ATTAQUE DE LA SUPPLY CHAIN : LE TOURNANT SOLARWINDS

digicert®

Sommaire

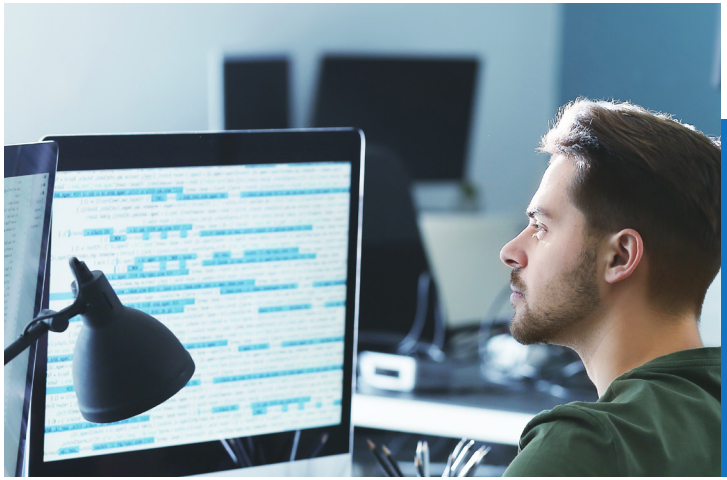
Le cybercrime de l'Orion Express

Le 13 décembre 2020, le gouvernement des États-Unis annonce que des hackers sont parvenus à s'infiltrer dans les réseaux de plusieurs agences fédérales. Les semaines suivantes, le monde découvre non sans stupeur que cette attaque a permis aux pirates – très probablement des agents russes – de dérober des données sensibles pendant des mois, sans éveiller le moindre soupçon. Au fur et à mesure des révélations, l'ampleur de la compromission ne fait plus aucun doute. Loin d'être une affaire classique d'espionnage, cette action représente l'une des plus gigantesques intrusions par malware jamais réalisées[1].

Cible de l'attaque, le logiciel Orion est un programme de gestion IT développé et commercialisé par SolarWinds, une entreprise de cybersécurité basée au Texas. Si son nom s'apparente désormais à la compromission dont elle a été victime, SolarWinds n'est pas pour autant la première entreprise à faire face à ce genre d'intrusion, comme le traitement médiatique de l'affaire a pu le laisser entendre.

Bien avant le lancement de l'attaque à proprement parler, les hackers avaient tout d'abord inséré un code malveillant dans le logiciel Orion. Une fois le malware intégré au pipeline CI/CD, le ver était dans le fruit. Dès lors, les pirates n'avaient plus qu'à attendre le moment propice pour passer à l'action, sachant que le temps jouait en leur faveur. Au cours des mois suivants, le malware s'est répandu comme une traînée de poudre dans le monde entier, à chaque installation du logiciel Orion sur les serveurs d'agences fédérales, de grandes entreprises et même d'autres éditeurs de logiciels.

Puis, un beau jour, les pirates sont passés à l'action. Mais une fois l'infiltration découverte, il était déjà trop tard – d'une part à cause des dégâts déjà occasionnés par le malware, et d'autre part du fait de l'omniprésence du logiciel Orion sur les systèmes de toute la planète. La priorité n'était plus de limiter l'impact d'une compromission, d'isoler un serveur ou d'appliquer un correctif. L'envergure de l'attaque était telle que personne ne pouvait – et ne pourra probablement jamais – évaluer avec précision le nombre de systèmes, d'entreprises et d'organismes touchés.



[1] https://www.reuters.com/article/us-cyber-solarwinds-microsoft/solarwinds-hack-was-largest-and-most-sophisticated-attack-ever-microsoft-president-idUSKBN2AF03R

SolarWinds – L'avant et l'après

Ce qui rend l'attaque SolarWinds si impressionnante, c'est son échelle : Orion, le logiciel compromis, a été vendu à plus de 33 000 clients. Sunburst, le code malveillant, a quant à lui été téléchargé par quelque 18 000 entreprises. Ce cheval de Troie a ainsi pu rester tapi pendant des mois sur le réseau de milliers d'entreprises et d'administrations à leur insu.

