PROGRAMMEZ!
Le magazine des développeurs

Forensic en Python

Chiffrer vos données

CodeWhisperer

Comprendre les malwares

Secure BY Design

L'ARCHITECTURE

10 ans

SPÉCIAL AUTOMNE 2023
M 01642 - 12 - F: 6,99 € - RD

Ce numéro a été sécurisé avec

aws
CAST
Software Intelligence for Digital Leaders
ECLIPSE FOUNDATION
HARDEN
salesforce
VERACODE

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

Toujours disponible !

PROGRAMMEZ!

Le magazine des dévs

N°259
09/10
2023

SPÉCIAL CARRIÈRE

Se former
Les profils recherchés
Les salaires
Devenir Freelance

SolidJS
Svelte
Qwik

Choisis ton prochain framework JS

M 04319 - 259 - F: 6,99 € - RD

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

La sécurité passe (aussi) par le code

La Cybersécurité est parfois un objet non identifié. Tout le monde en parle, mais sait-on toujours de quoi on parle? Tous les jours ou presque, de nouvelles failles apparaissent ou de nouvelles attaques sont évoquées. Pour être efficace, la sécurité doit se déployer partout : les utilisateurs, le code, l'infrastructure, le réseau.

Depuis 25 ans, nous sensibilisons à la qualité du code, aux tests, au secure by design. Le code, et donc les piles techniques liées aux environnements applicatifs, est un vecteur à ne pas négliger. Un exemple doit faire réfléchir : Log4J. Il y a 2 ans, des failles détectées dans la librairie à semer une véritable panique : 17 000 packages concernés soit 4 % de l'écosystème Java, +8 % des packages présents sur Maven Central étaient impactés soit +35 000 packages sur les 435 000 présents sur le référentiel, sans compter les dépendances directes (+ 7 000). C'est un effet domino qui s'est joué. Log4J est utilisé par des centaines de milliers d'apps Java. CQFD.

Ces failles se cachent partout : dans les API, les SDK, les librairies, le matériel. Les failles dans les processeurs ne sont pas rares. Une des dernières est Downfall sur certains processeurs Intel. Personne ne peut prétendre être protégée de la moindre faille. Les IoT ont parfois été un nid à vulnérabilités. Des objets étaient commercialisés avec un niveau de sécurité aléatoire : mot de passe par défaut dans le code, des broches de debug non désactivées, des portes dérobées actives, etc.

Le secure by design se fond naturellement dans la qualité du code et inversement. Un code clair, respectant les règles de programmation, bien structuré, sans codes morts, est un code qui, de facto, réduit la surface d'attaque potentielle. Il faut éviter d'utiliser des librairies et API exotiques dont l'origine est douteuse ou obscure (c'est le code d'un dev d'un dev d'une connaissance sur reddit que je suis). On n'utilise pas un code provenant d'un site ou générer par une IA générative aveuglément. Il faut d'abord faire une revue de code (WTF effect pour les puristes). Analyse statique du code, analyse dynamique, prise en compte des warnings de compilation, vérification des dépendances, versionning de tous les composants.

« Oui nous le faisons », « évidemment, c'est la base ». En 2023, quand je vois encore des buffers overflows ou de l'injection de code, je me dis qu'il y a un bug dans la matrice et que les bonnes pratiques élémentaires ne sont pas respectées.

Prenons un autre exemple. Les conteneurs. Aujourd'hui, rien de plus simple que d'utiliser un conteneur ou une image Docker. Même déployer une infrastructure Kubernetes devient banale. Mais appliquez-vous les bonnes pratiques de sécurité? Les conteneurs sont-ils à jour et maintenus? Utiliser une image obsolète est un risque potentiel qu'il faut réduire.

Il existe des méthodes, des bonnes pratiques, des outils. Il faut les utiliser. Il faut sensibiliser et former les développeurs. Et il faut laisser du temps pour faire de la sécurité par conception. Oui, cela prend du temps à mettre en place, mais, après 2 ou 3 projets, ces réflexes deviennent naturels.

Les outils de développement ne sont pas à l'abri de failles. Notepad++, éditeur bien connu, comble pas moins de 4 failles dans sa dernière mise à jour. La dernière faille importante de Sudo a été comblée en 2021 : un buffer overflow permettait à un utilisateur local de devenir root ! Durant l'été 2023, 11 nouveaux CVE ont été publiés pour le noyau Linux et 15 ont été fixés ces derniers mois.

Bienvenue dans l'édition 2023 de notre hors-série Sécurité & qualité du code.

François Tonic
alias Gandalf dans les mines de la Moria contre le Balrog

PROCHAIN NUMÉRO

PROGRAMMEZ! N°260

Disponible à partir du
3 novembre 2023

NUMÉRO EN COURS

PROGRAMMEZ! N°259

Disponible depuis le
8 septembre 2023

Contenus

3 Edito
La sécurité passe par le code
François Tonic

6 Agenda & les événements Programmez!

7 Les partenaires 2023

8 « développeurs, faites du droit ! »
Stéphane Dubreuil

9 « Rendons la vie difficile aux attaquants »
Sabrine Guihéneuf

10 AWS : 300 services dédiés à la sécurité
Andrew Vennekotter

12 Protéger l'écosystème open source : enjeux et défis
Mikaël Barbera

14 La notion d'AppSec
Bertrand Méens

15 Comment la sécurité est-elle implémentée et gérée dans la plateforme Salesforce ?
François Tonic

17 IA générative & sécurité
François Tonic

18 Sécurisation des API
Kiet Yap

19 Pour aller plus loin

20 Open source et sécurité
Sylvain Cailliau

26 Des outils open source pour la sécurité
Romy Alula

29 Sécuriser votre annuaire AD
Guillaume Mathieu

32 Secure by design : à la recherche des fuites mémoires en .Net
Christophe Nasarre

34 Techniques de développement d'un code sécurisé
Riadh Mnasri

37 Comment l'IA peut aider les développeurs et le RSSI à produire des logiciels sécurisés
Grégory Domagala

46 DevSecOps : de la sécurité dans mon DevOps
Adrien Pessu

51 Un peu de Forensic en Python
Franck Ebel

56 La BOLA

59 Azure AD et la sécurité partie 1
Arnaud Pilon, Théo Louis-Tisserand, Aymeric Palhiere

63 La sécurité des données dans MariaDB
Christophe Villeneuve

67 MegaLinter
Nicolas Vuillamy

71 Malwares à logique événementielle
Natacha Bakir

75 Améliorer la production et la sécurité du code avec Amazon Code Whisperer
Max Thomas

80 JobTeaser : la sécurité et le contrôle sur les données

81 MyTFS : supporter 3 millions de connexion simultanées
François Tonic

Divers

83 Abonnement à Programmez!

5 Boutique Programmez

Abonnement numérique (format PDF)
directement sur www.programmez.com

L'abonnement à Programmez! est de 55 € pour 1 an, 90 € pour 2 ans.
Abonnements et boutiques en pages 42 et 25

Programmez! est une publication bimestrielle de Nefer-IT.
Adresse : 57, rue de Gisors 95300 Pontoise – France. Pour nous contacter : redaction@programmez.com

Boutique Boutique Boutique Bouti

Les anciens numéros de PROGRAMMEZ!

Le magazine des dévs

Complétez votre collection....

Tarif unitaire 6,5 € (frais postaux inclus)

☐ 250	: ☐ ex	☐ HS07	: ☐ ex	Commande à envoyer à : **Programmez!** 57 rue de Gisors 95300 Pontoise
☐ 251	: ☐ ex	☐ 254	: ☐ ex	
☐ 252	: ☐ ex	☐ HS10	: ☐ ex	
☐ 253	: ☐ ex			

soit ☐☐ exemplaires x 6,50 € = ☐☐☐☐ € ..... ☐☐☐☐ € soit au **TOTAL** = ☐☐☐☐ €

☐ M. ☐ Mme Entreprise : ☐☐☐☐☐☐☐☐ Fonction : ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Prénom : ☐☐☐☐☐☐☐☐☐☐☐☐ Nom : ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Adresse : ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Code postal : ☐☐☐☐☐ Ville : ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Règlement par chèque à l'ordre de Programmez ! I Disponible sur www.programmez.com

Offres pouvant s'arrêter à tout moment, sans préavis. **PROG spécial 12**

Les événements Programmez!

Meetups Programmez!

Les meetups 2023 :

7 novembre - 5 décembre 2023
3 janvier 2024

Où : The Coding Machine
56 Rue de Londres, 75008 ParisParis
A 5 minutes de la gare Saint Lazarre : ligne L, H, métro 3 / 9 / 12 / 14

Conférence DevCon #20
100 % Sécurité
19 octobre au campus EFREI Paris

Conférence DevCon #21
100 % Game Dev
14 décembre au campus ESGI

Informations & inscription : programmez.com

octobre 2023

Lun.	Mar.	Mer.	jeu.	Ven.	Sam.	Dim.
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
	Paris TestCon / Paris	Assises de la sécurité / Monaco	Cloud Nord / Lille Volcamp 2023 / Clermont Ferrand Forum PHP / Paris			
16	17	18	19	20	21	22
			DevFest / Nantes Agile Tour Rennes DevCon / Paris			
23	24	25	26	27	28	29
		ScalaIO / Paris	Codeurs en Seine / Rouen			
30	31					

novembre 2023

Lun.	Mar.	Mer.	jeu.	Ven.	Sam.	Dim.
		1	2	3	4	5
6	7	8	9	10	11	12
				BDX I/O / Bordeaux		
13	14	15	16	17	18	19
		DevFest Strasbourg			Capitole du Libre/ Toulouse	
20	21	22	23	24	25	26
			DevOps D-Days / Paris Agile Grenoble			
27	28	29	30			
			Prestashop Developer Conference			

DÉCEMBRE 2023

- Open Source Experience : 6 & 7 décembre / Paris
- Agile Tour Aix-Marseille : 7 décembre
- DevFest Dijon : 8 décembre

JANVIER 2024

- SnowCamp : 31 janvier au 3 février, Grenoble

Merci à Aurélie Vache pour la liste 2023, consultable sur son GitHub : https://github.com/scraly/developers-conferences-agenda/blob/master/README.md

PROCHAIN NUMÉRO

PROGRAMMEZ! N°260

Disponible à partir du
3 novembre 2023

NUMÉRO EN COURS

PROGRAMMEZ! N°259

Disponible depuis le
8 septembre 2023

Les partenaires 2023 de

PROGRAMMEZ!

Le magazine des dévs

Niveau maître Jedi

salesforce

Progress®

Niveau padawan

OSAXiS

Vous voulez soutenir activement Programmez! ?
Devenir partenaires de nos dossiers en ligne et de nos événements ?

Contactez-nous dès maintenant :

ftonic@programmez.com

« Développeurs, faites du droit ! »

Depuis leurs premières lignes de code les développeurs sont sensibilisés à la sécurité. Leurs allocations mémoires sont surveillées, les contrôles de champs vérifiés et le choix de leur lib crypto challengé ! Des associations accompagnent même, tout au long de leur vie professionnelle, les développeurs dans leur appropriation du sujet (et pour n'en citer qu'une : The Open Worldwide Application Security Project - l'OWASP), ce contexte de formation initiale et continue devrait favoriser la sécurité du code... et pourtant…

Stéphane DUBREUIL

Directeur Cybersécurité du groupe SOFTEAM.

Stéphane Dubreuil est diplômé d'un master en Analyse des Systèmes Stratégiques (spécialisé dans les systèmes de Défense) et de l'Institut des Hautes Études de la Sécurité et de la Justice (INHESJ),

En 20 ans de carrières il a occupé différents postes de RSSI au sein de ministères (Défense, Finances, Justice), passé par l'ANSSI il a aussi exercé 6 ans comme RSSI de Thales SIX. Il est aujourd'hui le Directeur Cybersécurité du groupe Softeam.

Cet exigeant travail de sécurisation est encore mal valorisé voir incompris et dès lors que l'on rentre dans le « time to market », les bonnes pratiques de sécurité s'apparentent alors à de la perte de temps, de la complexité, du retard… pourtant à y regarder de plus près cette sécurisation protège utilisateurs et éditeurs.

La protection des utilisateurs

Que vos utilisateurs soient des particuliers, des établissements publics ou des entreprises (petites moyennes ou grandes) on sait l'impact de l'exploitation d'une vulnérabilité pour l'utilisateur – que cette vulnérabilité soit système ou applicative : Perte de ses données, de son système et dans les cas les plus tragiques, perte de son écosystème, comprendre la faillite de l'entreprise.

Cette dernière conséquence peut sembler excessive cependant la perte partielle ou totale du SI ou des données qu'il héberge – lié à l'exploitation d'une vulnérabilité a « *plusieurs conséquences cumulatives négatives pour une entreprise : perte financière directe […]; atteinte à l'image auprès des investisseurs, clients et opinion publique ; risque légal lié à une rupture contractuelle […]* » et « *peut conduire une entreprise, et notamment une TPE ou PME, structurellement plus fragile, à disparaître.* » comme le précise le rapport d'information du sénat : « La cybersécurité des entreprises - prévenir et guérir : quels remèdes contre les cyber virus ? » Rapport d'information n° 678 (2020-2021), du 10 juin 2021.

