LES ATTAQUES TEAMPCP : VOTRE PIPELINE CI/CD EST LA NOUVELLE LIGNE DE DÉFENSE

Les attaques TeamPCP : Quand votre pipeline CI/CD devient la nouvelle ligne de front de la cybersécurité

Nous avons collectivement bâti l'édifice moderne de la production logicielle sur une prémisse qui, avec le recul, semble d'une naïveté déconcertante : celle que les systèmes et les innombrables dépendances sur lesquelles nous nous appuyons étaient intrinsèquement dignes de confiance. Cette foi aveugle, souvent ancrée dans l'urgence du déploiement et la complexité croissante de nos infrastructures, a créé une vulnérabilité systémique que les acteurs malveillants ont su identifier et exploiter avec une efficacité redoutable, transformant nos processus de développement en de véritables vecteurs d'attaque. C'est une erreur fondamentale de jugement qui nous coûte cher aujourd'hui et continuera de le faire si nous ne réévaluons pas nos stratégies de sécurité.

Les attaquants, loin d'être novices, ont depuis longtemps compris que la voie la plus rapide et la plus efficace pour propager des logiciels malveillants à grande échelle ne consiste pas à cibler directement nos environnements de production, pourtant souvent robustes et surveillés. Leur stratégie s'est orientée vers la compromission des canaux mêmes qui construisent, testent et déploient nos logiciels : les fameux pipelines d'intégration et de livraison continues. Les récentes attaques menées par TeamPCP ne sont pas des incidents isolés, mais plutôt une démonstration éclatante

de cette nouvelle réalité, un signal d'alarme retentissant qui devrait nous faire prendre conscience de l'urgence de la situation.

Imaginez un instant : des identifiants dérobés permettent la publication de versions vérolées de Trivy, un scanner de vulnérabilités pourtant conçu pour la sécurité et de ses actions GitHub associées. Peu après, une attaque similaire frappe LiteLLM, injectant des charges utiles dérobant des identifiants et touchant des millions de développeurs via PyPI. Enfin, le package Python Telnyx, téléchargé près de 790 000 fois par mois, subit le même sort. Ces incidents, loin d'être anecdotiques, dessinent un playbook d'attaque clair et précis, une feuille de route pour les cybercriminels qui continueront d'exploiter nos faiblesses tant que nous traiterons nos systèmes de **CI/CD** comme des entités distinctes et moins critiques que nos environnements de production. Il est temps de changer notre paradigme de sécurité.

La Chaîne d'Approvisionnement Logicielle : Un Point Aveugle Critique

Les Pipelines CI/CD : Le Cœur Battant et Vulnérable

Considérez votre pipeline d'intégration et de déploiement continus, ou **CI/CD** (Continuous Integration/Continuous Delivery), comme le système nerveux central de votre processus de développement logiciel, l'équivalent numérique de la chaîne de montage automatisée d'une usine. C'est là que le code est fusionné, testé, packagé, puis préparé pour le déploiement. Il exécute des tâches critiques, accédant à une multitude de ressources, de vos dépôts de code source à vos plateformes de déploiement en passant par vos outils de gestion des **secrets** et vos clés de signature. C'est un environnement où la vitesse et l'automatisation sont

reines, mais cette même efficacité le rend incroyablement attrayant pour les attaquants.

Sur le terrain, ces pipelines jouissent d'un accès privilégié à des informations et des systèmes de haute valeur. Ils détiennent souvent les clés du royaume, avec des accès aux identifiants cloud, aux clés de signature de code et aux systèmes de déploiement qui permettent la mise en production de votre logiciel. De plus, ils exécutent couramment du code tiers, souvent intégré via des dépendances open source ou des actions préexistantes, sans toujours un niveau de contrôle ou de révision aussi rigoureux qu'il le faudrait. Cette combinaison explosive de privilèges élevés et de confiance implicite dans du code externe crée une surface d'attaque idéale, une véritable aubaine pour quiconque cherche à s'introduire discrètement dans votre écosystème.