Par la suite, les expertises réalisées ont découvert que Sunburst avait touché, rien qu'aux États-Unis, 425 sociétés du Fortune 500, les dix plus grands opérateurs télécoms, cinq des principaux cabinets comptables et même Microsoft. Toujours outre-Atlantique, des agents étrangers ont pu accéder aux réseaux de la Maison-Blanche, des cinq forces de l'Armée, du Département du Trésor, du Pentagone, du Département d'État, du Département de la Justice, du Département de la Sécurité intérieure, de l'Agence de sécurité nationale (NSA) et de l'Administration nationale de la sûreté nucléaire (NNSA). Excusez du peu. Ailleurs dans le monde, Sunburst s'est notamment attaqué à l'OTAN, au gouvernement britannique et au Parlement européen. Les hackers se sont infiltrés dans les réseaux d'entreprises et d'organismes publics et privés de toutes tailles, le tout sous couvert d'un logiciel signé et considéré comme sûr.

Il y a à peine 10 ans, nombre d'experts en cybersécurité considéraient comme hautement improbable l'attaque d'une supply chain. Selon eux, ce type d'intrusion ne cadrait pas avec les vulnérabilités que cherchaient à exploiter les cybercriminels. Et devant l'impossibilité pour un particulier ou une entreprise de protéger chaque étape du développement de la supply chain, entreprendre une telle action leur semblait de toute façon futile. Seule voix discordante à l'époque, Stacy Simpson, membre de l'association SAFECode, avait alors corédigé avec des leaders du secteur un rapport invitant les professionnels DevOps à ignorer ces recommandations et à implémenter des bonnes pratiques de signature de code pour verrouiller les points d'intrusion dans la supply chain CI/CD[2].

Mais son appel n'a pas été entendu : pour la plupart des acteurs du secteur, une attaque de supply chain était tout simplement inenvisageable – et de toute façon impossible à contrer. L'affaire SolarWinds a radicalement changé la donne.

Chronique d'une catastrophe annoncée

Le type de vulnérabilité qui a permis aux cybercriminels de cibler SolarWinds n'est pas nouveau. Depuis l'avènement du pipeline CI/CD, les infections par malware sont une préoccupation majeure des entreprises pour qui la sécurité est une priorité. Mais c'est seulement avec SolarWinds que la planète entière a découvert l'ampleur potentielle de ce type d'attaque pourtant tout à fait évitable.

[2] http://safecode.org/publication/SAFECode_Software_Integrity_Controls0610.pdf

À l'instar de nombreuses entreprises, SolarWinds a signé son code pour attester de la légitimité de son logiciel appelé Orion. En signant son logiciel, SolarWinds s'est aussi engagé à informer ses clients de toute modification ou de toute présence de malware après le développement et la sortie du logiciel. Mais avec la compromission d'Orion, la refonte complète du processus de supply chain logicielle s'est imposée comme une nécessité absolue. La question centrale était de savoir comment protéger les clients et utilisateurs en cas d'injection d'un malware dans le code avant sa signature.

L'attaque SolarWinds a clairement montré qu'une signature de code – technologie pourtant éprouvée – ne permettait pas, à elle seule, de protéger la supply chain. Outre la signature du code, la sécurisation d'un logiciel passe en effet par l'analyse du code source, l'analyse des bibliothèques tierces, le contrôle des collaborateurs et la mise en place d'un environnement de développement sécurisé. Miser uniquement sur la signature de code, c'est un peu comme s'équiper d'un système de sécurité domestique que l'on active une heure de temps à autre.

Avec l'affaire SolarWinds, le monde a découvert qu'il existait de nouveaux vecteurs d'attaque des supply chains, mais aussi que la signature d'un logiciel ne devait pas se limiter à la dernière étape du cycle DevOps. De fait, c'est tout le processus qui doit être sécurisé, contrôlé et validé. Sans sécurité intégrale, impossible de boucler la boucle DevOps.

