GitHub

Développez en toute sécurité pour les services financiers

Table of Contents

2 INTRODUCTION

4 SAFEGUARDING CUSTOMER DATA

Build a culture of security

Always encrypt data

- Encryption in transit
- Encryption at rest
- Encryption with GitHub

Adopt strong identity management practices

- Using GitHub with identity management systems

Prevent database attacks

Segregate sensitive data

- Preventing sensitive data from reaching repositories with pre-receive hooks

Enforce protections on source code

10 SECURING THE ENTIRE DEVELOPMENT PROCESS

Include legacy core systems

Automate processes to prevent bugs and errors

- Integrating CI/CD with GitHub

Enforce manual reviews with required code review

Move security all the way left

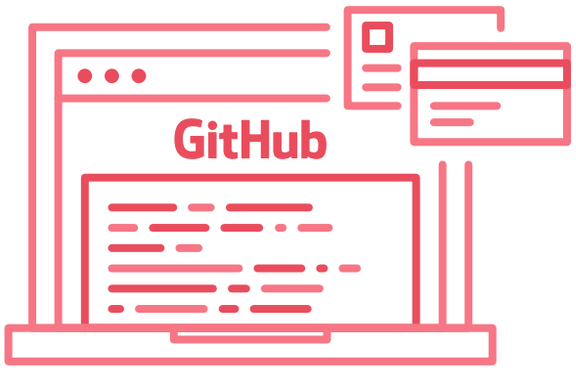
- Creating a security culture

16 CUSTOMER STORY: ANAPLAN

18 SUMMARY

We're here to help

Introduction



La confiance est la base même des relations entre les institutions financières et leurs clients, mais aussi entre elles-mêmes. Les moyens dont disposent ces institutions pour empêcher que les données sensibles ne se retrouvent entre de mauvaises mains sont au cœur de cette confiance. À l'échelle internationale, les gouvernements ont créé divers cadres réglementaires contraignant les institutions financières à protéger les données de leurs clients, Ne pas remplir ces obligations, c'est s'exposer à de lourdes sanctions , risquer sa réputation et mettre en péril son activité à long terme.

Pour compliquer davantage la situation, les logiciels jouent un rôle central dans les activités des institutions financières, et ce rôle continue de gagner en importance. Or, des exigences strictes en matière de réglementation et de conformité régissent la façon dont les institutions financières développent leurs logiciels, depuis les processus déployés jusqu'aux outils qu'elles utilisent. Ces exigences, souvent rédhibitoires, compromettent les performances et freinent l'innovation.

Bonne nouvelle : le développement sécurisé n'est pas forcément un obstacle à la collaboration ou à l'innovation. Les institutions financières sont particulièrement bien placées pour concevoir des expériences utilisateur innovantes, ce que bon nombre d'entre elles font déjà, par ailleurs.
Des milliers d'entreprises utilisent GitHub pour libérer leurs workflows des méthodes de développement isolé et créer des processus sécurisés accordant aux équipes d'ingénieurs la souplesse nécessaire à un travail optimal.

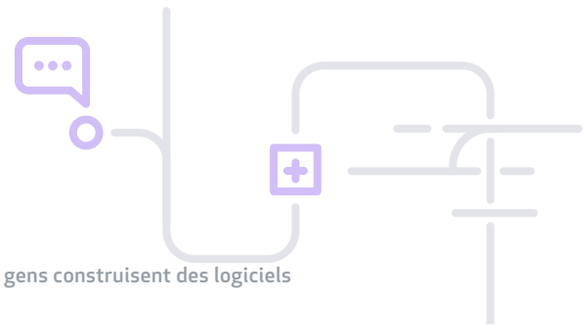
Face à des clients plus exigeants que jamais et sous la pression croissante de la maîtrise des coûts, des workflows efficaces, collaboratifs et sécurisés peuvent aider les équipes à se concentrer sur ce qui compte le plus : le développement de logiciels les plus performants et les plus innovants pour ces clients.

Dans ce guide, nous aborderons les problématiques réglementaires et techniques auxquelles sont confrontées les institutions financières, et en proposant des solutions et en expliquant en quoi GitHub peut aider.