L'impact d'une compromission est potentiellement dévastateur, car le rayon d'action d'une attaque réussie sur un pipeline **CI/CD** est colossal. Une seule version malveillante injectée dans votre **chaîne d'approvisionnement logicielle** peut se propager instantanément à des milliers d'organisations en quelques heures seulement, contaminant leurs propres systèmes et leurs utilisateurs finaux. Nous l'avons vu par le passé avec la compromission de l'action `tj-actions/changed-files`, exposant des **secrets** sur plus de 23 000 dépôts et plus récemment avec les incidents TeamPCP, où Trivy, LiteLLM et Telnyx ont été ciblés. Ces exemples ne sont que la pointe de l'iceberg et illustrent la capacité de ces attaques à créer un effet domino à l'échelle industrielle, menaçant l'intégrité de tout l'écosystème logiciel.

L'Illusion de la Confiance Implicite

La racine de bon nombre de nos problèmes de sécurité réside dans cette notion de confiance implicite, une sorte de confiance aveugle par défaut que nous accordons

à chaque maillon de notre **chaîne d'approvisionnement logicielle**. C'est comme laisser la porte de votre maison grande ouverte sous prétexte que le quartier est généralement sûr, ou confier les clés de votre usine à un inconnu en espérant qu'il n'en fera rien de mal. Par défaut, les outils **CI/CD** sont souvent configurés avec des niveaux de sécurité qui privilégient la facilité d'utilisation et la rapidité au détriment d'une posture de sécurité robuste, créant ainsi des vulnérabilités structurelles que les attaquants exploitent sans effort.

Dans la pratique, cela se traduit par des configurations où les contrôles sont faibles, les permissions sont souvent trop permissives et la confiance est accordée sans vérification. Par exemple, l'exécution de scripts ou d'actions tierces au sein de nos pipelines sans une analyse approfondie de leur contenu ou de leurs permissions est monnaie courante. Ce n'est pas tant que les équipes de développement commettent des erreurs volontaires, mais plutôt que les valeurs par défaut des outils **CI/CD** incitent à une approche moins sécurisée. Cette facilité de compromission a d'ailleurs été brillamment démontrée par des entités comme Hackerbot-Claw, un bot IA autonome, prouvant à quel point ces flux de travail peuvent être exploités aisément, même avec des méthodes peu sophistiquées.

L'impact direct de cette illusion de confiance est une **surface d'attaque** démesurément large et facile à exploiter. Les attaques TeamPCP, comme beaucoup d'autres, n'étaient pas d'une sophistication technique extraordinaire ; le seuil d'entrée pour les attaquants est tout simplement bas. En l'absence de contrôles stricts et d'une vérification constante, n'importe quel élément de votre pipeline, de la moindre dépendance à une action GitHub, peut devenir un point d'entrée pour des acteurs malveillants. Ce manque de rigueur dans l'évaluation de la confiance, combiné à la rapidité de la **distribution de malware à grande échelle** permise par les **pipelines de déploiement**, fait de chaque composant non vérifié un risque existentiel

pour l'intégrité de l'ensemble de votre système.

Quand le Système Fonctionne Trop Bien

Le paradoxe de nos systèmes **CI/CD** est qu'ils excellent dans ce pour quoi ils ont été conçus : automatiser, accélérer et fluidifier le processus de livraison logicielle. C'est une machine parfaitement huilée pour la production, mais cette même efficacité se retourne contre nous lorsqu'il s'agit de sécurité. Imaginez une autoroute ultra-rapide sans péages ni contrôles ; elle excelle à transporter des véhicules, mais aussi tout ce qu'ils contiennent, qu'il s'agisse de biens légitimes ou de contrebande. Nos pipelines, optimisés pour la vitesse, permettent ainsi une propagation tout aussi rapide du code, qu'il soit bienveillant ou malveillant.

Sur le terrain, cela signifie que la moindre faille, la plus petite brèche, est instantanément amplifiée. Un identifiant volé, une dépendance compromise, ou une action GitHub malveillante et c'est tout un écosystème qui est exposé. Le système, en fonctionnant comme prévu, c'est-à-dire en automatisant aveuglément la chaîne de confiance, permet à une attaque isolée de se transformer en une épidémie logicielle en un temps record. Les infrastructures basées sur des technologies comme **Kubernetes**, bien que robustes en production, sont particulièrement sensibles si les conteneurs et leurs images transitant par le **CI/CD** sont déjà compromis, car le malware pourrait alors s'exécuter à l'échelle sans entrave.