La protection de l'éditeur

De fait si, en France, il existe peu de jurisprudence concernant la responsabilisation des éditeurs sur les conséquences de l'exploitation de vulnérabilité dans leur produit, ce n'est pas nécessairement le cas au-delà de nos frontières et cela ne sera sans doute pas toujours le cas chez nous.

En effet les pays anglo-saxons ont une judiciarisation plus forte de ses évènements – il faut lire les CGU de leurs produits pour bien saisir les enjeux, une note de Bercy : « La responsabilité des fournisseurs de systèmes numériques » est assez éloquente sur le sujet.

Quant à notre propre droit, il évolue significativement, la directive européenne NIS 2 applicable à plusieurs milliers d'entreprises dès octobre 2024 précise en particulier que « *Les entités essentielles et importantes devraient donc évaluer […] les pratiques de cybersécurité de leurs fournisseurs et prestataires de services, y compris de leurs procédures de développement sécurisées.* ». À cela s'ajoute un texte en cours de rédaction, le « Cyber Resilient Act » dont « *les exigences et obligations essentielles contraindraient les fabricants à tenir compte de la cybersécurité dans la conception, le développement et la fabrication des produits comportant des éléments numériques, à exercer une diligence raisonnable sur les aspects de sécurité lors de la conception et du développement de leurs produits, à être transparents sur les aspects de cybersécurité* »

Dans l'intérêt des utilisateurs et au regard de l'évolutivité du droit – qui va très fortement responsabiliser les éditeurs dans leur pratique de développement sécurisé. Il est indispensable que les développeurs imposent le temps de la sécurisation.

Enfin si ce n'est pour les utilisateurs ou les éditeurs il faut qu'ils le fassent pour eux. Car si le code n'a pas de frontière, le droit applicable lui en dépend et l'imputabilité des vulnérabilités ne sera peut-être pas – partout – celle de l'éditeur – qui pourra aussi être amené à regarder dans quelle condition le code a été produit (matériel personnel hors du périmètre de sécurité de l'entreprise ?).

Rendons la vie difficile aux attaquants !

Sabrine Guihéneuf
Présidente des Assises de la Cybersécurité 2023 et administratrice du CESIN

Depuis plusieurs années, la cybersécurité est venue sur le devant de la scène à l'aune de la multiplication des cyberattaques ainsi qu'une augmentation de la pression réglementaire en la matière. De manière concomitante, et probablement du fait d'une plus grande médiatisation des attaques et fuites de données, le consommateur est de plus en plus exigeant concernant la protection de ses données et de sa vie privée, au point d'en devenir un véritable avantage compétitif mis en avant dans certaines publicités.

Rappelons l'évidence : dans notre quotidien, nous ne pouvons pas nous passer des applications, que ce soit pour des raisons professionnelles ou personnelles. La sécurité applicative est donc un des remparts les plus importants pour la protection des données, d'où la nécessité de renforcer la collaboration entre populations cyber et de développement. Car c'est ensemble que nous pourrons progresser collectivement !

En effet, même si par le passé la collaboration a pu être laborieuse dans certains contextes, elle peut devenir un duo gagnant lorsque un partenariat de confiance se construit dans la durée entre la cyber et les développeurs, permettant ainsi d'anticiper la prise en compte des éléments cyber dans les développements, et réciproquement dans la prise en compte des contraintes de développements dans les guidelines cyber, plutôt que de subir des recommandations d'audit à mettre en œuvre avant la mise en production prévue pour hier.

Et concrètement ? Osons collaborer ensemble, dès les prémices d'une idée de projet, afin de coconstruire la cybersécurité du produit final, car vous détenez la partie la plus importante de la réponse, et nous pouvons aider à la compléter avec nos expertises et outils.

Il en existe à ce jour de nombreux pouvant aider à intégrer la sécurité dans le cycle de développement, qu'il soit en mode agile ou en cycle en V plus classique, et de manière plus ou moins automatisée. Ces outils doivent être choisis en collaboration entre les équipes cyber et les développeurs, et s'intégrer dans le quotidien des développements. Il faut que cela puisse permettre aux développeurs d'identifier par eux-mêmes les failles potentielles (clé oubliée dans du code, failles les plus communes identifiées par l'OWASP...) et d'être en capacité de les corriger en toute autonomie, sans que l'équipe cyber soit derrière leur dos, ce qui n'est agréable pour personne. Le rôle de l'équipe cyber serait plutôt celui de « consultant interne » pour aider les développeurs à comprendre le risque de telle ou telle faille, et aider à la coconstruction de solutions, plutôt que d'être la « police » du code.

Je suis convaincue d'une chose : l'équipe cyber ne doit pas avoir le monopole des solutions de sécurité, au contraire ! Ils sont d'autant plus percutants lorsqu'ils sont dans les mains de ceux qui font ou opèrent, et lorsqu'ils se sont inscrit comme un réflexe et une évidence, à l'instar de la qualité ou la performance du produit. Je suis persuadée que chaque développeur souhaite que ses développements soient de qualité et exempts de failles, il n'y a rien de plus désagréable de voir le fruit de son travail mis à mal par des attaquants. Et c'est là que nous pouvons vous aider en vous apportant les clés de compréhension.

Que ce soit pendant ou en dehors des projets, les équipes doivent se parler et apprendre l'une de l'autre. Et soyons fous, pourquoi pas former des développeurs au test d'intrusion web et inversement, former les équipes cyber au développement et les différents frameworks qui existent dont certains peuvent embarquer by design des fonctions de sécurité. L'objectif n'est pas de faire une reconversion professionnelle (quoi que, nous manquons de bras en cyber, à bon entendeur !), mais en « vivant la vie » de l'autre, nous pouvons mieux appréhender les contraintes et s'y adapter. Nos recommandations en matière de cyber n'en seront que plus pertinentes, et vos propositions mieux adaptées face à la menace. Ce ne sont que des propositions, pouvant être débattues, questionnées, remises en cause, améliorer, et surtout à compléter avec votre moitié de l'équation : ensemble, nous pouvons mener la vie dure aux attaquants, leur rendre la tâche vraiment difficile voire impossible.

Pour conclure, je vous propose un défi : collaborez avec la communauté cyber, et ensemble, faisons en sorte qu'au prochain test d'intrusion de l'application, les auditeurs ne trouvent aucune faille pouvant mettre à mal les données gérées par le produit.

AWS : 300 services dédiés à la sécurité respectant 143 normes, standards et certifications

La sécurité est un sujet critique pour les clients. AWS possède une solide équipe dédiée à la sécurité en interne et envers les utilisateurs. Nous avons voulu en savoir plus avec Andrew Vennekotter. Andrew est senior manager security chez AWS. Il dirige une équipe sécurité travaillant avec les entreprises et les autorités en Europe.

Quels sont les engagements d'AWS en matière de sécurité ?

AWS soutient les engagements des gouvernements en matière de sécurité des données. Nous proposons plus de 300 services dédiés à la sécurité dans le Cloud, respectant 143 normes de sécurité et certifications de conformité au niveau mondial. Nous pensons que quiconque travaillant dans le Cloud AWS peut contrôler l'ensemble de ses données sans compromis. Notre engagement de souveraineté numérique, le "Digital Sovereignty Pledge", que nous avons présenté en novembre dernier, offre à tous nos clients l'ensemble le plus avancé de contrôles et de fonctions de souveraineté disponibles dans le cloud. L'un des principaux piliers de cet engagement est la capacité de tout chiffrer partout, que ce soit en transit, au repos ou en mémoire. Tous les services AWS qui impliquent des données de clients supportent déjà le chiffrement.

L'utilisation du cloud a-t-elle résolu les problèmes de sécurité (logiciel, infrastructure, etc.), si, bien sûr, les directives des fournisseurs sont respectées ?

Les clients d'AWS utilisent en toute confiance notre infrastructure cloud, car il s'agit de l'option la plus sûre qui soit. En tant que l'un des plus grands fournisseurs de cloud, nous avons une large visibilité sur l'évolution des besoins en matière de sécurité à l'échelle mondiale. Par exemple, les clients peuvent utiliser notre service AWS Shield Advanced pour obtenir une couche de sécurité supplémentaire au niveau périmétrique afin de prévenir automatiquement les attaques par déni de service distribué (DDoS).

Disposer d'un cadre de sécurité de bout en bout comme notre Well-Architected Framework aide les organisations à identifier, protéger, détecter, répondre et récupérer des risques de sécurité plus rapidement. L'informatique dématérialisée favorise également l'automatisation des contrôles de sécurité de l'infrastructure et des applications internes afin de créer une protection et des contrôles de conformité 24h/24 et 7j/7. Parallèlement, des outils de raisonnement automatisés aident les entreprises à prouver mathématiquement les plus hauts niveaux de sécurité nécessaires.

La sécurité de l'infrastructure est parfois sous-estimée. Comment un acteur comme AWS aborde-t-il cette question et la met-il en pratique ?

La sécurité est le pilier central autour duquel AWS construit son infrastructure, conçoit et gère ses produits et services. Notre architecture cloud et réseau est conçue pour être résistante et sécurisée afin de répondre aux besoins des organisations et secteurs les plus réglementés, notamment les secteurs de la santé, les services financiers et les institutions du secteur public. Avec 102 zones de disponibilité (centres de données dans un lieu physique) dans 32 régions dans le monde, l'infrastructure cloud d'AWS est la plus étendue. Nous continuons à innover pour la conception et les systèmes de nos centres de données afin de les protéger contre les risques naturels et d'origine humaine. Pour aller plus loin, AWS met en place des contrôles, construit des systèmes automatisés et se soumet à des audits de tiers pour valider régulièrement sa position en matière de sécurité et de conformité.

Les services AWS offrent également des capacités avancées d'"autoréparation" et de récupération automatisée. Par exemple, l'utilisation de groupes Auto-Scalling dans plusieurs zones de disponibilité permettent à l'infrastructure de surveiller la santé des instances EC2 et de remplacer rapidement une instance défaillante par une Amazon Machine Image (AMI). En outre, l'utilisation d'Amazon CloudWatch, d'AWS Lambda et d'autres services et capacités permet d'automatiser les actions de récupération, notamment le déploiement d'un environnement AWS complet et d'une application, le basculement vers une autre région AWS, la restauration de données à partir de sauvegardes, etc. Pour savoir comment nous assurons la sécurité dans nos centres de données, consultez le lien suivant : https://aws.amazon.com/fr/compliance/data-center/controls/

Dans leurs propres environnements, les clients peuvent construire une infrastructure plus sûre et "immuable" en exploitant les services d'infrastructure en tant que code (IaC), notamment AWS CloudFormation.

AWS, comme d'autres acteurs de l'insustrie, parle beaucoup de Rust. Ce langage a la réputation d'être sûr dès sa conception et d'éviter bon nombre des problèmes de fuites de mémoire que l'on rencontre avec C++, par exemple. Comment et où Rust est-il utilisé dans les services AWS ?

Chez AWS, Rust est rapidement devenu essentiel à la construction d'une infrastructure à l'échelle. Firecracker est une technologie de virtualisation open source qui alimente AWS Lambda et d'autres offres serverless. Il a été lancé publiquement en 2018 en tant que notre premier produit notable mis en œuvre en Rust. Nous utilisons Rust pour fournir des services tels que Amazon Simple Storage Service (Amazon S3), Amazon Elastic Compute Cloud (Amazon EC2), Amazon CloudFront, et plus encore. En 2020, nous avons lancé Bottlerocket, un système d'exploitation de conteneurs basé sur Linux et développé avec Rust, et notre équipe Amazon EC2 utilise Rust comme langage de choix pour les nouveaux composants du système AWS Nitro, y compris les applications sensibles, telles que Nitro Enclaves.

Les failles successives de Log4J et la nécessité de déployer plusieurs versions pour combler

les lacunes ont montré qu'une librairie peut inclure des centaines de milliers d'applications et des millions d'utilisateurs. Avez-vous été surpris par l'ampleur du problème et la difficulté de combler toutes les failles ?

À mesure que la société adopte de nouvelles technologies numériques, il est naturel que de nouvelles vulnérabilités apparaissent. Chez Amazon, nous sommes très attentifs à nos clients et nous innovons constamment pour les aider à réagir rapidement et automatiquement à l'évolution des menaces de sécurité. Avec Log4j, les clients ont pu mettre en œuvre un correctif dans les deux jours suivant la divulgation de la faille afin de fournir une solution immédiate à grande échelle pendant qu'ils se préparaient à mettre à jour la prochaine version de Log4j. Dans l'intervalle, les clients ont utilisé d'autres services tels que AWS Network Firewall, AWS WAF et AWS Shield Advanced pour la défense active, l'inspection et la protection du trafic réseau. Ces services sont gérés par AWS et sont conçus pour s'adapter aux exigences de nos plus gros clients en matière de réseau. Lorsque de tels événements se produisent, Amazon Inspector peut aider les clients à prendre des mesures immédiates et hiérarchiser les correctifs à apporter en découvrant et en acheminant automatiquement les découvertes de failles en temps quasi réel vers les équipes appropriées.