La signature de code, un impératif à chaque étape du développement

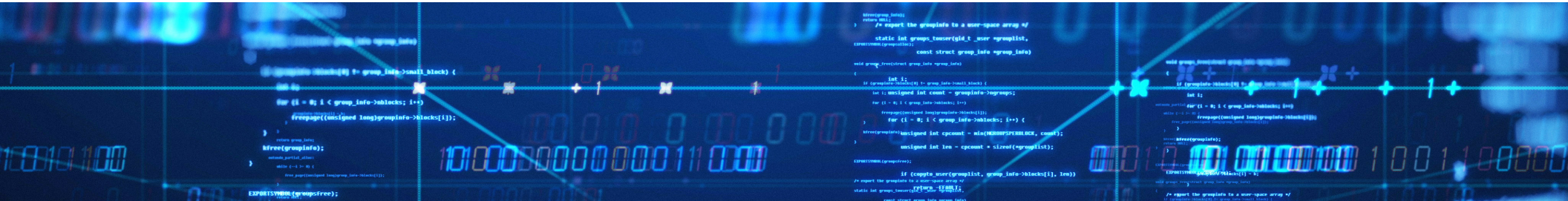
Une fois signé, le logiciel est protégé contre les modifications. Mais le chiffrement n'en est pas la seule composante. La signature de code permet de contrôler l'identité de l'éditeur, puis de vérifier l'intégrité du code après le téléchargement. L'utilisateur a ainsi la garantie que le logiciel n'a pas été modifié.

Aux yeux de l'utilisateur, un logiciel signé est un logiciel digne de confiance. Il est donc crucial que les équipes DevOps proposent des versions logicielles garanties sans malware. Lorsqu'il contient un code malveillant, le logiciel signé contourne tous les contrôles de signature de code présents dans les écosystèmes de distribution de logiciels publics et privés. Signer un logiciel dont le code est compromis, c'est donner pour directives aux passerelles de sécurité de ne pas rechercher les éventuels failles sur tout le reste de la supply chain et au point d'accès utilisateur.

Or, la seule manière de mettre en confiance l'utilisateur est de protéger l'intégrité du logiciel tout au long du processus CI/CD. Il ne suffit pas de signer le logiciel à la fin du cycle de développement. Vous devez rechercher la présence de malware ou de toute menace dans le code écrit par les autres intervenants de la supply chain, y compris au niveau du code open-source et du code de bibliothèques tierces. La création du logiciel nécessite l'utilisation d'un code « propre », qu'il faut ensuite signer à l'aide de clés de signature adéquates, émises, gérées et stockées dans les règles de l'art. L'accès à ces clés de signature doit être limité aux seuls utilisateurs ou serveurs de développement autorisés.

Craignant que ces pratiques ne soient trop contraignantes, les développeurs ont souvent tendance à prendre des raccourcis ou d'ignorer certaines précautions d'usage pour avancer dans leur développement. Chacun le sait : la vitesse et l'efficacité font partie des valeurs cardinales du DevOps. Toute étape fastidieuse peut entraîner des retards qui paralysent le processus CI/CD. Autrement dit, toute solution peu agile – c'est-à-dire qui crée des retards et des interruptions – n'est pas une bonne solution.

Dans bien des cas, il est possible d'automatiser ces processus de signature pour limiter les interventions humaines et permettre aux développeurs de se concentrer entièrement sur leur code, sans pour autant faire l'impasse sur les questions de sécurité.



Bonnes pratiques de signature de code : un impératif incontournable

En tant que telle, la signature d'un logiciel est indispensable. Mais la valeur de cette signature ne vaut que par la gestion des politiques et pratiques qui la sous-tendent. En effet, c'est à ce niveau que des failles de sécurité peuvent apparaître. En matière de sécurité logicielle, le choix est vite fait : pour éviter d'exposer votre supply chain aux attaques, vous devez mettre en place des pratiques adaptées.

En agissant ainsi, vous évitez de faire de la signature de code le maillon faible de votre supply chain. Correctement implémentée, la signature de code peut également stopper les infections par malware.

1. Protégez et contrôlez l'utilisation des clés de signature et des clés privées

Dupliquer et stocker des clés privées dans plusieurs emplacements est tout à fait possible. Avec les clés physiques telles que les jetons USB, il existe un risque de perte ou d'oubli dans des endroits accessibles à tous. Quant aux collaborateurs négligents ou mécontents, ils peuvent égarer voire dérober des clés.