La protection des données client

Les Institution finacières traitent certaines des données les plus sensibles au monde. Elles sont entraînées dans une course effrénée contre ceux qui cherchent à s'approprier ces données à des fins malveillantes, qu'il s'agisse de pirates informatiques isolés ou d'intervenants agissant en groupe, ils font preuve de ressources et de compétences dans le but de porter atteinte à la sécurité. Fort heureusement, de bonnes pratiques et certains outils peuvent aider ces institutions à sécuriser leurs processus de développement logiciel et à protéger les données sensibles.

Aucune solution ou approche unique ne peut garantir la sécurité. En réalité, une sécurisation efficace fait appel à une approche à plusieurs niveaux visant à appliquer des contrôles et des mesures de protection en plusieurs points du cheminement des données et des workflows. Cette approche en niveaux prévoit différents volets : les outils, les pratiques et la culture.

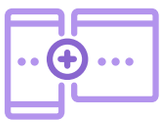


Instaurez une culture axée autour de la sécurité

Pour appliquer une politique de sécurité à grande échelle, la sécurité doit devenir une priorité. Cela peut se faire, par exemple, en créant des processus qui facilitent le travail de tous les collaborateurs. Il s'agit notamment de prévoir des formations régulières sur les pratiques de sécurité et de sensibiliser aux moyens techniques et sociaux d'infiltration, comme le phishing.

PARMI LES BONNES PRATIQUES, CITONS :

l'application d'une politique pour utiliser des mots de passe et en renforcer la sécurité avec, entre autres, des mots de passe complexes, un changement régulier des mots de passe et l'utilisation de gestionnaires et de générateurs de mots de passe ;



la mise en œuvre de l'authentification à deux facteurs ;

la formation et la sensibilisation aux dangers des réseaux publics, ainsi qu'à la protection physique des appareils tels que les ordinateurs et téléphones portables, et les tablettes ;

l'utilisation de VPN pour sécuriser l'accès aux réseaux internes d'entreprise.

Chiffrez systématiquement les données

Le chiffrement est l'un des meilleurs moyens d'empêcher les accès non autorisés aux données. Même si un imposteur réussissait à accéder à la transmission de données, un chiffrement puissant l'empêcherait de porter atteinte à leur sécurité. Là encore, une approche à plusieurs niveaux est plus efficace dans la mise en œuvre d'une stratégie de chiffrement. Cela implique :

le chiffrement des données en transit lors lors de leur transfert entre une source de stockage et un utilisateur final ;

le chiffrement des données sur disque dans les bases de données où elles sont stockées.

CHIFFREMENT EN TRANSIT

En pratique, tout initiative Web repose sur un chiffrement de bout en bout utilisant des protocoles de chiffrement modernes tels que TLS. TLS contribue à sécuriser les communications grâce à un chiffrement fort, basé sur des certificats, un système en pratique infaillible. TLS vérifie également chaque partie impliquée dans une transaction réseau. Il empêche ainsi les attaques dites de l'homme du milieu, où un imposteur se présente comme partenaire de confiance au cours d'une transaction. Des services comme LetsEncrypt ont considérablement réduit le coût et la complexité d'une mise en œuvre efficace du protocole TLS et de l'obtention des certificats requis.

CHIFFREMENT SUR DISQUE

Certaines applications fournissent des services de chiffrement au niveau logiciel. Ce type de chiffrement est préférable à une absence de protection, mais peut engendrer des coûts importants en termes de complexité et de dégradation de la performance. Les solutions de chiffrement qui fonctionnent au plus près du système de stockage, au niveau du système de fichiers ou du périphérique physique, peuvent fournir une couche transparente de chiffrement pour toutes les applications.

CHIFFREMENT AVEC GITHUB

Toutes les communications à destination et en provenance de GitHub Enterprise Server sont protégées par TLS. Il est possible d'utiliser LetsEncrypt pour maintenir les certificats de manière simple et économique. GitHub sécurise également la communication entre les clients Git utilisés par les développeurs et le serveur à l'aide de SSH. Ces deux puissants protocoles de chiffrement sont capables de faire barrage aux mécanismes d'interception non autorisée ou à la fuite de données sensibles.