L'impact de ce fonctionnement trop bien est un effet d'entraînement exponentiel. Chaque compromission réussie permet de dérober de nouveaux identifiants, de nouvelles clés, qui sont ensuite utilisées pour compromettre encore plus de systèmes. Le rayon d'action de l'attaque ne cesse de croître, créant un cercle vicieux

où la confiance est érodée à chaque nouvelle victoire des attaquants. C'est ainsi que les attaques sur la **chaîne d'approvisionnement logicielle** prennent de l'ampleur, transformant des incidents isolés en menaces systémiques. Nous sommes les témoins des premières phases d'un phénomène qui va s'intensifier si nous n'agissons pas pour rompre ce cycle.

Armer Vos Pipelines : Stratégies de Durcissement Essentielles

Adopter l'Identité Fédérée et les Accès Éphémères

La première étape et sans doute la plus cruciale pour réhabiliter la sécurité de vos pipelines, consiste à se débarrasser des identifiants statiques. Ces jetons de longue durée, clés API permanentes ou mots de passe enregistrés sont de véritables bombes à retardement. Pensez-y comme à une clé principale que vous confiez à tout le monde : le jour où elle est perdue ou volée, c'est l'ensemble de votre sécurité qui est compromise pour une durée indéterminée. Nous devons partir du principe qu'ils seront volés, car l'histoire nous a montré qu'ils le seront inévitablement.

L'application de cette logique sur le terrain implique une migration vers des systèmes d'identité fédérée, notamment en exploitant **OIDC** (OpenID Connect). Cette approche permet de générer des identifiants de courte durée, des clés qui expirent après quelques minutes, limitant drastiquement la fenêtre d'opportunité pour un attaquant même s'il parvient à les dérober. De plus, ces accès doivent être strictement circonscrits, c'est-à-dire que chaque jeton ne doit avoir que les permissions minimales nécessaires pour accomplir sa tâche spécifique, rien de plus. Par exemple, une étape de déploiement vers **Kubernetes** ne devrait avoir accès qu'à

son namespace désigné et non à l'ensemble du cluster.

L'impact de cette transformation est immédiat et significatif en matière de réduction des risques. En adoptant les **identités fédérées** et les **accès éphémères**, nous rendons les **secrets** dérobés obsolètes en un temps record, minimisant ainsi le blast radius d'une compromission. Même si un attaquant réussit à s'emparer d'un jeton, celui-ci sera rapidement invalide, rendant l'exploit caduc et ralentissant considérablement la progression de l'attaque. C'est une mesure fondamentale qui élève la barre pour les cybercriminels et renforce de manière significative la résilience de votre **chaîne d'approvisionnement logicielle**.

Maîtriser les Dépendances avec un Pinning Rigoureux

Notre deuxième pilier consiste à reprendre le contrôle total de nos dépendances, une tâche qui nécessite une discipline rigoureuse. C'est un peu comme préparer une recette complexe : vous ne laisseriez jamais le chef choisir ses propres ingrédients au hasard à chaque fois, n'est-ce pas ? Vous insisteriez pour utiliser des quantités précises et des fournisseurs vérifiés. Dans le monde du développement, le pinning des dépendances, c'est exactement cela : s'assurer que chaque composant, chaque bibliothèque, chaque image Docker utilisée dans votre pipeline **CI/CD** est spécifiquement identifiée et ne peut pas être modifiée à la volée.

Concrètement, cela signifie non seulement de pinner vos dépendances à un hachage de commit spécifique plutôt qu'à une version mutable ou une balise flottante, mais aussi d'auditer de manière proactive les dépendances transitives. Il ne suffit pas de s'assurer que l'action principale est sécurisée si celle-ci, à son tour, tire des composants tiers par une balise mutable qui pourrait être redirigée vers un commit malveillant. Vous n'êtes jamais plus en sécurité que l'élément le plus faible de votre **chaîne d'approvisionnement logicielle**. Des outils d'**observabilité** de la chaîne

d'approvisionnement peuvent aider à cartographier et à surveiller ces dépendances.