Aujourd'hui, quand on regarde une architecture moderne (app ou infra), c'est un "mille-feuille" de couches techniques. La complexité n'a jamais été un allié de la sécurité. Sommes-nous condamnés à cette accumulation de couches, ou pouvons-nous réellement simplifier la distribution interne ?

L'un des principes de leadership d'Amazon est "Inventer et Simplifier", que nous intégrons dans notre processus de développement de logiciels. Les développeurs ne sont pas récompensés pour avoir rendu les solutions plus complexes, mais pour avoir trouvé des solutions simples à des problèmes difficiles. Parfois, il faut une communauté pour y parvenir. C'est pourquoi nous nous engageons activement auprès de fondations open source, d'associations professionnelles, d'organismes de normalisation et d'organisations réglementaires afin d'améliorer la sécurité des utilisateurs finaux dans l'ensemble du secteur du cloud. Prenons l'exemple de notre annonce du projet Open Cybersecurity Schema Framework (OCSF). Avec la prise en charge de l'OCSF, Amazon Security Lake centralise automatiquement les données de sécurité des environnements AWS, des fournisseurs de logiciels en tant que service (SaaS), des sources sur site et dans le cloud, afin de fournir une compréhension holistique des données de sécurité dans l'ensemble d'une organisation. Il est ainsi plus facile pour les équipes de sécurité de regrouper les données de sécurité à grande échelle dans un format commun pour les utiliser avec leurs solutions d'analyse privilégiées, ce qui permet de réagir plus rapidement aux menaces et d'y remédier.

Les API sont régulièrement critiquées pour des problèmes de sécurité, des versions obsolètes, des flux mal codés ou incompréhensibles, etc. Comment AWS traite-t-il ce problème spécifique ?

Au fond, les API sont des contrats visant à fournir une certaine réponse ou à effectuer une certaine action en réponse à une demande authentifiée et sécurisée. La nature de la technologie en cloud axée sur les API offre un niveau de visibilité et d'automatisation jusqu'alors impossible. La vraie question est donc de savoir comment assurer la sécurité de ces services. La sécurité commence par une culture d'appropriation, d'automatisation et de test. Les équipes d'AWS exploitent les services qu'elles construisent et doivent intégrer des examens de sécurité, des tests de pénétration, des analyses de code et des vérifications formelles dans le processus de développement. C'est pourquoi nous disposons d'un pipeline automatisé d'intégration continue et de livraison continue (CI/CD). Notre pipeline CI/CD intègre des tests rigoureux et des mesures visant à limiter les risques en divisant d'abord les déploiements en environnements plus petits, puis en les diffusant par vagues de plus en plus importantes au fur et à mesure que chaque vague fait ses preuves. Nous veillons également à ce que les changements restent rapidement réversibles grâce à des retours en arrière faciles et automatiques. Les clients peuvent ensuite utiliser des services tels que AWS CloudTrail, Amazon CloudWatch et Amazon GuardDuty, entre autres, dans le cadre d'une stratégie globale d'opérations de sécurité pour la surveillance continue et la détection des menaces.

L'OWASP reste un outil indispensable pour les développeurs. Que pensez-vous des attaques et des failles répertoriées ? Est-il normal de voir des injections de code/données non autorisées toujours dans le top 3 de l'OWASP en 2023 ?

Au niveau conceptuel, il existe une nouvelle catégorie OWASP qui peut être à l'origine de nombreux autres problèmes de sécurité : "Insecure Design". Développer des logiciels signifie avoir accès à des outils qui peuvent faire des choses étonnantes, mais il est essentiel que les développeurs comprennent leur propre modèle de menace lorsqu'ils conçoivent des logiciels. Les développeurs qui construisent sur AWS peuvent bénéficier de notre Well-Architected Framework, qui considère la modélisation des menaces comme une meilleure pratique fondamentale dans le pilier de la sécurité. En ce qui concerne les injections de code et de données non autorisées, nos services vont de la gestion de l'identité et de l'accès (IAM) à Amazon Cognito, en passant par le centre d'identité AWS IAM, le service d'annuaire AWS et des fonctions telles que l'authentification multifacteurs. En choisissant des services personnalisables, évolutifs et abordables, les entreprises peuvent mettre en œuvre, gérer, sécuriser, surveiller et établir des rapports sur les identités des utilisateurs, les normes d'authentification et les droits d'accès. De nouveaux produits comme Amazon Verified Permissions permettent de gérer des permissions et des autorisations précises au sein d'applications personnalisées.

Autre problème récurrent en matière de sécurité : que faire de tout le code Legacy ? Il y a Cobol, bien sûr, mais il y a aussi des millions d'applications et de services Java ou C# qui ont plus de 10-12 ans !

Pour en revenir au sujet des API axées sur les microservices, celles-ci peuvent être utiles pour les codes Legacy, car vous pouvez toujours remplir le contrat de l'API tout en améliorant le code sous-jacent de manière continue. Il s'agit d'un processus qui se produit organiquement lorsque les développeurs d'un service sont également chargés de l'exploiter.

Nous encourageons l'innovation pour rendre le codage plus rapide, cohérent et intuitif afin que les développeurs puissent construire rapidement et se concentrer davantage sur les aspects créatifs du codage. En avril, nous avons annoncé la disponibilité générale d'Amazon CodeWhisperer, un assistant de codage IA qui utilise un modèle fondamental pour améliorer radicalement la productivité des développeurs. Il génère des suggestions de code en temps réel sur la base des commentaires des développeurs en langage naturel et du code antérieur dans leur environnement de développement intégré (IDE). Cela contribue à la cohérence du code, permet d'accélérer le processus de mise à jour du code existant et aide à prévenir certaines failles de conception courantes en matière de sécurité.

La sécurité est notre priorité absolue. Nous veillons à ce que nos clients disposent du meilleur environnement cloud possible et puissent développer le bon niveau de sécurité pour leurs applications, afin qu'ils puissent consacrer plus de temps à la réalisation d'autres objectifs fondamentaux de l'entreprise.

Protéger l'écosystème open source : enjeux et défis

Mikaël Barbero
Head of Security Eclipse Foundation

ECLIPSE FOUNDATION

Aujourd'hui, environ 80 à 90 % des applications se basent sur du code open source. Cette réalité joue le rôle de catalyseur d'innovation dans le développement logiciel, accélérant celle-ci grâce à sa gratuité et à sa disponibilité à faible contrainte juridique. C'est ce phénomène que les Anglo-saxons nomment "permissionless innovation". Cependant, en raison d'un nombre croissant de menaces, il devient nécessaire de sécuriser efficacement la chaîne d'approvisionnement logicielle afin de garantir que l'open source continue de contribuer à l'innovation. Une meilleure cybersécurité pourrait également contribuer à atténuer toute tentative gouvernementale de surréglementation inutile.

Des exemples concrets d'open source qui ont changé le monde sont nombreuses : par exemple Internet et le World Wild Web. Les plus évidents ne sont autres que l'internet et le Web. Ils ont fondamentalement été rendus possibles grâce à l'open source et à la mise à disposition des technologies de manière globale et gratuitement. Le code open source a un impact positif sur la façon dont nous créons la technologie : chaque fois que nous construisons un nouveau produit ou une nouvelle application, nous nous appuyons sur les travaux préalables des pionniers qui ont créé la plateforme technologique précédente. Les développeurs sont ainsi aux commandes de la création et de l'adoption de ces nouvelles technologies. Cependant, cette incroyable influence qu'ont les développeurs accroît également leurs responsabilités vis-à-vis de leur code.

La cybercriminalité est une autre tendance majeure qui affecte notre monde numérique. À ce jour, si la cybercriminalité était une nation, elle serait la troisième plus grande économie mondiale, juste après les États-Unis et la Chine : un rapport sur la cybercriminalité de Cybersecurity Ventures estime par ailleurs que cette activité illégale coûtera plus de 10,5 trillions de dollars de dommages à nos sociétés d'ici 2025. En particulier, les attaques sur la chaîne d'approvisionnement logicielle ont augmenté de manière stupéfiante de 742 % au cours des trois dernières années, selon un rapport de Sonatype. Cette accélération fulgurante et ce n'est pas par accident : les cybercriminels sont dans une logique de rentabilité, et une attaque réussie sur un logiciel open source peut leur permettre d'atteindre des milliers de cibles potentielles et peut-être même déclencher une réaction en chaîne et atteindre des objectifs à deux ou trois niveaux de l'attaque initiale.

Chaîne d'approvisionnement logicielle

Que signifie-t-on par chaîne d'approvisionnement logicielle ? Fondamentalement, les développeurs écrivent, construisent, empaquettent et livrent leur code source aux consommateurs, tandis que certaines dépendances constituent également une part de plus en plus importante de nos produits et applications. Toutes ces étapes forment une chaîne d'approvisionnement, de manière assez similaire à ce que l'on peut observer dans l'industrie traditionnelle. Chaque étape de cette chaîne, y compris les dépendances, peut être la cible d'attaques et devenir une menace.

Les dépendances sont un cas intéressant en raison de leur enchevêtrement complexe et profond. Il n'est pas rare de voir une dépendance open source d'une application se trouver à six, sept ou huit niveaux de profondeur. Ces dépendances sont elles-mêmes la cible potentielle d'attaques concernant leurs constructions, leurs empaquetages et leurs livraisons au consommateur final.

Cela signifie-t-il que personne ne gère cette chaîne d'approvisionnement ? C'est une question légitime lorsque l'on sait que beaucoup de paquets open source sont d'une importance cruciale pour nos infrastructures numériques. De manière assez effrayante, ces paquets critiques sont maintenus par des équipes de tailles extrêmement réduites, composées tout au plus de 1 ou 2 personnes, voire dans certains cas sans plus aucun mainteneur à la barre.

Il est alors évident que nous avons un véritable problème du côté de l'approvisionnement concernant la maintenance et le support des chaînes d'approvisionnement logicielles. Un autre facteur est le nombre croissant de développeurs de logiciels adoptant une position défiante, mais techniquement correcte : "Je vous fournis du code open source *tel quel*. Je ne suis pas votre fournisseur." Le contrat qui sous-tend l'open source depuis le premier jour consiste à fournir du code gratuitement, mais aussi sans responsabilités ni garanties. Et il est vrai que dans la grande majorité des cas, il n'y a pas de relation commerciale entre les développeurs et les consommateurs de leur code. C'est donc une nouvelle faiblesse dans la chaîne d'approvisionnement logicielle : sans relation client/fournisseurs, comment remonter cette chaîne et l'améliorer globalement ?

Open source sous pression

Ces dernières années, certains piratages ont fait l'objet d'une large médiatisation. L'un des plus célèbres a été le bug Heartbleed qui a affecté OpenSSL en 2014. Celui-ci a aidé à la réalisation que l'open source est un maillon essentiel de l'infrastructure numérique. De manière intéressante, personne n'était surpris qu'un bug important dans OpenSSL puisse avoir des conséquences dramatiques en termes de cybersécurité, étant donné qu'OpenSSL est un composant de sécurité en soi. Mais plus récemment, nous avons eu Log4J et Log4Shell. Identifiée à la fin de l'année 2021, il s'agissait de l'une des pires failles de sécurité jamais enregistrées, d'autant plus choquante que personne ne s'attendait à un problème de sécurité provenant d'un composant aussi trivial qu'un framework de journalisation Java. Lorsque les gens ont commencé à regarder s'ils étaient affectés par Log4Shell, ils ont réalisé que Log4J était utilisé partout : c'est un composant simple, mature et qui remplit une fonction nécessaire à toute application serveur. Cet évènement à conduit à faire réaliser que l'open source est partout et potentiellement dangereux, indépendamment de l'innocuité relative des fonctions remplies par ces composants.

https://www.lunasec.io/docs/img/log4shell-logo.png

En raison de ces bugs et d'autres, nous commençons à constater un intérêt croissant des gouvernements en faveur d'une certaine régulation du code open source, tant aux États-Unis (décret exécutif

sur l'amélioration de la cybersécurité nationale) qu'en Europe (loi sur la cyberrésilience). Plus récemment (mars 2023), la Maison-Blanche a publié sa stratégie nationale en matière de cybersécurité, qui appelle à des changements dans la réglementation et la législation sur la responsabilité afin de garantir que ceux qui fournissent du code aux consommateurs soient tenus responsables de leur qualité.

Fort heureusement, les réglementations américaines indiquent que les développeurs open source ne sont pas responsables du code en raison de sa fourniture *"tel quel"*. Au lieu de cela, les réglementations désignent les producteurs — ceux qui tirent profit du code — comme les parties obligées.