Adoptez une gestion stricte des identités

Plus les mots de passe à mémoriser sont nombreux, plus les utilisateurs sont susceptibles de commettre des imprudences, par exemple, d'écrire des mots de passe ou de réutiliser des mots de passe faciles à mémoriser (et donc faciles à déchiffrer). L'utilisation d'un système de gestion des identités centralisé tel que LDAP avec une solution d'authentification unique (SSO) peut réduire considérablement le nombre de mots de passe que les utilisateurs doivent gérer. Ils peuvent ainsi facilement adopter des pratiques sûres pour gérer leurs mots de passe.

GITHUB ET LES SYSTÈMES DE GESTION DES IDENTITÉS

Enterprise Server supporte l'authentification et l'autorisation au moyen de diverses solutions de gestion des identités telles qu'Active Directory et LDAP. GitHub supporte également SAML, ce qui permet aux entreprises d'offrir une expérience SSO aux utilisateurs.

Empêchez les attaques de base de données

pirates exploitent pour tenter d'accéder aux systèmes. Ces attaques reposent souvent sur la transmission de charges utiles nuisibles à des systèmes mal sécurisés et l'exploitation de failles pour en prendre le contrôle. La technique SQLi (SQL Injection) est un cas classique : l'imposteur insère du code SQL dans des données en apparence inoffensives, par exemple, le nom d'un client dans une application Web ; des failles dans le logiciel sous-jacent peuvent alors entraîner l'exécution arbitraire de ce code, ce qui engendre le renvoi involontaire des données à l'utilisateur. Ces attaques sont particulièrement dangereuses car elles ne provoquent pas nécessairement d'erreur ou d'autre événement susceptible d'attirer l'attention des administrateurs chargés de la sécurité.

Ici aussi, une approche à plusieurs niveaux est plus efficace pour garantir que seules des données saines atteignent votre environnement. Cette sécurisation commence au niveau du code de l'interface utilisateur et doit s'appliquer à tout le cheminement des données, du middleware jusqu'aux systèmes de stockage sous-jacents. L'intégration d'une validation dans le code à chaque étape peut réduire considérablement votre vulnérabilité face à ce type d'attaque. Cet effort de sécurisation doit absolument s'accompagner de tests et d'essais manuels et automatisés fréquents.

Trois façons de prévenir les failles de sécurité courantes :

- Acceptez uniquement les données validées dans les systèmes de production
- Surveillez les systèmes à la recherche d'erreurs et de cas anormaux
- Utilisez un système de gestion des identités pour restreindre l'accès aux systèmes de données des enregistrements au maximum

Isolez les données sensibles

Moins il y a d'endroits où les données sensibles sont stockées, moins il existe de failles (et de « surface ») à exploiter. La réduction de la surface implique la mise en œuvre de diverses stratégie. Par exemple, vous pouvez utiliser un gestionnaire de mots de passe plutôt que de laisser chacun les gérer individuellement. Autre exemple : vous ne devez jamais publier des mots de passe, jetons d'accès/API, clés de chiffrement et autres données sensibles dans des repositories accessibles publics.

Les données sensibles ne doivent jamais être utilisées ou stockées dans des systèmes hors production. Bien entendu, les équipes peuvent avoir besoin d'importantes quantités de données imitant fidèlement la réalité à des fins de test. À ce titre, certains ingénieurs sont tentés d'« actualiser » les environnements de développement et de test avec les données de production.

EMPÊCHEZ QUE DES DONNÉES SENSIBLES PARVIENNENT AUX REPOSITORIES À L'AIDE DE PRE-RECEIVE

Outre le fait de stocker séparément les données sensibles, vous pouvez empêcher la publication involontaire (ou volontaire) de données protégées dans des repositories grâce à des **pre-receive hooks**. Il s'agit de scripts simples s'exécutant dans un environnement isolé sur l'appliance GitHub. Ces scripts sont déclenchés avant de pousser le code sur GitHub pour examiner les commits, identifier les informations sensibles et empêcher leur ajout à vos repositories.