La protection des clés privées peut s'effectuer de deux manières. Soit en les stockant dans un module HSM correctement installé et mis à jour, soit en confiant cette mission à un service PKI managé.

2. Appliquez une authentification multifacteur et un stockage conforme à la norme FIPS

Oubli ou compromission de mot de passe ; partage, perte ou vol de clés ; gestion trop laxiste des jetons USB... les failles de sécurité sont nombreuses.

Pour mieux vous protéger face aux usages imprudents ou malveillants, vous devez mettre en place des contrôles d'accès renforcés. Avec l'authentification multifacteur, les utilisateurs doivent prouver leur identité au moyen de deux types d'identifiants. Les modules HSM assurent une authentification forte tout en résistant aux tentatives d'effraction physique. Résultat, seuls les utilisateurs autorisés peuvent accéder aux clés sécurisées.

3. Opérez avec une bonne visibilité

Sans audit complet et sans responsabilisation des utilisateurs, il est difficile, voire impossible de tracer les usages abusifs de clés ou les anomalies de signature.

Tout comme les utilisateurs, les clés doivent impérativement être authentifiées avant toute signature de code. L'une comme l'autre de ces deux actions devraient être réalisées à chaque phase du processus CI/CD. En effectuant un traçage et un audit des signatures, vous contribuez à responsabiliser les utilisateurs tout en sachant qui signe quoi. L'utilisation de clés de signature par des utilisateurs autorisés, mais sur des créneaux non autorisés, peut mettre la puce à l'oreille des équipes de sécurité IT avant que le logiciel ne progresse vers l'étape suivante du pipeline.

4. Contrôlez la sécurité au niveau des comptes

L'absence de contrôle de permission rend difficile, sinon impossible, la protection et le suivi des clés et des signatures.

L'attribution des clés s'effectuera donc en fonction de permissions et de rôles, avec une séparation par produit, projet ou équipe. Les administrateurs devront pouvoir générer des clés et des certificats privés, puis distribuer les clés aux équipes. De même, l'accès aux clés devra être restreint aux seuls utilisateurs affectés aux workflows comprenant des signatures. Enfin, l'attribution de clés aux composants non humains, tels que les serveurs de développement, permettra quant à elle de favoriser l'automatisation.

5. Contrôlez la sécurité au niveau de l'entreprise

Les politiques organisationnelles jouent un rôle tout aussi important que les permissions et rôles individuels. Sans une structure de politique et de contrôle globale, le manque d'alignement potentiel des processus de sécurité individuels risque d'engendrer des vulnérabilités.

Créez et appliquez des politiques de sécurité internes à tous les niveaux, de l'entreprise dans sa globalité jusqu'au développement des produits. Celles-ci devraient spécifier les algorithmes ou la taille des clés cryptographiques, les types et les durées de validité de certificats et les workflows d'approbation.

6. Modifiez vos clés

Lorsque l'on utilise une clé unique pour la signature de multiples sections de code, la compromission d'une seule section met en danger tout le reste.

Pour éviter qu'une intrusion isolée ne se transforme en attaque d'envergure, vous devez pouvoir émettre rapidement et facilement de multiples clés, puis mettre en place un système de rotation régulière.

7. Définissez des politiques attribuant certains types de clés à certains projets

Sans un contrôle renforcé des différents types de clés, les entreprises peuvent manquer de visibilité sur les accès et les usages, ouvrant ainsi la porte à toutes sortes d'abus.

Réutilisez des clés de signature spécifiques pour des produits ou des projets particuliers dont l'accès est limité à des équipes et des utilisateurs bien précis. Dans la mesure du possible, utilisez des clés et des certificats privés à la volée, de sorte que chaque version logicielle soit signée par une clé et un certificat privé unique.

8. Ne laissez jamais un logiciel non signé quitter votre environnement

Le transfert de fichiers entiers pour la signature de code est une opération lente et gourmande en ressources qui, de surcroît, peut permettre aux hackers d'intercepter et de modifier le code lors du transit.