Cependant, la situation est différente avec la loi sur la cyberrésilience de l'Union européenne, qui semble vouloir rendre les communautés, les projets, les développeurs et les fondations open source responsables de la cybersécurité de tout le code produit. Il est important de comprendre, cependant, que cette position n'affecte pas seulement l'Europe. La loi sur la cyberrésilience a une portée extraterritoriale, car elle s'applique à toute personne, fondation, projet ou entreprise qui met du code à disposition sur le marché européen. C'est un changement radical, car il modifie fondamentalement le contrat social mentionné précédemment, dans lequel les développeurs fournissent un code gratuit, mais sans responsabilité.

https://digital-strategy.ec.europa.eu/sites/default/files/styles/large/public/newsroom/items/Cyber_Resilience_SoMe_2_169_a9bur3VT9vNeg1mO4nzooVQfw_103596.png?itok=XVebnevT

Malheureusement, rompre ce contrat social entre les développeurs et les consommateurs de leur code aura des conséquences négatives involontaires. En effet, la chaîne d'approvisionnement du code open source est un moteur majeur de l'innovation mondiale et de la croissance économique, et modifier cette chaîne d'approvisionnement de manière que chaque étape soit légalement obligée en matière de sécurité aura un effet négatif considérable sur l'innovation.

La solution

Alors, que peuvent faire les fondations open source pour améliorer la cybersécurité de la chaîne d'approvisionnement logicielle et espérer atténuer certaines des tentatives de sur-réglementation de l'économie de l'innovation ? À la Fondation Eclipse, nous mettons en œuvre un certain nombre de mesures importantes, notamment :

- réviser nos politiques et processus ;
- améliorer et renforcer notre infrastructure ;
- renforcer la sécurité de nos projets ;
- et aider nos projets à améliorer leurs compétences et leur culture en termes de cybersécurité.

Nous investissons actuellement pour améliorer et élargir nos politiques et processus de sécurité, en veillant à suivre les meilleures pratiques. Ainsi, au cours des 12 derniers mois, nous nous sommes attelés à créer une équipe de cinq personnes spécialisées dans la sécurité afin de mettre en œuvre les points précédents. Cette équipe veille également à ce que tous les rapports de vulnérabilités rapportées par des chercheurs en sécurité soient traités et adressés par les projets de la Fondation. De nouveaux mécanismes et de nouveaux outils sont mis à disposition des projets afin de discuter de ces rapports avec plus d'efficacité : mailing list, projet Gitlab privé, et rapports de vulnérabilités privés GitHub.

Nous souhaitons aussi nous appuyer sur un bassin plus large d'expertise impliquant des experts en sécurité et du personnel spécialisé. Dans ce but, nous mettons en place une initiative de gestion des risques des menaces cyber, avec l'aide de nos entreprises membres, existantes et à venir.

De même, nous révisons nos politiques et procédures de sécurité de l'infrastructure. Pour ce faire, nous planifions des audits de sécurité effectués par des tiers ainsi que des tests de pénétration. La sécurité de cette infrastructure est cruciale pour nombre de nos projets. En effet, plus de la moitié d'entre eux utilisent cette infrastructure pour construire et déployer leurs projets. L'atout de cette infrastructure de construction commune est qu'il nous est ainsi plus facile d'ajouter de nouveaux services liés à la sécurité, notamment en termes d'analyse de dépendances et de vulnérabilité au moment de la construction des binaires. L'équipe est en train d'expérimenter plusieurs solutions avant d'en faire une plus large diffusion à nos projets.

Sécurité des projets

Nous aidons également nos projets à améliorer plus directement leur propre sécurité. Nous finançons par exemple des audits de sécurité réalisés par des tiers, tout en effectuant des analyses des dépendances des projets. Ainsi, nous pouvons atténuer les risques présentés par les packages tiers obsolètes ayant des vulnérabilités connues. Une autre initiative consiste à aider autant de projets que possible à passer à des constructions reproductibles, tandis que la création d'un programme de certification de projet basé sur SLSA révélera le niveau de maturité de chaque projet à nos consommateurs.

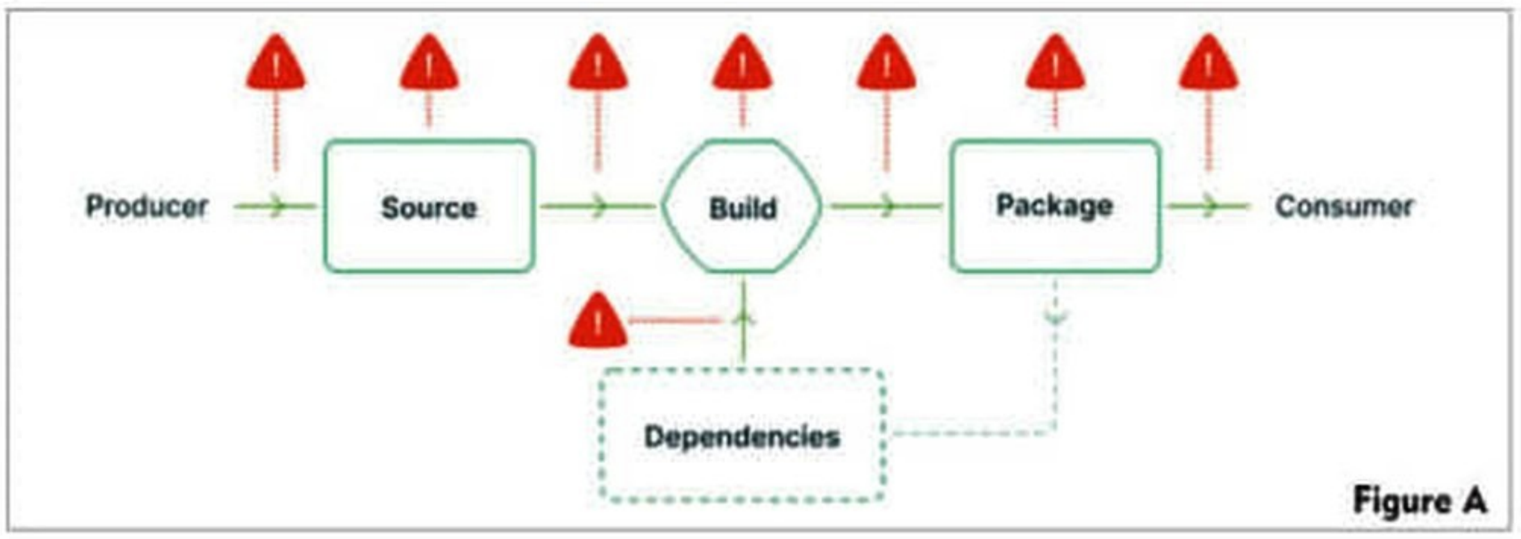


Figure A

https://slsa.dev/images/SupplyChainDiagram.svg **(Figure A)**

Enfin, nous sommes en cours de déploiement d'un outil de gestion d'organisation et de dépôt GitHub. En effet, nous avons à gérer plus de 150 organisations GitHub à ce jour, comptant plusieurs milliers de dépôts. Afin de s'assurer que les configurations des toutes ces entités soient sécurisées, il nous fallait un outil qui n'existait pas sur le marché, aussi nous l'avons créé : OtterDog. C'est un outil avec une approche *"as code"*, qui nous garantit une excellente auditabilité ainsi qu'une grande transparence.

Conclusion

Atteindre le niveau de cybersécurité requis sera un processus pluriannuel. Cependant, nous n'allons pas entreprendre ce voyage seuls. Nous avons eu la chance de bénéficier d'un financement de l'Open Source Security Foundation (OpenSSF) dans le cadre de son projet Alpha-Omega pour soutenir nos initiatives. Comme dit plus haut, nous travaillons également à la création d'un groupe de travail pour nous aider à poursuivre cette stratégie et rendre l'investissement durable à long terme au profit de l'ensemble de la communauté open source.

La notion d'AppSec

BERTRAND MÉENS A RÉPONDU À NOS QUESTIONS AUTOUR DE LA SÉCURITÉ APPLICATIVE ET DE L'OWASP.

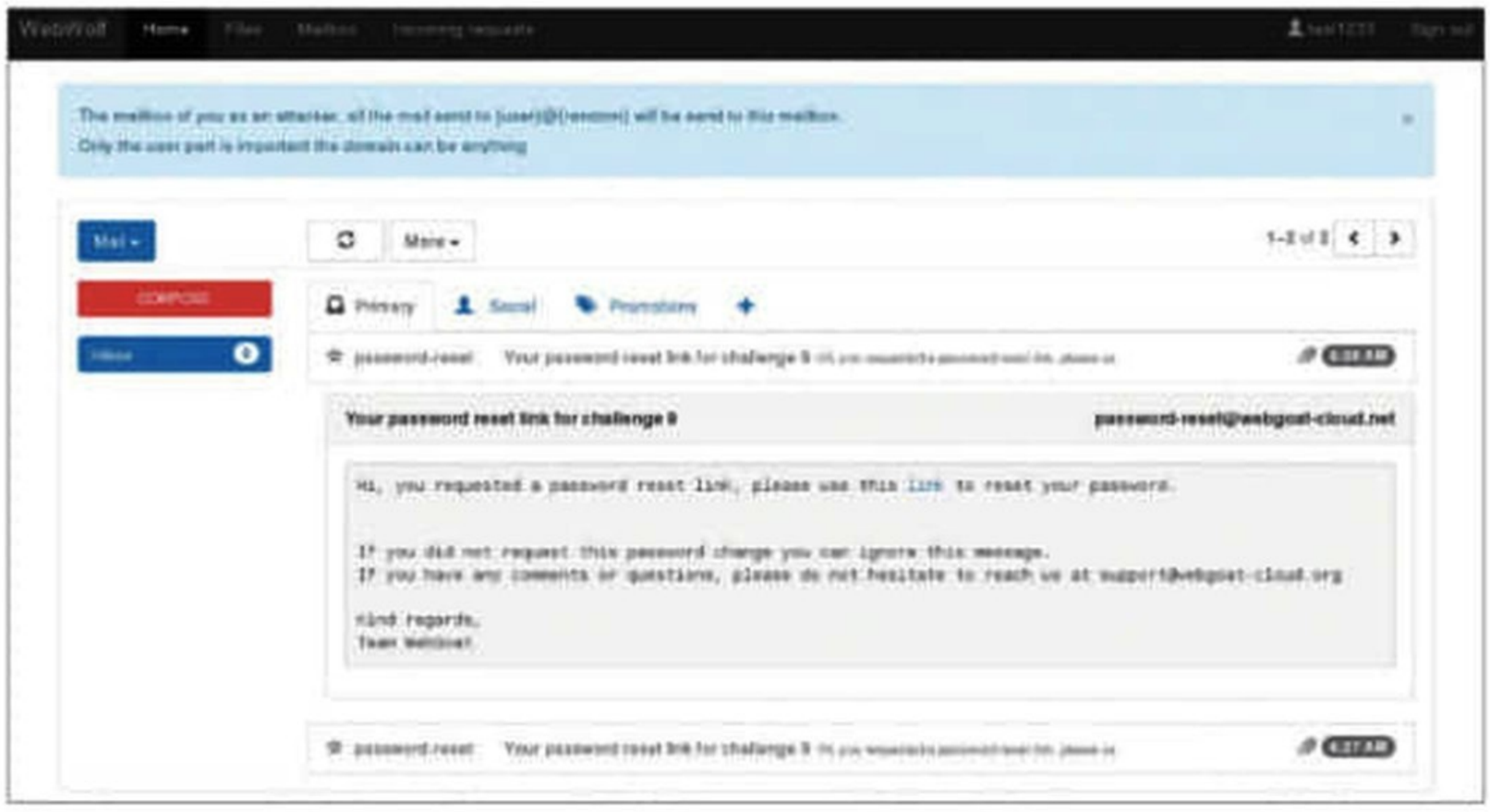
Bertrand est expert cybersécurité et DSI Adjoint au Groupe IRCEM.

Bertrand Méens

Actuellement DSI adjoint et toujours passionné par la cybersécurité, Bertrand Méens est un ancien RSSI puis CTO et travaille le sujet de la sécurité applicative depuis 2014. Il est membre du CESIN.

Le développeur est un maillon important de la sécurité, pensez-vous que sa place est toujours prise en compte ? Comment l'intégrer à la sécurité de l'entreprise, de l'IT en général ?

L'AppSec (sécurité applicative) est un carrefour entre les disciplines du développement et de la cybersécurité. Hormis l'impression de partager le métier de l'IT, ce sont deux disciplines qui ont peu de points communs. Pour se comprendre, il est essentiel que chacune des disciplines fasse l'effort de comprendre les enjeux et objectifs de l'autre. Le métier de développeur ne se résume pas à écrire des lignes de codes, c'est un métier complet et passionnant qui va de la compréhension fonctionnelle du secteur d'activité dans lequel il évolue à des connaissances techniques pour échanger avec le monde de l'infrastructure, et bien sûr celui de la cybersécurité ? Ce complément de compétences fera la différence sur le plan de l'employabilité et participe grandement aux nouvelles attentes des entreprises.