Après examen du code, les pre-receive hooks ne peuvent renvoyer qu'une de ces deux valeurs : succès ou échec. En cas d'échec, les développeurs voient un message les informant que leur commit n'a pas été enregistré, message auquel s'ajoute toute autre information que votre équipe a jugé utile d'inclure. Le code n'est validé dans le repository que si les hooks de pré-réception réussissent. Le résultat ? Aucun code indésirable n'est ajouté à vos repositories. Votre entreprise est ainsi à l'abri des violations de sécurité, des menaces de poursuite en responsabilité civile et des sanctions réglementaires.

Protégez le code source

EUne sécurisation efficace passe par le contrôle de l'accès aux informations sensibles et au code source des logiciels qui les gèrent. Les permissions granulaires vous permettent de protéger efficacement ces informations sans créer un environnement qui refuse l'accès de manière trop généralisée. GitHub peut renforcer vos stratégies d'accès grâce à des **branches protégées**. Elles contribuent à maintenir l'intégrité de votre code en limitant plusieurs fonctionnalités de Git aux branches que l'administrateur choisit de protéger. Par exemple, les administrateurs peuvent limiter le nombre de personnes autorisées à publier dans une branche à certains utilisateurs et équipes. Ils peuvent également désactiver les force pushes susceptibles de modifier ou de supprimer du code sur certaines branches.

QUELQUES AUTRES MESURES DE PROTECTION :

Empêchez la suppression d'une branche

Exigez des revues manuelles et vérifiables par une ou plusieurs personnes

Empêchez la fusion du code qui échoue aux tests automatisés et aux contrôles d'intégration continue (CI)

Définissez des « propriétaires de code » pour n'importe quelle partie d'une base de code et exigez son passage en revue

Quelques mesures à prendre pour protéger les informations les plus importantes de vos clients :

- Utilisez des connexions sécurisées partout (pas d'excuses !) et utilisez uniquemet des protocoles tels que SSH et SFTP pour transférer des données
- Ne publiez jamais de mots de passe, jetons d'accès/API, clés de chiffrement, etc. dans des repositories accessibles au public
- Utilisez des branches protégées et des pre-receive hooks pour tenir compte des erreurs humaines et empêcher les données sensibles d'atteindre l'environnement de production
- Utilisez une solution de gestion des identités pour limiter l'accès et s'assurer que seules les personnes autorisées peuvent accéder aux données sensibles

La Sécurisation de l'ensemble du processus de développement

Votre base de code et les processus de développement qui influencent la façon dont votre équipe travaille sont au centre de votre stratégie de sécurité à plusieurs niveaux. À mesure que les développeurs collaborent et apportent des changements, il est important de mettre en place certaines mesures de protection grâce à de bonnes pratiques et à des fonctionnalités GitHub qui garantissent la fiabilité du code transmis en production.

Intégrez les systèmes existants

Certains outils, dont les mainframes, restent au cœur des opérations des institutions financières, plus de 50 ans après leur première introduction. Ces systèmes qui offrent des capacités inégalées peuvent également poser des problèmes de sécurité uniques en leur genre. Il faut donc procéder à des évaluations régulières de tous les systèmes existants (dits « legacy ») et comparer le coût de leur mise à niveau aux failles de sécurité auxquels ils peuvent exposer l'entreprise. En effet, la perspective de remplacer un système legacy ne semble pas financièrement intéressante jusqu'à ce que l'on pense à la menace d'une sanction de plusieurs milliards de dollars ou d'un incident de sécurité venant anéantir la confiance du public.

BIEN CHOISIR VOTRE SYSTÈME MAINFRAME

L'écosystème des partenaires de GitHub comprend de nombreux fournisseurs qui fabriquent, vendent et réparent des systèmes mainframe. Par exemple, IBM a récemment annoncé des extensions pour Git, la technologie sur laquelle repose GitHub. Elles permettent d'appliquer les pratiques DevSecOps aux workflows de développement mainframe.