La signature de hachage permet d'éviter ce genre de risques. Lors de la signature, seul le hachage est chargé dans le cloud. Résultat, votre processus est presque aussi rapide qu'avec une signature en local. Le code lui-même reste sur vos serveurs internes, ce qui élimine tout risque de vol ou d'altération de votre propriété intellectuelle.

9. Contrôlez régulièrement votre propriété intellectuelle

Cette étape simple n'en est pas moins indispensable. Car pour les utilisateurs, signature de code rime avec sécurité, et donc confiance. Or un développeur peut tout à fait envoyer un code infecté par un virus ou un malware et présenté comme sécurisé.

Pour ne pas en arriver là, analysez le code source et le code de bibliothèques tierces, mais aussi le code compilé. Effectuez une analyse anti-malware et ne signez votre code qu'une fois que vous avez la certitude que votre logiciel est parfaitement propre.

10. Assurez la reproductibilité de vos signatures pour valider ce que vous signez

Faute de pouvoir comparer les développements à une base de référence, il est impossible de savoir si les fichiers binaires sont identiques et s'ils peuvent être reproduits par une majorité d'utilisateurs. Cette situation peut favoriser l'injection de malwares indétectables.

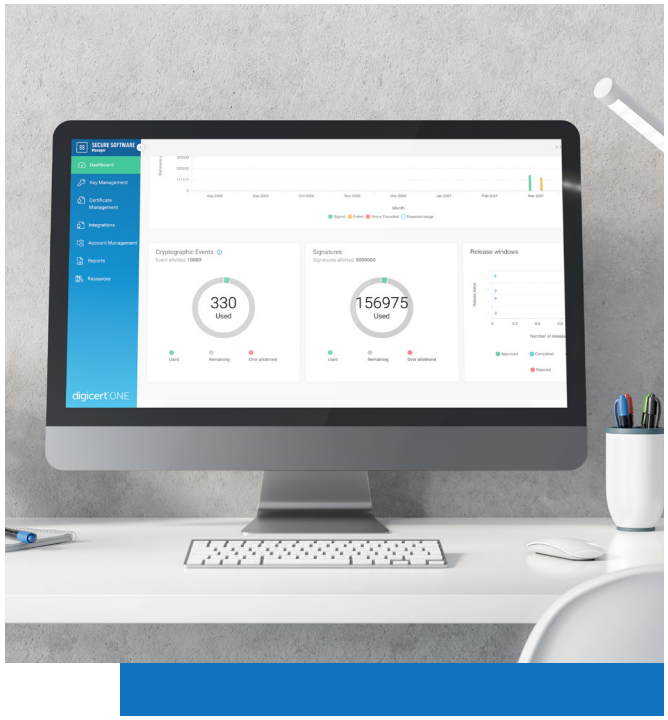
Avant la signature, comparez les hachages des fichiers binaires envoyés pour signature à une base de référence. Vous aurez ainsi la confirmation que la version de production se comporte de façon conforme aux attentes formulées lors des cycles de tests et d'assurance qualité.

La signature continue, une réponse adaptée à

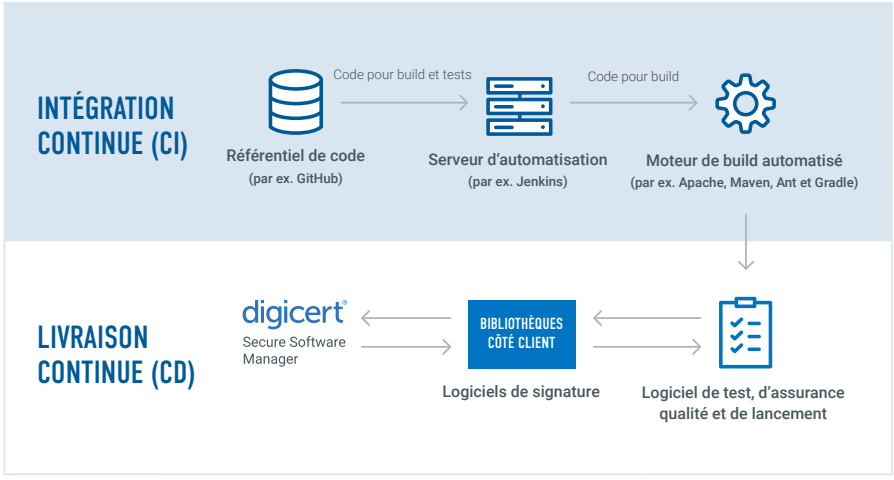
des délais serrés

DigiCert Secure Software Manager a été développé pour renforcer l'agilité du DevOps. Cette solution s'appuie sur des processus automatisés pour assurer une signature continue et soumise à des contrôles de sécurité de bout en bout, sans étape supplémentaire ni perte de temps inutile. Elle apporte ainsi une garantie de confiance absolue.