En intégrant des compétences dans le domaine de la cybersécurité, le développeur pourrait ainsi découvrir un nouvel aspect passionnant de son métier. Le professionnel de la cybersécurité doit quant à lui comprendre que l'ajout de sécurité dans les développements doit passer par des formations et de l'automatisation pour éviter que sa discipline soit vue comme une contrainte et ainsi faire preuve de pédagogie. Un RSSI prendra en compte la place du développeur s'il comprend déjà l'importance du développement dans son modèle de sécurité et si l'entreprise a un niveau de maturité suffisante dans son décloisonnement entre les métiers. Mon conseil est la gamification dans les formations avec l'usage de la solution WebGoat, demandez aux développeurs de se mettre dans la peau d'un hacker sans accéder au code source, c'est déroutant au premier abord puis les développeurs se prennent au jeu sur cette solution volontairement vulnérable. Les expériences de "vis ma vie" entre métiers permettent aussi de comprendre l'autre et ses contraintes, en vivant son quotidien pendant une journée.

L'OWASP est un des référentiels les plus connus. Est-ce pour vous une référence ? Si oui, pourquoi ? Si non, pourquoi ?

En sécurité applicative, les Tops 10 de l'OWASP sont clairement des références bien qu'il existe d'autres références comme le CWE Top 25 du SANS Institute. L'OWASP n'est d'ailleurs pas réellement un référentiel, mais une association qui s'est fait connaître pour son Top 10 des vulnérabilités web. Depuis d'autres Tops 10 ont été publiés comme celui concernant les API, et dont la mise à jour date de cette année. Cette association fournit aussi beaucoup de documentation comme les Cheat Sheets ou des guides (par exemple : AI Security and Privacy Guide), de la méthodologie comme SAMM ou des projets open source comme Zap, WebGoat ou Dependency-Track. Pour finir, l'OWASP est organisé en chapitres, avec une représentation territoriale et des rencontres. Les développeurs sont toujours les bienvenus pour partager leurs compétences ou simplement découvrir le monde de l'AppSec.

Les métiers de la sécurité sont parfois obscurs ou mal connus. Lorsqu'on lit la nomenclature de l'ANSSI, difficile de s'y retrouver non ?

Comme tout référentiel de métiers, la nomenclature de l'ANSSI utilise des acronymes et un jargon propres à la cybersécurité comme le font d'autres métiers. Le plus simple n'est-il pas de demander ? Toute entreprise, passée une certaine taille, aura des collaborateurs avec des compétences en cybersécurité, que ce soit des administrateurs sécurité, des RSO ou des RSSI. Il ne faut pas hésiter à pousser la porte de certains événements comme le FIC à Lille qui est gratuit et ouvert au public. Le dernier conseil pour les plus autodidactes est d'apprendre en pratiquant la veille sur des réseaux sociaux (LinkedIn, Twitter...) et en s'abonnant à des newsletters ou à des Podcasts techniques ou non, comme NoLimitSecu, NO LOG, ou Cybersécurité All Day. Programmez organise aussi des DevCon 100 % sécurité !

Comment la sécurité est-elle implémentée et gérée dans la plateforme Salesforce ?

François Tonic
Rédacteur en chef de Programmez!

Inutile de présenter Salesforce. Il est connu pour ses services de CRM mais aujourd'hui, la plateforme regroupe des centaines de services, avec une forte dominante autour de la donnée et de couches techniques allant de l'infrastructure au front-end, en passant par une marketplace, des IDE, des API, etc. Tout naturellement, la sécurité est cruciale. Sans sécurité, impossible de faire du CRM ou même de stocker, de traiter et d'échanger des données. Le secure by design n'est pas une illusion ni la sécurisation des API. Dans ce dossier, les experts Salesforce nous parlent de la manière dont la sécurité est pensée et intégrée.

« Dans une application comme Salesforce, la sécurité est une responsabilité partagée entre Salesforce et les clients. Comme tout SaaS, la partie infrastructure est gérée par Salesforce. Cela inclut la sécurité des datacenters, la continuité de service avec plusieurs sites d'hébergement, la redondance des équipements et toutes les certifications nécessaires pour assurer le service. Par exemple, toutes les connexions, et les flux qui y passent, sont chiffrées. Les vulnérabilités sont surveillées et des tests d'intrusion sont réalisés régulièrement. Les clients ont la responsabilité de configurer les accès de leurs utilisateurs avec le bon niveau de sécurité : SSO, MFA obligatoire et droits d'accès sur les données par exemple. Des options de sécurité sont disponibles comme le chiffrement applicatif avec BYOK, les logs, l'anonymisation des données sur les sandbox, etc. » résume Cédric Pottier.

Un framework pour tous !

Pour simplifier les choses, les équipes Salesforce ont créé un framework unique qui est utilisé par tous les services. Cela signifie que ce framework inclut l'ensemble des fonctions de sécurité selon les besoins.

Basiquement, nous trouvons, comme le disait Cédric Pottier plus haut, 3 niveaux : les services infrastructures, les services réseaux et les services applicatifs. Dans le détail, nous trouvons :

Infrastructure	Réseau	Application
Sécurité des datacenters Backup, plan de reprise après incident Réplication temps réel Certifications Audits des clients	HTTPS Tests de pénétration Détection proactive des failles Firewalls Restriction de login PI	Identification SSO Politique de mot de passe Authentification multifacteur Gestion des permissions Sécurité champs, formulaires, etc.

Naturellement, les couches infrastructures et réseaux sont gérées et sécurisées directement par l'éditeur. Par contre, la partie services applicatifs et les fonctions de sécurité optionnelles (encryption, moniteur des événements, sécurité mobile, etc.) sont de la responsabilité des utilisateurs finaux.

Par exemple, le chiffrement des données est par défaut côté Salesforce mais sur la partie utilisateur. Il est donc possible de les chiffrer côté front. Autre possibilité, récupérer et utiliser les logs pour pouvoir suivre l'état de fonctionnement et détecter les problèmes le plus tôt possible. A noter que toutes les fonctionnalités de sécurité sont accessibles en interface web.

La partie utilisateur

Quand on déploie des services Salesforce aux équipes d'une entreprise, il ne s'agit pas de le faire n'importe comment. Il faut pouvoir gérer les accès, gérer les utilisateurs et leurs droits d'accès (= les permissions), mettre en place les mécanismes d'authentification et d'identifications, etc. Il n'est pas question de faire un accès uniquement par un mot de passe. Qui accède à quoi et comment. Il faut donc paramétrer chaque utilisateur, autoriser ou non l'accès aux données et aux services. Salesforce simplifie cette mise en place mais il faut le faire en respectant les guidelines officiels. Par exemple, on va utiliser un annuaire couplé à du SSO tout en utilisant une authenfication multifacteur (MFA).

TRUST.SALESFORCE.COM : VISIBILITÉ DE L'ÉTAT DES SERVICES

Comme tout fournisseur SaaS et hyperscaleur, nous disposons d'un tableau de bord de la santé des services. Cette page indique l'état des services, par exemple, on peut voir les incidents et leur état, et des maintenances en cours ou planifiées. Cela concerne l'ensemble de l'écosystème Salesforce en incluant Heroku, Slack, etc.

Les pachts sont déployés en interne et à tous les services. Seuls les patchs importants sont documentés et indiqués. Ils sont indiqués sur le trust. Salesforce déploie 3 releases par an qui impactent de nombreuses couches.

MFA est disponible par défaut. Pour ce faire, il s'appuie sur une app mobile pour faire le multifacteur avec une gestion de mot de passe classique. On peut aussi faire du passwordless si les équipes le veulent. Par exemple, il est possible d'utiliser un FaceID pour déverrouiller un accès. Mais vous pouvez utiliser toute solution MFA.

Salesforce supporte par défaut SAMLv2, OIDC, OAuth2 et les certificats pour l'authentification.

Un clé unique générée à la demande

Les clés de chiffrement sont des clés RSA classiques. La clé symétrique n'est jamais stockée. Elle est générée à la demande quand on en a besoin. Pour ce faire, la génération utilise l'algorithme PBKDF2 qui est une dérivation de clé. Cela signifie que pour générer une clé, plusieurs paramètres sont pris en compte pour la créer par exemple d'autres clés, le salt, un hash, etc.

Cette génération est assurée par les HSM qui sont déployés près des serveurs.Le HSM, Hardware Security Module, est une appliance matérielle pour générer et stocker les clés de cryptographies. Cela signifie qu'il n'est pas possible de générer une clé depuis son OS local. Basiquement, le HSM est déployé entre le serveur et la base de données. Quand un service a besoin de lire une donnée, il doit la déchiffrer et pour ce faire, il va envoyer une requête au HSM qui renverrait la clé.

Le fonctionnement est totalement transparent.

LES ATTENTES CÔTÉ RSSI

« Typiquement, le RSSI a besoin de reprendre le contrôle des risques des applications Saas qui sont souvent vues comme une "boîte noire".Les risques sont nombreux : exfiltration de données, usages anormaux, connexion non autorisée, vols de compte, etc. On constate que les clients n'ont pas assez de visibilité sur la sécurité des services Saas. Quand un incident arrive c'est déjà trop tard pour agir. La question qui se pose alors et de savoir quelles données sont concernées. Le RSSI doit donc disposer d'outils » précise Cédric Pottier.

« La première réponse est la collecte des logs de la plateforme. Savoir qui fait quoi et à quel moment. Pour cela, beaucoup de clients connectent leur SOC à la plateforme Salesforce. Par exemple, 57 fichiers de logs sont disponibles traitant de la sécurité mais aussi de la performance et de l'usage métier. Le but est d'être pro-actif sur les usages anormaux et de générer des incidents précis. Par exemple, un employé qui partirait de l'entreprise avec la base de client serait détecté. La bonne pratique est de chiffrer les données sensibles au niveau applicatif. Beaucoup de clients demandent aussi à avoir une gestion des clefs de chiffrement externalisée. » poursuit Cédric.

DES SERVICES SUPPLÉMENTAIRES PAYANTES

Au-delà des sécurités par défaut et inclus avec son compte Salesforce, plusieurs addons additionnels sont disponibles. Par exemple, nous pouvons citer :

Event Monitoring : récupérer des logs pour tracer les problèmes et remonter l'incident, les services concernées ou encore voir les données impactées

Security Center : une sorte de tableau de bord pour avoir une vision d'ensemble de tous les orgs Salesforce, suivre le cycle de vie des permissions, envoyer des alertes, etc.

Cette sécurité évite qu'un hack puisse accéder librement aux données. D'autre part, tous les accès aux données passent la couche TLS (Transaction Layer Security). Cela signifie que toute transmission de données / informations est par défaut sécurisée. Et ensuite, nous pouvons chiffrer telle ou telle avec telle ou telle clé.

La partie développement : surveiller le backend externe

La plateforme permet des développements sur mesure, par exemple avec le langage Apex. Il dérive de Java. Les apps Apex s'exécutent sur Salesforce. La plateforme surveille par défaut les codes Apex. Par défaut, le langage est limité et évite d'introduire des failles. La plateforme de l'analyse statique. Seules les apps externes ne sont pas analysées par Salesforce.

Par exemple, si nous utilisons un service de la Marketplace utilisant un backend externe, c'est à nous de s'assurer de la sécurité sur cette partie externe. Ainsi, en cas de crash du backend externe, c'est à nous de s'assurer du backup et de la reprise après incident. L'API assurant la connexion est sécurisée par défaut via les solutions Mulesoft.

L'API permet de borner les accès. L'injection de code n'est pas possible.

Le cache local du navigateur n'est pas utilisé pour stocker les pages ou les données. Les pages sont générées à la volée et les données sont récupérées elle aussi à la volée.
« La sécurité c'est aussi dès la conception d'une application. Par exemple, les applications tierces et les développements spécifiques doivent codés en incluant la sécurité dès le départ.Un guide securité est disponible à l'attention des développeurs : https://developer.salesforce.com/developer-centers/security

Les développeurs suivent les meilleures pratiques de codage telles que celles spécifiées dans OWASP et CWE25. Tous les codes avant la mise en production sont examinés. Les outils de qualité et de sécurité du code sont exécutés fréquemment pour détecter d'éventuelles anomalies du programme. » explique Cédric Pottier.

IA générative & sécurité : les questions à se poser

François Tonic
Rédacteur en chef
Programmez!

Depuis 10 mois, on parle de ChatGPT, de GPT, d'IA générative. Mais une question se pose : quid de la sécurité dans les infrastructures, dans les données et envers les utilisateurs ? Mamadi Keita (Industry Senior Executive Advisor, Salesforce) évoque pour Programmez! les pistes et les garde-fous déployés chez Salesforce.