Automatisez pour éviter les bugs et les erreurs

Les organisations qui mettent en œuvre la gestion automatisée des dépendances, par exemple, réduisent de **60 %** les failles de sécurité dans les logiciels déployés par rapport à celles qui ne le font pas.

La supervision humaine est une composante essentielle d'une stratégie de sécurité efficace à plusieurs niveaux. Cependant, le fait de se reposer outre mesure sur la vigilance naturelle des collaborateurs peut comporter des risques : ils peuvent être fatiguent, se laisser distraire ou tout simplement commettre des erreurs. Afin de permettre à vos équipes de travailler là où elles apportent le plus de valeur, laissez les tâches critiques mais répétitives ou fastidieuses aux machines. Votre équipe peut exploiter entièrement **l'intégration et la livraison continues (CI/CD)** dans vos workflows GitHub.

Les outils de CI/CD testent et évaluent le code chaque fois qu'un commit est poussé vers un repository. GitHub peut ainsi évaluer le nouveau code par rapport au code existant en production pour s'assurer que les changements proposés fonctionnent comme prévu et ne créent pas de failles de sécurité. Ces tests peuvent également s'étendre à l'examen de l'ensemble du code dont dépend votre code (ses dépendances).

La mise en œuvre de l'automatisation n'est pas difficile et peut donner des résultats immédiats et mesurables. Dans une étude datant de 2017, l'IEEE révèle que les entreprises qui mettent en œuvre la gestion automatisée des dépendances, par exemple, réduisent de 60 % les failles de sécurité dans les logiciels déployés par rapport à celles qui ne le font pas.

INTÉGRATION DE LA CI/CD À GITHUB

L'intégration de la CI/CD est simple avec GitHub et elle ne se répercute pas sur les workflows existants des développeurs, sauf si un résultat de test automatisé requiert une intervention de leur part. Moins l'automatisation est intrusive pour les développeurs, plus elle a de chances d'être adoptée et plus ses bénéfices seront visibles à l'échelle de l'entreprise.

GitHub s'intègre à une variété d'outils de CI tels que Jenkins, Travis et CircleCI. Ceux-ci extraient automatiquement le code d'un repository GitHub chaque fois que le code y est poussé, exécutent des tests et renvoient un statut (succès ou échec) à GitHub. Ils renvoient également des notifications sur l'état de chaque test, afin que les développeurs puissent voir à quelle étape leur code échoue.

Votre équipe et vous pouvez décider s'il faut ou non que les résultats des tests de CI empêchent la fusion du code dans la base de code, ou s'il faut simplement alerter les développeurs sans prendre aucune mesure. Si des statuts de CI sont requis, la pull request ne peut être fusionnée qu'à la condition que toutes les tâches de CI requises soient terminées. Si vous choisissez d'empêcher la fusion, la pull request ne peut pas être fusionnée tant que le code ne passe pas les tests requis avec succès.

L'intégration d'outils et de workflows de sécurité

Des centaines d'outils s'intègrent à GitHub. Nombre d'entre eux peuvent vous aider à protéger votre code et vos données client.

- **CI** : des outils de CI, tels que Travis CI, CircleCI et AppVeyor, créent et testent automatiquement du code lorsque vous le transmettez à GitHub, empêchant ainsi le déploiement de bugs en production.
- **Reporting d'erreurs** : des outils comme Snyk, Sentry et Dependabot aident votre équipe à identifier, corriger et prévenir les failles de sécurité.
- **Qualité du code** : Code Climate et Codeacy automatisent les validations avec des contrôles de sécurité et de qualité pour prévenir les erreurs humaines et simplifier les processus de revue de code pour votre équipe.

Vous êtes prêts à découvrir les types d'outils que nous proposons à votre équipe ? Rendez-vous sur GitHub Marketplace (github.com/marketplace) ou sur github.com/works-with._

Imposez des validations manuelles du code

Automatiser, c'est important, mais la validation manuelle du code reste un volet crucial de votre stratégie de sécurité. Plus il y a de personnes pour évaluer une base de code, plus il est probable que les erreurs ou les failles seront détectées. Au-delà de l'aspect sécuritaire, les validations remplissent également une fonction indispensable : elles permettent de partager la connaissance et offrent des possibilités de formation et de mentorat aux développeurs, quel que soit leur niveau d'expérience. Des validations de code régulières en conformité avec la politique de l'entreprise permettent d'obtenir une base de code non seulement plus fiable, mais également plus saine et cohérente.