L'impact de ce **pinning des dépendances** est de solidifier l'intégrité de votre logiciel dès la phase de construction. En fixant chaque composant par son **hachage de commit** cryptographique, vous garantissez que la version utilisée est exactement celle que vous avez approuvée, à chaque exécution du pipeline. Cela élimine la possibilité qu'un attaquant insère subrepticement du code malveillant en modifiant une balise ou en publiant une nouvelle version sous le même nom. C'est une défense essentielle contre les attaques de type typosquatting ou les compromissions de registres de paquets, car elle impose une transparence et une immutabilité que les attaquants peinent à contourner.

Appliquer les Fondamentaux de la Sécurité des Dépôts

Enfin, le troisième pilier, bien que souvent relégué au second plan car jugé basique, est pourtant fondamental : il s'agit d'appliquer sans compromis les bonnes pratiques de sécurité à vos dépôts de code. Si vous ne sécurisez pas la source, le reste de vos efforts sera vain. C'est comme installer des serrures complexes sur toutes les portes de votre usine, mais laisser la porte d'entrée principale grande ouverte. Ces mesures, bien que connues, sont trop souvent négligées ou appliquées avec une flexibilité excessive.

Sur le terrain, cela implique l'activation et l'application stricte de la protection de branche pour éviter les modifications directes, la mise en place de revues de code (PR reviews) obligatoires pour chaque changement et l'interdiction formelle de contournements administratifs, même pour les plus expérimentés. De plus, l'authentification multi-facteurs (**MFA**) doit être étendue à l'ensemble de l'organisation, pour chaque compte ayant accès aux dépôts ou aux systèmes **CI/CD**. L'utilisation de commits signés, facilitée par des outils comme Gitsign qui

simplifient la gestion des clés par rapport à PGP, assure l'authenticité et l'intégrité de chaque modification, bien que l'intégration native par des plateformes comme GitHub puisse encore être améliorée pour en faciliter l'adoption massive.

L'impact combiné de ces fondamentaux est de créer une posture de sécurité beaucoup plus résiliente pour votre code source. La **revue de code** et la protection de branche empêchent les injections de code non autorisées, tandis que la **MFA** et les **commits signés** renforcent l'identité de l'auteur et l'intégrité historique du dépôt. Ensemble, ces mesures érigent des barrières robustes contre les manipulations internes et externes, garantissant que le code qui entre dans vos **pipelines de déploiement** est légitime et vérifié. Cela permet une meilleure traçabilité et une responsabilité claire, des éléments essentiels pour une **chaîne d'approvisionnement logicielle** digne de confiance.

Agir Maintenant : Le CI/CD est Votre Nouvelle Ligne de Front

Ce que les attaques TeamPCP nous ont clairement montré, c'est que nos systèmes **CI/CD** ne sont plus de simples outils techniques, mais des extensions à part entière de nos environnements de production. Les traiter différemment, avec une sécurité moindre, est une erreur stratégique qui nous expose à des risques inacceptables. L'heure n'est plus à la procrastination ; il est impératif d'appliquer à nos pipelines la même rigueur, les mêmes protocoles et la même vigilance que ceux que nous réservons à nos systèmes critiques en production, qu'il s'agisse de nos clusters **Kubernetes** ou de nos bases de données. C'est une question de survie dans un paysage de menaces qui ne cesse d'évoluer.

Nous sommes actuellement témoins des prémices d'un phénomène qui ne fera que s'amplifier : chaque compromission réussie de la **chaîne d'approvisionnement logicielle** engendre de nouvelles fuites d'identifiants, de **secrets** et de points d'accès. Ces informations sont ensuite réutilisées pour compromettre d'autres systèmes, agrandissant sans cesse le rayon d'action des attaques. C'est ainsi que se construisent et s'étendent les menaces de grande envergure, transformant des incidents isolés en défis systémiques. La bonne nouvelle, c'est que nous avons déjà les connaissances et les outils nécessaires pour stopper cette spirale infernale. Nous devons simplement faire preuve de la volonté d'appliquer ces solutions avec constance et détermination. Le moment d'agir est maintenant.