L'IA générative est une technologie, et une fonctionnalité, désormais incontournable. Salesforce l'intègre dans l'ensemble des couches et des services, même si l'éditeur n'a pas attendu ChatGPT pour intégrer de l'IA (sous-entendu du machine learning). Dans une plateforme telle que Salesforce, où la donnée est omniprésence, l'IA générative devient un outil de production. Elle est capable d'interroger les bases de données, les documents, les fichiers. C'est une promesse de productivité et d'efficacité.

Comment lui faire confiance ? *« La notion de confiance signifie qu'il faut éviter les fuites de données sensibles, mais elle concerne aussi les utilisateurs dont l'usage du système doit être sécurisé. L'IA est un service informatique que l'on met à disposition. Les utilisateurs ne doivent pas se poser de questions sur la sécurité. Le service doit être fiable. » introduit Mamadi.*

« Ce n'est pas nouveau sur notre plateforme. L'IA existe depuis 2016. La nouveauté est la partie générative. Comment la fournir de manière sécurisée à l'entreprise à travers notre plateforme? Nous rajoutons des modules pour travailler la productivité, pour renforcer la confiance envers nos services. Nous cryptons, nous vérifions les comportements. Nous intégrons aussi des modèles de LLM. Nous n'imposons pas un modèle. L'avenir est d'avoir x modèles selon les usages. Nous voulons une plateforme. » complète Kheira Boulhila (Senior Vice President Solutions Engineering EMEA chez Salesforce).

Le socle technique des différents IA est Einstein. Tout naturellement, le tout repose sur l'infrastructure Hyperforce et donc, les services IA utilisent les mêmes fonctionnalités de sécurité. Les données sont cryptées. *« Nous rajoutons un traitement d'anonymisation des données sensibles au niveau de la couche applicative. Ce traitement est appliqué sur le prompt avant de l'envoyer au service d'IA génératif. Le flux et la consommation sont faits via des API, HTTPS, etc. »* poursuit Mamadi.
Il y a une partie liée à Hyperforce et une partie à la couche de confiance.
« Dans la plateforme, nous avons la capacité de qualifier les données, mesurer le niveau de criticité et le niveau de confidentialité » explique Mamadi. Cela signifie que l'IA générative ne doit pas utiliser des données interdites ou à accès limité.

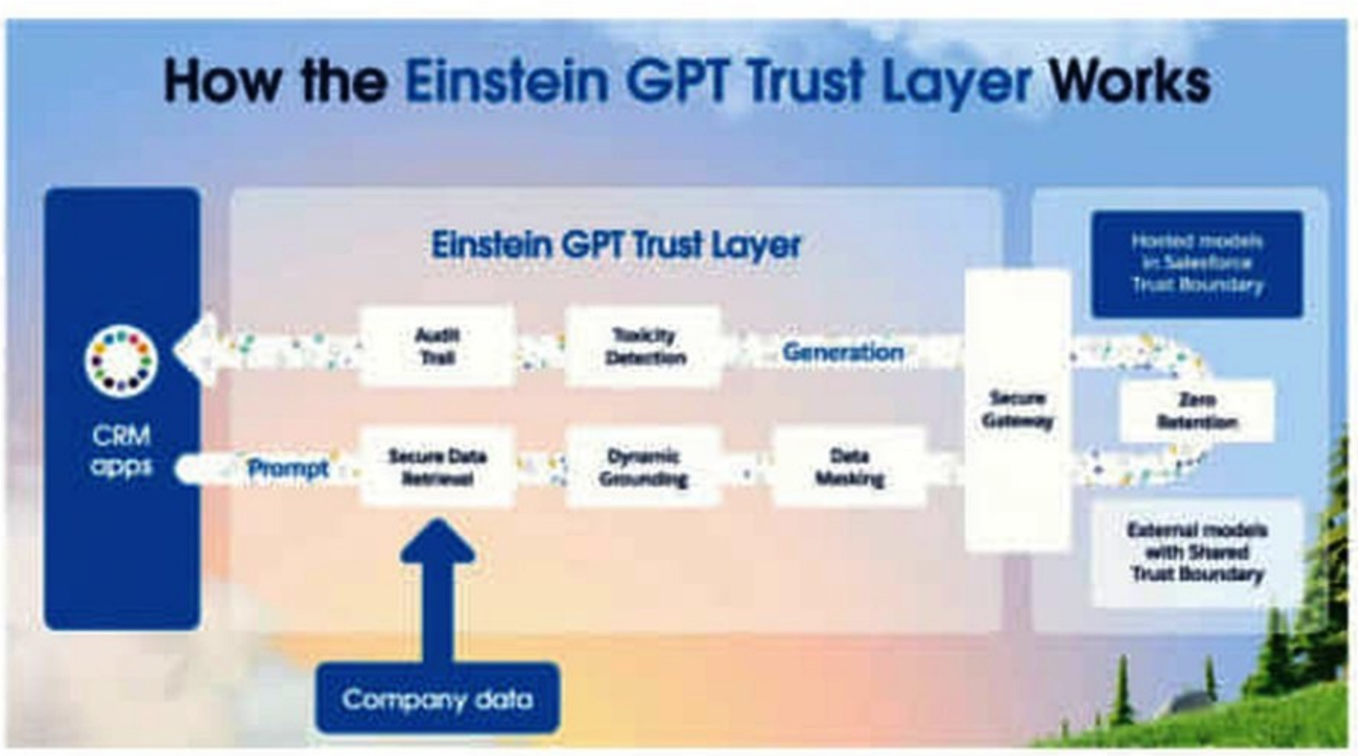
Des couches de confiance et de sécurité

Au niveau de la couche Einstein GPT Trust Layer, tout est fait pour sécuriser les flux, les accès aux données autorisées, mais aussi de qualifier le résultat du prompt. Nous avons un catalogue de prompts et un traitement de notre Trust Layer va agréger des données de la plateforme pour enrichir le prompt avec des informations de contexte. C'est le dynamic grounding. *« Nous avons la maîtrise de la sécurité sur cette partie. Nous appliquons du masking : remplacer les données sensibles par une donnée générale fictive tout en reconstituant la donnée initiale à la fin du processus. »*

Dans le cas de l'utilisation du service OpenAI, donc en service externe, Salesforce a des accords avec OpenAI. *« Chaque prompt est détruit après la génération d'une réponse. C'est ce que nous appelons la non-rétention. D'autre part, nous faisons des audits chez OpenAI pour la garantir. »* cadre Mamadi. D'autre part, les prompts générés sont analysés pour voir des résultats n'utilisent des termes ou des réponses interdites (= notion de toxicité du prompt).

« Il existe des standards que nous mettons en oeuvre pour garantir la sécurité et la bonne programmation des prompts dans notre outil Prompt Studio. Nous devons prévenir des risques autour du prompt injection, comme l'injection de code. Potentiellement, on pourrait faire dire tout et n'importe quoi. Dans le Prompt Studio, il existe un mode avancé pour contrôler des paramètres telle que la capacité à avoir des réponses variées avec le même prompt. » explique Mamadi.

Existe-t-il un risque sur le caching du navigateur et les DevTools ? Là encore, les couches IA reposent sur les mêmes mécanismes de sécurité et de flux que les autres services de la plateforme. Aucune donnée ou état de session ne sont stockés dans le cache.

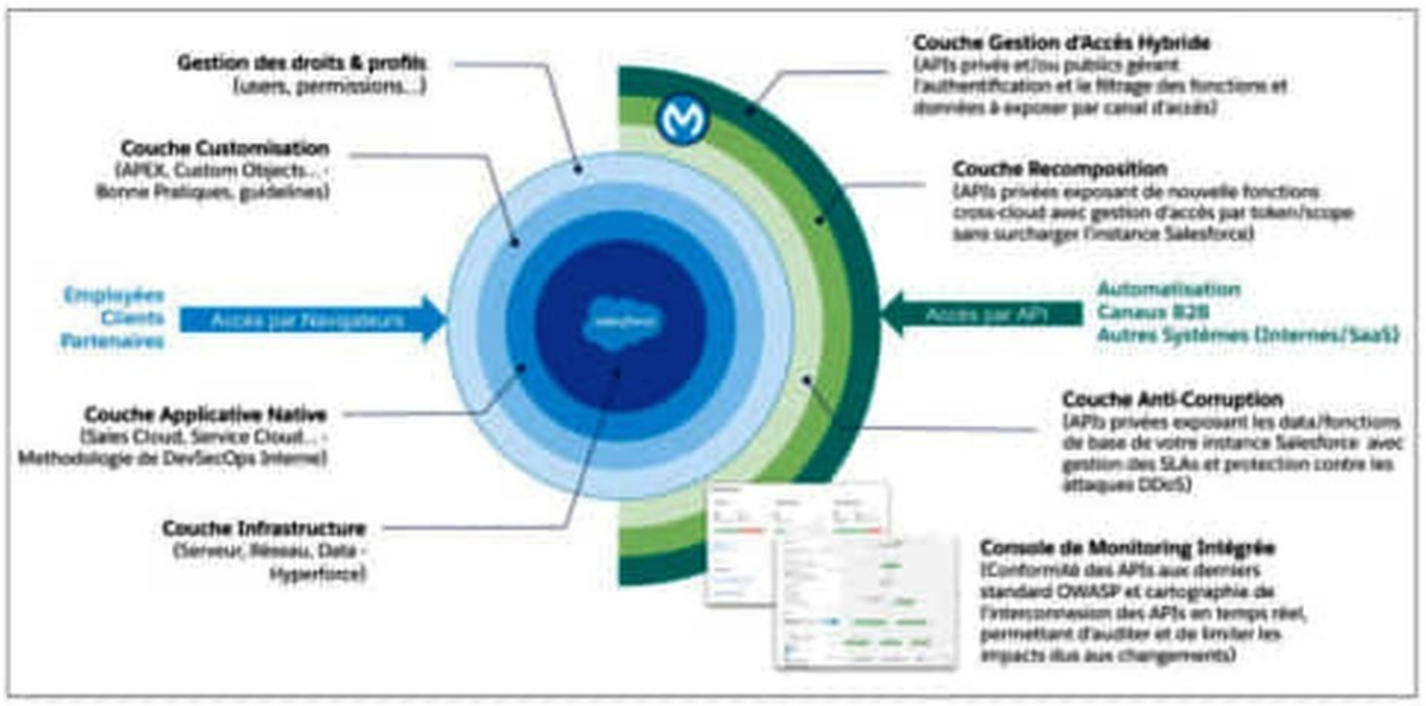


Sécurisation des API (Zéro Confiance)

Kiet Yap

Distinguished Client Architect & Security SpecialisMuleSoft

Comme nous venons de le découvrir, la sécurité est au cœur de la plateforme Salesforce, depuis la couche d'infrastructure jusqu'à la couche d'engagement avec une gestion d'accès complète exposée au public. Et avec la richesse fonctionnelle que la plateforme fournit, il est naturel de vouloir capitaliser sur ces ressources pour apporter une meilleure synergie avec le reste de votre écosystème, ou même vos partenaires.



Pour créer cette synergie, l'adoption des API au niveau de Salesforce et de vos systèmes externes, qu'ils soient dans le Cloud ou dans votre datacenter, est certainement la marche à suivre. Ceci permet de mettre en commun les ressources métiers disponibles, et avec partage et réutilisation de ces API nous pouvons en créer d'autres à forte valeur ajoutée qui bénéficieront à votre organisation. Mais avec cette couche supplémentaire comment sécuriser ces interactions entre ces systèmes divers dont vous avez plus ou moins une visibilité sur leur niveau de sécurité intrinsèque, et localisé potentiellement dans des zones de réseau plus ou moins sécurisées ?

Pourquoi un API Gateway ne suffit pas

La tendance actuelle du marché pour adresser cette problématique est "mettons un API Gateway", et par magie tous les problèmes de sécurité seront résolus. Oui, les API Gateways sont nécessaires, mais ce n'est pas suffisant. En effet, par philosophie, un API Gateway permet sécuriser les trafics au niveau "périmétrique" dans la frontière "Nord-Sud", où le Nord est catégorisé comme la zone de réseau / applications à risque, et le "Sud" la zone de réseau avec des ressources API exposées à protéger. Chaque communication par API doit obligatoirement passer à travers cet API Gateway afin d'appliquer les règles de sécurité prédéfinies.

Mais dans un contexte Cloud/Hybride en général, avec ou sans Salesforce, comment définissons-nous le Nord et le Sud ? En effet, dans un contexte Cloud/Hybrid nous allons nous retrouver avec une multitude de frontières, où le Sud peut potentiellement être le Nord de quelqu'un d'autre. Bien entendu on pourra placer autant de Gateways que nécessaire au niveau de chaque frontière, mais cela aura un impact opérationnel et d'intégrité énorme à chaque changement. Et quid des interactions Nord/Nord et Sud/Sud dans un même périmètre, qui sont toujours possibles, ou même encourager dans un contexte de partage et de réutilisabilité ? L'adoption d'un API Gateway dans ce contexte va obligatoirement engendrer des "rebonds" réseaux qui nuit non seulement à performance, mais également une valeur sécuritaire limitée car les API sont adressables directement depuis le même réseau.