GitHub fournit un framework flexible pour rendre obligatoires les validations du code. Vous pouvez :

Exiger des validations d'un ou de plusieurs utilisateurs ayant un accès en écriture au repository contenant les changements.

Exiger une validation et une approbation supplémentaires par un ou plusieurs propriétaires de code. Les **propriétaires de code** peuvent être des utilisateurs ou des équipes ayant la responsabilité d'une section du code, d'une technologie particulière ou d'une combinaison des deux.

Permettre aux personnes qui valident de faire des commentaires détaillés, voire ligne de code par ligne de code, sur chaque changement proposé.

Vous assurer que toutes les validations sont enregistrées et contrôlables à l'avenir.

Trois étapes pour aider votre équipe à éviter les bugs et les erreurs :

1. Développez une analyse de sécurité automatisée, afin de réduire l'erreur humaine et rechercher les vulnérabilités qui ne pourraient pas être introduites autrement et faire gagner du temps aux développeurs.
2. Intégrez l'automatisation directement dans le workflow GitHub de votre équipe avec des vérifications d'état.
3. Assurez-vous que tous les membres de votre équipe se sentent responsables à part égale du développement et de la maintenance de logiciels sécurisés.

Audit et conformité de votre workflow

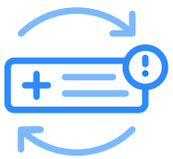
Les normes réglementaires en vertu desquelles les institutions financières fonctionnent de nos jours exigent une solide capacité de journalisation et d'audit. Les équipes doivent non seulement élaborer une stratégie de sécurité efficace, mais aussi prouver auprès des régulateurs qu'elles l'ont mise en place. GitHub fournit des frameworks de journalisation, d'audit et de reporting flexibles et puissants pour vous aider à assurer votre conformité et prouver la mise en application de cette politique.

JOURNALISATION DES AUDITS :

Enterprise Server conserve des logs complets de l'activité des utilisateurs et des systèmes. L'activité auditée inclut chaque opération « git push », y compris l'identité de l'initiateur de l'opération, son adresse IP et les repositories concernés par l'opération. Les logs peuvent être transférés vers un système externe (comme Logstash ou Splunk) à des fins d'analyse, de reporting et de stockage, garantissant ainsi le respect des exigences réglementaires.

ARCHIVAGE DES REPOSITORIES :

les repositories qui ne sont plus mis à jour peuvent être archivés et définis en mode de lecture seule. Ainsi, aucun nouveau code ne peut y être validé, mais les utilisateurs peuvent continuer de visualiser son contenu.



BLOCAGE DES FORCE PUSHES :

Git permet aux développeurs de réécrire l'historique des validations en imposant les changements par push. Bien qu'elle soit parfois nécessaire pour corriger les erreurs, cette fonctionnalité peut poser probléme si la réglementation exige que les données soient immuables. Enterprise Server permet aux administrateurs de bloquer les force pushes de repositories individuels ou de l'appliance entière afin de garantir que l'historique des validations n'est jamais réécrit.

INTERDICTION POUR CERTAINS UTILISATEURS DE SUPPRIMER LES REPOSITORIES :

la possibilité de supprimer des référentiels peut être limitée aux administrateurs.

Pensez sécurité en amont

Par le passé, la sécurité a trop souvent été une réflexion après coup. En fait, les protocoles qui régissent Internet n'ont pas du tout été conçus dans une optique de sécurité. Certes, un mécanisme de sécurité mis en œuvre après coup sera toujours bon à prendre, mais ce sera loin d'être suffisant dans le contexte de menace effrénée et constante qui pèse aujourd'hui. Pour être efficace, la sécurité doit être intégrée au projet de développement logiciel en amont. Il faut donc mener cette réflexion autour des aspects relatifs à la sécurité dès le début de votre conception, et en faire une priorité. La sécurité restera ainsi au premier plan tout au long du cycle de vie du produit.

