétences d'un expert, qui peut se concentrer exclusivement sur l'identité, les menaces et la sécurité des données.

Conclusion

Les API constituent une excellente opportunité pour l'entreprise d'intégrer les applications de manière rapide et aisée. Elles peuvent toutefois être une arme à double tranchant. Elles promettent d'une part l'agilité, mais augmentent les risques, d'autre part. Toutefois, si une organisation peut anticiper la sécurité des API , bien avant la phase de développement, dans le cadre d'un problème global d'architecture, elle pourra bénéficier des avantages de cette technologie révolutionnaire, en toute sécurité.



Pour en savoir plus sur les risques inhérents et les avantages des API :

www.ca.com/techinsights/security
Consultez des documents de présentation technique et rapports de recherche de premier plan pour mieux comprendre les tendances dominantes en matière de sécurité.

www.ca.com/fr/api-security-and-management.aspx
Découvrez comment connecter votre entreprise aux applications mobiles, aux plates-formes Cloud et aux réseaux de développeurs, de manière sécurisée grâce aux API.

ca technologies