De la Sécurité "Périmétrique" à la Sécurité "Zéro Confiance"

Avec ce nouveau paradigme, il nous faut une nouvelle philosophie pour sécuriser ces API qui opèrent dans des contextes beaucoup plus hétérogènes. Il ne s'agit pas uniquement de la technologie, mais d'une vraie méthodologie avec un pattern de design d'API où chaque API doit avoir un rôle bien défini et de pouvoir appliquer les sécurités directement depuis son point d'entrée, sans faire confiance à aucun mécanisme de sécurité externe ou périmétrique. Cette approche Zéro Confiance permet d'adopter une posture de sécurité plus homogène et évolutive, dans un contexte de déploiement hétérogène.

MuleSoft Anypoint Platform, que Salesforce a acquis en 2018, propose une plateforme complète de gestion d'API suivant ce concept Zéro Confiance. Appliqué sur Salesforce (ou n'importe quels systèmes de votre organisation), Anypoint Platform propose une sécurité bout en bout sur 3 couches API avec des outils de gestion et de pilotage intégrés pour faciliter l'adoption et l'innovation par API :
https://www.mulesoft.com/platform/api-security

Pour plus d'information sur l'adoption de cette approche, un article en 2 parties a été développé dans de précédents numéros, couvrant les aspects suivants :

- Limitation de la Sécurité Périmétrique
- Appliquer le concept Zéro Confiance à l'échelle
- Identification et hiérarchisation des API avec Domain Driven Design
- Adoption de l'API Risk Matrix pour identifier les règles de sécurité à appliquer

Pour aller plus loin sur la plateforme Salesforce Einstein1

Salesforce a dévoilé sa nouvelle architecture Einstein 1 Platform lors de Dreamforce en septembre 2023. Cette nouvelle plateforme introduit des avancées majeures conçues pour Data Cloud et l'IA Einstein en s'appuyant sur le framework de métadonnées de Salesforce. Ainsi les entreprises connectent leurs données librement et en toute sécurité, afin de développer des applications low-code alimentées par une intelligence artificielle, et de proposer des expériences de CRM révolutionnaires.Voyons les ressources disponibles.

Les ressources de Salesforce sont multiples pour tous ceux qui souhaitent approfondir leurs connaissances, être en lien avec les communautés et bien entendu installer des applications tierces complémentaires.

Trailhead, le site d'autoformation gratuit de Salesforce et de sa communauté d'experts "Les Trailblazers" https://trailhead.salesforce.com/

On estime que la communauté technique de Salesforce "Les Trailblazers" regroupent près de 100 000 personnes en France.

Il existe différentes façons de commencer un parcours d'apprentissage sur Trailhead. Voici quelques-unes des catégories d'éléments :

- Les modules sont de petites unités qui présentent des sujets spécifiques de manière guidée.
- Les projets aident à acquérir de nouvelles compétences via la réalisation d'exercices pratiques dans le Trailhead Playground (avec l' environnement de développement personnel) en donnant des instructions détaillées pendant que vous travaillez.
- Les parcours sont des programmes de formation guidés, composés de modules et de projets qui aident à explorer des sujets, à acquérir des compétences pertinentes et à se familiariser avec les produits Salesforce.
- Les trailmix sont des programmes de formation personnalisés sur lesquels il est possible de créer ses parcours, modules, projets préférés et plus encore au trailmix. Il peut être partagé à d'autres Trailblazers.
- Les superbadges permettent de mettre en application les nouvelles compétences acquises pour résoudre des problèmes métiers concrets.
- Les certifications Salesforce font office de preuve de son expertise et de son expérience pratique quant à l'utilisation des produits Salesforce. Elles permettent d'enrichir un CV et de valider vos compétences.

AppExchange, une des plus grandes plateformes d'échange et d'applications tierces dans le monde professionnel

appexchange.salesforce.com

Avec plus de 12 millions de téléchargements à ce jour, AppExchange a quelque chose à offrir pour chaque Cloud et produit Salesforce. Toutefois, en règle générale, deux types de références sont proposés : des solutions et des consultants.

- Une solution est un élément qui se raccorde à Salesforce et élargit ses fonctionnalités. Elle peut être gratuite ou payante.
- Les consultants sont des professionnels Salesforce spécialisés dans l'élaboration et l'implémentation de solutions personnalisées pour des Clouds ou des secteurs d'activité spécifiques.

Les ressources conseillées pour en savoir plus sur la plateforme Einstein1

Les formations sur Trailhead

Titre	Thèmes abordés	Lien
Generative AI Basics	Les principes fondamentaux de l'IA générative	https://trailhead.salesforce.com/content/learn/modules/generative-ai-basics
Prompt Fundamentals	Les notions de base sur le Prompt, apprendre à faire un prompt	https://trailhead.salesforce.com/content/learn/modules/generative-ai-basics
Large Language Models	Tout savoir sur les LLM, et comment Salesforce utilise les LLM	https://trailhead.salesforce.com/content/learn/modules/large-language-models
Get Started with Einstein & Generative AI	Décrire les capacités essentielles de l'IA générative. Décrire comment Einstein intègre la puissance de l'IA générative.	https://trailhead.salesforce.com/content/learn/modules/einstein-gpt-quick-look/get-started-with-einstein-gpt
Meet the Einstein Trust Layer	Ce module aborde certaines fonctionnalités futures de la couche de confiance d'Einstein.	https://trailhead.salesforce.com/content/learn/modules/the-einstein-trust-layer/meet-the-einstein-trust-layer
Protection de vos applications avec Salesforce Shield	Apprenez à mettre en place une gouvernance et à appliquer des stratégies de conformité au sein de votre organisation à l'aide de cet ensemble puissant de services	https://trailhead.salesforce.com/fr/content/learn/trails/shield
Dynamic and Secure MuleSoft RPA Process Implementation	Ajouter des fonctionnalités plus robustes et plus sûres à un processus RPA	https://trailhead.salesforce.com/content/learn/modules/dynamic-and-secure-mulesoft-rpa-process-implementation
Secure Salesforce Configuration	Configurez votre organisation Salesforce en tenant compte de la sécurité	https://trailhead.salesforce.com/content/learn/modules/secure-salesforce-configuration

Les liens utiles du site Salesforce

Titre	Thèmes abordés	Lien
Explorer la plateforme Salesforce Einstein1	Annoncée à Dreamforce 2023, overview de la nouvelle infrastructure de la plateforme Salesforce	https://www.salesforce.com/products/platform/overview/
Trust & Security	Guide de sécurité de Salesforce	https://security.salesforce.com/fr
Salesforce AI Research	Le site de référence de Salesforce sur l'Intelligence Artificielle	https://www.salesforceairesearch.com/about
Trusted AI	Le site de référence pour comprendre comment Salesforce développe son Intelligence Artificielle sur des principes de confiance	https://www.salesforceairesearch.com/trusted-ai
Generative AI: 5 Guidelines for Responsible Development	Les recommandations de Salesforce	https://blog.salesforceairesearch.com/generative-ai-5-guidelines-for-responsible-development/
La place de marché d'applications AppExchange	Le site de solutions tierces autour de l'adminsitration et la sécurité	https://appexchange.salesforce.com/category/it-admin
La documentation de chaque produit	Chercher en particulier Hyperforce, Salesforce Shield et Marketing Cloud Shield	https://help.salesforce.com/s/
Brochure produit Salesforce Shield	Tout savoir sur les outils de protection de données	https://www.salesforce.com/products/platform/products/shield/
Brochure produit Mulesoft Anypoint Platform	Comment protéger ses API	https://www.mulesoft.com/platform/api-security
Brochure Mulesoft Security	Cryptographie, configuration de la sécurité, FIPS 140-2, TLS, config et bien d'autres choses	https://docs.mulesoft.com/mule-runtime/4.3/securing?_ga=2.260192589.1395495959.1695028622-1972771478.1679905400

Open Source et sécurité

Sylvain Cailliau

Évangéliste et Expert Technique, CAST France

Sylvain est spécialiste de l'ADM (Application Development Management), des méthodes Agiles et de DevOps. Il accompagne les Directions informatiques dans leur transformation grâce à une connaissance approfondie des systèmes complexes et sa prise en compte des enjeux métier. Sylvain est Ingénieur de formation à SUPINFO, sa devise est "In Code we Trust".

CAST
Software Intelligence for Digital Leaders

L'utilisation de composants Open Source est une bonne pratique reconnue, leur utilisation accélère les développements et leur qualité augmente régulièrement. Cependant le code source de ces composants n'est en général ni connu ni maîtrisé par les développeurs qui les utilisent et, au même titre que le code « fermé » (développé en interne, ou mis à disposition par des éditeurs), il est susceptible d'introduire des vulnérabilités de sécurité dans le Système d'Information. C'est d'ailleurs en étudiant le code source, qui est public, que certains malveillants vont trouver des failles et le moyen de les exploiter ou plus directement encore introduire dans les dépôts de code public des logiciels qu'ils ont eux-mêmes infectés. Les sujets de l'Open Source et de la Sécurité sont donc liés et du point de vue de la Sécurité, on doit assurer une gestion serrée du code Open Source.

L'Open Source est partout

D'après une étude de l'organisation OPEN UK (State of Open(1) publiée en 2021), 97% des entreprises utilisent aujourd'hui du logiciel Open Source sous une forme ou une autre (incluant les systèmes d'exploitation et les langages de programmation). Github fait état dans son rapport Octoverse 2022 The state of Open Source(2) de plus de 413 millions de contributions de la part des développeurs en une seule année. Le nombre de téléchargements depuis les principaux dépôts de code Open Source se compte en centaines de milliards par année. Considérez simplement que le composant « github.com/jonschlinkert/is-odd » est téléchargé actuellement plus de 450.000 fois par semaine. **Figure 1**

Figure 1

Depuis 2014, la qualité est devenue la principale raison du choix de l'Open Source par les développeurs. Les progiciels Open Source offrent la transparence en ouvrant leur code source - ce qui est très important pour garantir la sécurité notamment - et défient les éditeurs de progiciels qui n'offrent pas forcément cette transparence. Selon l'étude The state of Enterprise Open Source(3) publiée en 2022 par Red Hat, 89 % des responsables informatiques pensent que l'Open Source est aussi sécurisé ou plus sécurisé que les logiciels propriétaires (APAC = 89%, EMEA = 90%, LATAM = 87%, US = 90%). Dans cette même étude, les raisons invoquées pour s'être formé une telle conviction sont les suivantes :

- Pour 55% des décideurs : Mon équipe peut utiliser du code Open Source bien testé pour construire nos applications développées en interne.
- Pour 52% : Les correctifs (patches) de Sécurité sont bien documentés et peuvent être scannés.
- Pour 51% : Les communautés mettent les correctifs de Sécurité à disposition rapidement.
- Pour 44% : Une population plus large que ce n'est le cas pour les logiciels propriétaires a pu examiner ces composants.
- Pour 38% : Mon équipe peut en auditer le code.

Enfin, c'est aussi un facteur de collaboration et d'innovation pour l'ensemble du Marché: dans le rapport 2023 State of Open Source(4) d'OpenLogic, l'une des statistiques importantes est que 36,79 % des organisations contribuent à l'Open Source. Il s'agit d'une augmentation de 5 % par rapport à l'année dernière. De nombreux projets innovants (Cloud, Big Data, IoT...) ne sont menés que sur des plateformes Open Source et en mode collaboratif.

Le revers de la médaille

La réalité n'est cependant pas faite uniquement de bonnes nouvelles. De nombreux sujets de préoccupations relatifs à l'utilisation de l'Open Source doivent alerter les décideurs comme les développeurs, car le constat est saisissant :

- 87% des logiciels embarquent au moins un composant Open Source porteur d'une vulnérabilité de sécurité (selon le rapport OSSRA - Open Source Security and Risk Analysis Report 2023(5)
- 88% de croissance des vulnérabilités en 2 ans et même jusqu'à 557% de croissance sur 5 ans dans le secteur de la Distribution et du e-Commerce (toujours selon le rapport OSSRA 2023)
- 37% des développeurs Open Source ne mettent pas en œuvre de tests de sécurité,
- 54 % des développeurs ne réalisent aucun test de sécurité sur les images Docker (les 10 images Docker les plus populaires embarquent au moins 30 vulnérabilités de sécurité),
- 2 ans en moyenne entre la découverte d'une vulnérabilité et sa correction,
- 1 développeur sur 3 n'a pas les compétences nécessaires pour résoudre les problèmes de vulnérabilité.
- Une pratique consistant à mettre à disposition dans les dépôts Open Source de logiciels porteurs de vulnérabilité (originaux, mais aussi souvent des copies de logiciels populaires) se développe depuis 2016 (voir l'article de JFROG relatif à une attaque massive du dépôt NPM ciblant les développeurs AZURE(6)).
- Plusieurs centaines de types de licences portant sur des composants Open Source avec de très grandes disparités en termes d'obligations en cas de réutilisation.