La participation en amont des équipes de sécurité peut également influencer les décisions de conception, de développement et de maintenance avant qu'elles ne deviennent trop difficiles et trop coûteuses à changer. Ce conseil est assez rudimentaire, mais le processus importe peu si les équipes dirigeantes et techniques ne sont pas de la partie. La démarche de développement sécurisé exige que chacun contribue à une culture axée sur l'importance d'une sécurité efficace.

INSTAURATION D'UNE CULTURE AXÉE SUR LA SÉCURITÉ

L'approche DevSecOps permet de mettre en œuvre la sécurité tout au long du cycle de développement et de répartir la responsabilité entre les équipes et les rôles. Il est désormais anormal que les équipes de sécurité prennent part au processus en aval du développement d'une application pour se rendre compte qu'il existe déjà des failles exploitables. Si les experts en sécurité sont étroitement impliqués dès le début, les équipes peuvent établir des processus collaboratifs prenant en charge de manière proactive la sécurité, au fil du développement.

Les principes DevSecOps obligent parfois les entreprises à instaurer une toute autre culture au sein de leurs équipes en plus de transformer leur infrastructure. Les résultats obtenus en matière de fiabilité et de stabilité des logiciels en valent la peine. Certaines entreprises intègrent même des spécialistes en sécurité aux équipes Scrum pour s'assurer que la sécurité reste une priorité tout au long du processus.

Témoignage client : Société Générale

Société Générale est une multinationale de services bancaires et financiers qui emploie 146 000 personnes dans 66 pays. Global Banking & Investor Solutions (GBIS) est l'une des trois activités principales de la banque. Elle regroupe les activités de banque de financement et d'investissement, de gestion d'actifs, de banque privée et de services titres.

Sous la direction de Carlos Goncalves, responsable des technologies de l'information, GBIS s'est lancé dans une refonte majeure de ses systèmes d'information, dans l'optique d'y déployer la livraison continue (continuous delivery). En 2014, le service Systèmes d'information décide de déployer Enterprise Server sur sa plate-forme de livraison continue. Il a depuis été adopté par plus de 3 000 collaborateurs dans le monde entier.

« Notre transformation numérique est une priorité stratégique pour la banque [...]. Nous sommes convaincus qu'en adoptant des outils et pratiques que les plus grands éditeurs de logiciels du monde mettent déjà en œuvre, nous garantissons notre réussite ! »

AMIR JABALLAH,
Global Head of Continuous Delivery Platform,
Société Générale

« Les frais administratifs ont baissé de 80 % par rapport à la précédente solution de Société Générale. [Enterprise Server] nous a permis de gagner du temps, ce qui nous a permis d'accompganer le changement auprès des équipes concernées. »

AMIR JABALLAH

Pour atteindre son objectif stratégique de transformation de 50 % des applications logicielles GBIS d'ici à la fin de 2016, le service IT de GBIS avait besoin d'une plate-forme de contrôle des versions rapide et hautement performante, parfaitement compatible avec ses systèmes

d'intégration et de test. Pour offrir à ses développeurs un niveau élevé de fonctionnalités et de performances, la banque est passée d'un système de contrôle des versions centralisé à Enterprise Server, la meilleure solution pour répondre à ces besoins.

Cinq des fonctionnalités clés de GitHub qui ont convaincu Société Générale :

1. **Repositories :** les développeurs peuvent créer leurs propres repositories et travailler en équipe, ce qui réduit considérablement les délais de livraison des projets.
2. **Revue du code :** les développeurs peuvent valider et proposer des changements au code développé par leurs collègues et s'assurer d'éliminer toutes les erreurs.
3. **Collaboration :** les développeurs peuvent également rechercher et réutiliser le code existant dans l'entreprise au lieu de réinventer la roue lorsqu'ils disposent de solution développées pour un autre projet et déjà testées sur le terrain.
4. **Gestion des versions de la documentation :** les développeurs peuvent créer et héberger une documentation au plus près des applications.
5. **Sécurité :** toute la plateforme est hébergée sur des serveurs internes. GitHub utilise l'annuaire d'entreprise pour gérer l'authentification des utilisateurs et enregistre toutes les activités des utilisateurs et du système.