Secure Software Manager effectue une signature simple, rapide et à grande échelle du code, du logiciel et de tous les fichiers binaires. Vous pouvez créer des processus de développement reproductibles permettant d'automatiser certains paramètres de base, ce qui vous permet de détecter toute injection de malware avant la sortie du code compromis. Vous protégez la gestion des clés, contrôlez les accès en fonction des permissions et utilisez les fonctionnalités de reporting et de traçage pour améliorer votre visibilité et responsabiliser les utilisateurs. Avec DigiCert Secure Software Manager, vous disposez d'une solution unique pour déployer et pérenniser vos bonnes pratiques de signature de code tout au long du processus CI/CD.



Signature logicielle automatisée tout au long du pipeline CI/CD avec DigiCert® Secure Software Manager



Maintenant que nous savons, agissons

Lorsque, en 2010, Stacy Simpson publie un rapport SAFECode dans lequel elle fait de la protection de la supply chain CI/CD une priorité, elle ne sait pas qu'elle décrit ce qui inspirera dix ans plus tard l'une des cyberattaques les plus impressionnantes jamais recensées. Mais à l'époque, peu de gens croient en la possibilité d'une attaque de la supply chain, perçue comme une intrusion bénigne. C'était avant que l'affaire SolarWinds ne vienne tout chambouler.

Dès 2010, le rapport SAFECode expliquait pourtant comment faire face aux menaces visant les supply chains CI/CD. Comme les équipes DevOps et les entreprises ne peuvent contrôler toutes les parties de la supply chain, la seule manière de lutter contre les attaques potentielles est de mettre en place de bonnes pratiques permettant de surveiller l'intégrité du code pendant toute sa phase de développement. Pour protéger les maillons faibles de la supply chain, la solution consiste à imposer à toutes les parties prenantes d'analyser et de signer leur code tout au long du processus de développement.

À l'époque du rapport SAFECode, cette approche pouvait sembler impossible à envisager. La mise en place de pratiques de sécurité draconiennes pouvait en effet ralentir, voire interrompre le processus CI/CD, ce qui s'avérait tout aussi dommageable que le risque de compromission en tant que tel.

Avec l'automatisation des processus de signature de code, les choses ont bien changé depuis. Grâce à l'automatisation, les équipes DevOps disposent de tous les outils nécessaires pour réduire la surface d'attaque de l'environnement CI/CD. La mise en place d'une signature continue leur permet de protéger le code, les logiciels et les applications en se reposant sur un système d'intégrité et de cryptage complet, avec une intervention humaine réduite au strict minimum.

Le succès de l'attaque SolarWinds risque de faire des émules parmi les cybercriminels. Face aux menaces planant sur les supply chains, une partie de la solution passe par le déploiement de bonnes pratiques de signature de code. En surveillant et en sécurisant leur code, les différents acteurs de la supply chain contribueront à protéger tous les maillons de la chaîne lors du développement et du franchissement des étapes successives du pipeline CI/CD.

Bien plus qu'un acte de cyberespionnage d'une ampleur inédite, l'affaire SolarWinds marque un nouveau tournant dans l'histoire du développement de logiciels et de code. Aujourd'hui, nous ne pouvons plus dire que nous ne savions pas. Nous connaissons les risques potentiels après avoir constaté l'étendue des dégâts. Mais heureusement, nous disposons d'outils pour nous protéger rapidement et efficacement contre ce type de menace. L'heure est venue de mettre à profit ces outils et d'implémenter des bonnes pratiques de signature de code pour protéger la supply chain DevOps de A à Z.