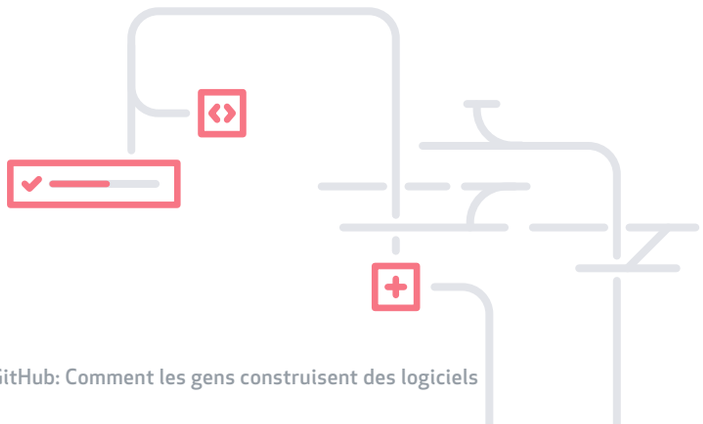
Résumé

La sécurisation des données n'a jamais été aussi essentielle à la réussite, voire à la survie des entreprises. Mettre en œuvre une sécurité efficace ne devrait pas être une tâche ardue, .Une stratégie de défense sur plusieurs niveaux présente l'avantage de décomposer la démarche de sécurisation en plusieurs sous-projets plus simples à entreprendre. Le plus important est de commencer (ou de continuer) à tester, réviser, remettre en question et améliorer. Les attaques de sécurité ne vous laisseront jamais de répit et vous devez rester vigilants.

L'une des problématiques consiste à trouver un équilibre entre une sécurisation efficace et la liberté dont les développeurs ont besoin pour innover, s'épanouir et être productifs.

Trop peu de sécurité peut se révéler inefficace, alors que trop de sécurité risque d'étouffer la créativité voire de pousser les collaborateurs à contourner vos stratégies de sécurité. Avec des outils tels que GitHub, vous disposez de contrôles pour trouver cet équilibre dans votre organisation.

Au final, notre objectif est de vous aider à faire évoluer la perception de la sécurité d'une menace à une opportunité d'améliorer la satisfaction client, d'attirer de nouveaux clients et de vous différencier. Offrir une sécurité performante, c'est un élément clé pour gagner la confiance de vos clients. À ce titre, des partenaires comme GitHub vous accompagnent pour réussir la mise en place d'une stratégie de sécurité efficace.



Ce qu'il faut retenir:

1. Vos données font l'objet d'attaques constantes. Les conséquences d'un incident de sécurité peuvent être très lourdes autant dans le cadre de vos activités professionnels, mais aussi, à titre personnel.
2. Il n'existe aucune solution miracle pour vous mettre à l'abri. Une stratégie de défense à plusieurs niveaux prévoyant des outils, des processus et de bonnes pratiques est beaucoup plus fiable et performante.
3. Une sécurisation effiicace passe par une culture elle-même effcace autour de la sécurité. La sécurité doit être une priorité dans toutes les initiatives et décisions importantes.
4. Chiffrez tout, sans exception.
5. Suivez une approche en niveaux pour vérifier et valider toutes les données entrant dans vos systèmes afin de prévenir les attaques malveillantes de type SQL Injection.
6. Pensez à intégrer les systèmes plus anciens, mais stratégiques, tels que les mainframes. Tenez compte d'éventuels coûts liés aux sanctions, aux poursuites en responsabilité civile et à la perte de confiance du public, lors de l'évaluation et de la mise à niveau des outils existants.
7. Automatisez autant que vous le pouvez, notamment pour les tests de logiciels.

Nous sommes là pour vous aider

Ceci n'est qu'une des manières dont GitHub soutient le développement de code sécurisé et robuste.

Vous souhaitez nous contacter ?
sales@github.com

GitHub