(1) State-of-Open-Phase-Two.pdf (openuk.uk)

(2) Octoverse 2022: The state of open source | The State of the Octoverse (github.com)

(3) The State of Enterprise Open Source: A Red Hat report

(4) 2023 State of Open Source Report - Success | OpenLogic by Perforce

(5) 2023 OSSRA Report (synopsys.com)

(6) Malicious Packages in npm Targeting Azure Developers (jfrog.com)

Les organisations interrogées dans les diverses études indiquent clairement les difficultés particulières qu'elles doivent résoudre face à l'extrême prégnance de l'Open Source dans tous les systèmes d'information modernes. Pour ne citer que l'étude d'OpenLogic qui pose la question « Quels sont les principaux défis de votre organisation pour supporter l'utilisation de logiciels Open Source ? », les réponses sont éloquentes, et le top 5 est :

- Maintenir les politiques de sécurité ou la conformité pour 42%
- Pallier le manque de compétences, d'expérience et de maîtrise pour 38%
- Assurer le suivi des mises à jour et correctifs pour 37%
- Pallier le manque de support technique de bas niveau pour 36%
- Gérer la maintenance des versions en fin de vie pour 36%

Les raisons évoquées sur l'intérêt de l'Open Source dans le paragraphe précédent ressemblent donc plus à des souhaits qu'à une réalité : la confiance dans le code Open Source s'appuie sur la capacité à pouvoir auditer ou scanner le code, mais les organisations manquent de personnel compétent pour le faire vraiment !

Quand on parle de risque Open Source, de quoi parle-t-on ?

L'Open Source apporte de grands avantages, mais les risques ne doivent pas être minimisés et la confiance doit forcément s'appuyer sur le contrôle. Si l'Open Source est ouvert, il est également ouvert aux hackers qui en profitent pour exploiter les faiblesses qu'ils découvrent en analysant le code des composants Open Source. Les risques ne sont pas seulement des risques de sécurité, il existe en fait 3 types de risque :

- Le risque lié à l'obsolescence
- Le risque sur la Propriété intellectuelle (lié aux licences portant les composants OSS)
- Le risque de Sécurité

Le Risque lié à l'obsolescence

Comme pour les logiciels propriétaires, les communautés Open Source qui font évoluer les logiciels apportent des corrections sur les dernières versions des composants. Afin d'intégrer rapidement ces corrections, il est donc préférable de mettre à jour régulièrement les composants. Malheureusement, la pratique observée au sein des équipes de développement est la suivante : une équipe choisit presque systématiquement la dernière version d'un composant pour l'intégrer dans un nouveau développement (parfois même en choisissant une version qui n'est qu'une "release candidate") et ne met presque jamais à jour ce composant par la suite. Nous rencontrons même parfois des dépendances de deux versions différentes du même composant dans la même application. Les deux captures d'écran réalisées à partir de la solution Highlight de CAST illustrent ces comportements : **Figure 2**

Nous voyons ici que le composant "Commons-fileupload" est utilisé par 14 applications d'un portefeuille client en 6 (!) versions distinctes. On constate même parfois qu'une même application utilise deux ou plusieurs versions différentes d'un même composant. **Figure 3**

Et dans l'écran ci-dessus, qui affiche la chronologie d'un composant, nous voyons que la version utilisée est la version

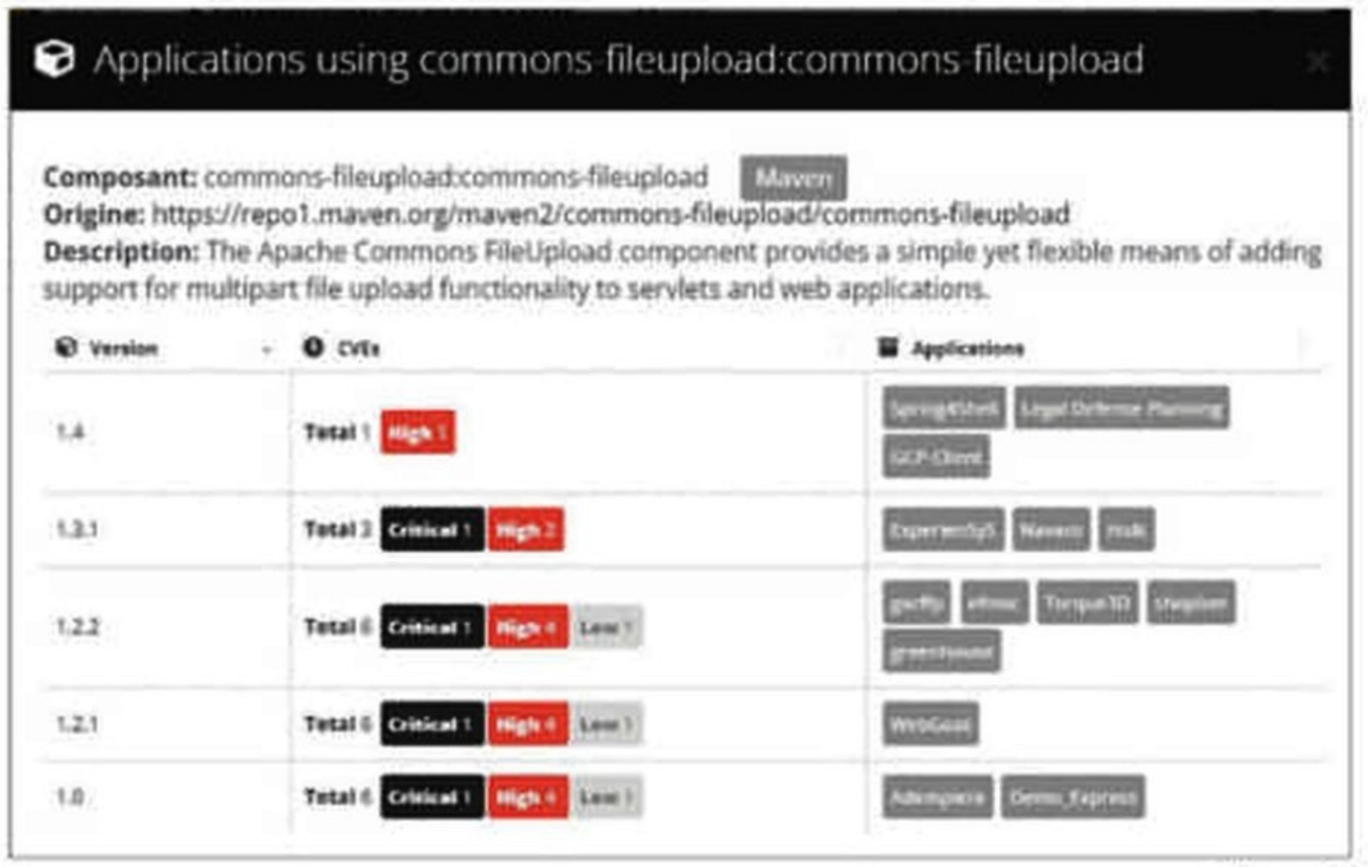


Figure 2

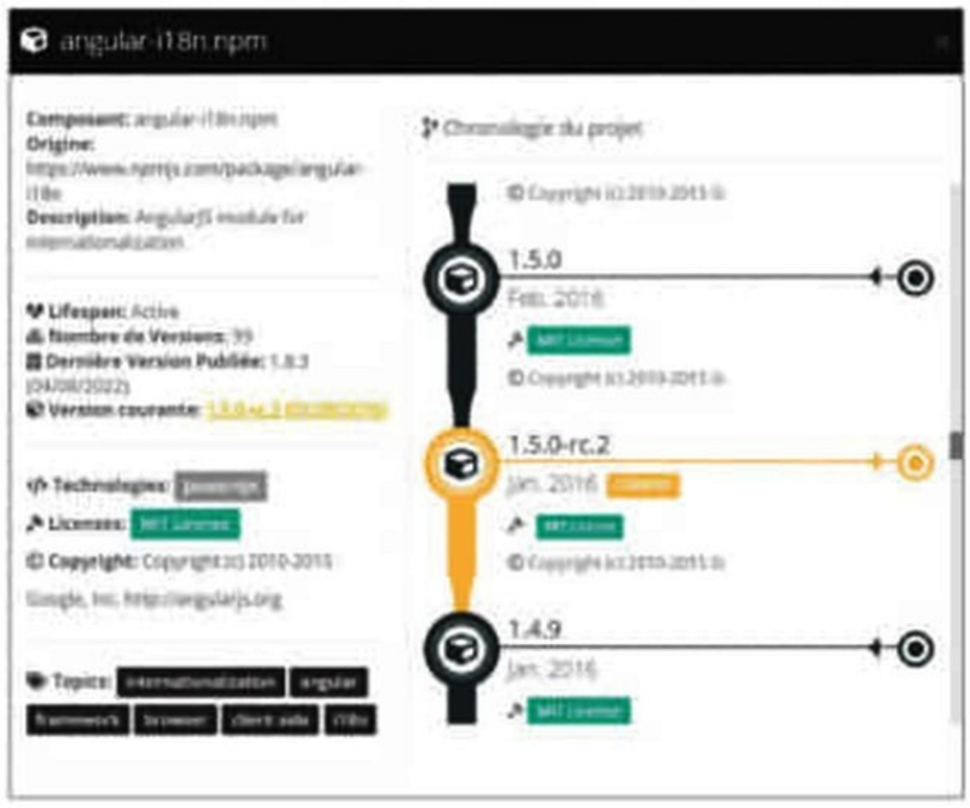


Figure 3

de janvier 2016 alors que la version la plus récente date d'août 2022. De plus, la version utilisée est une version 1.5.0 "release candidate" alors que nous pouvons clairement voir sur la timeline que la version 1.5.0 "officielle" a été mise à disposition en février 2016 (l'équipe aurait pu facilement intégrer la version officielle sans réels coûts de tests supplémentaires).

Le Risque sur la propriété intellectuelle (ou risque de licence)

L'Open Source n'est pas synonyme de "gratuit" ou "sans règle", bien au contraire : l'Open Source est un mode de diffusion des logiciels qui permet d'accéder au code source, de le modifier et de le redistribuer, mais uniquement sous certaines conditions et contraintes.

La nature des licences qui portent les composants Open Source que vos développeurs ont intégré et la manière dont les applications sont utilisées ou distribuées, vous contraignent et vous soumettent aux lois régissant la propriété intellectuelle. Un exemple bien connu est celui de FREE qui a intégré dans ses Freebox des logiciels Open Source distribués sous licence GPL v2 sans rendre le code source public. En conséquence elle a été poursuivie par la Free Software Foundation devant le Tribunal de Grande Instance de Paris avec une demande de 1€ de dommages et intérêts par abonné Freebox (3 millions). Un accord amiable, pour un montant non divulgué, a été conclu en 2011.

En outre, comme le nombre de licences est très élevé (le

groupe de travail SPDX maintient une liste des licences connues et leur texte intégral — https://spdx.org/licenses/) : au 9 février 2020, la liste comprenait 381 licences différentes, et comme certains logiciels peuvent changer de licence entre les versions, il faut être particulièrement prudent. En effet, ce risque de licence, souvent mal compris, s'avère complexe à gérer (composants intégrés dans une application, mais avec des licences incompatibles entre-elles ou risques transitifs — un composant OSS intègre un autre composant OSS, ce qui "transfère" les obligations de la licence associée à tout l'ensemble).

Sur la **Figure 4**, on peut voir que le composant "commons-fileupload" repose sur 4 autres composants et que l'un d'entre eux possède une licence potentiellement "toxique" (en rouge).

Le Risque de sécurité

Comme tout logiciel, le code Open Source peut intégrer des failles de sécurité. Un certain nombre d'organisations à but non lucratif telles que le NIST (National Institute of Standards and Technology : https://www.nist.gov/) ont décidé de collecter et de rendre public les failles de sécurité découvertes par des utilisateurs du monde entier qui les signalent. La base de données du NIST s'appelle National Vulnerability Database (NVD : https://nvd.nist.gov/). Chaque faille de sécurité est décrite en détail, avec des liens vers les moyens possibles de résoudre ou contourner chaque faille (appelée CVE : Common Vulnerability and Exposure). Les CVE sont évaluées à l'aide d'un indicateur calculé selon la méthodologie CVSS (Common Vulnerability Scoring System). En fonction du score CVSS, un niveau de criticité est attribué aux failles de sécurité connues.

Dans la **Figure 5**, nous voyons la synthèse des failles de sécurité, découvertes dans un portefeuille de 232 applications, classées par niveau de criticité.

Figure 4

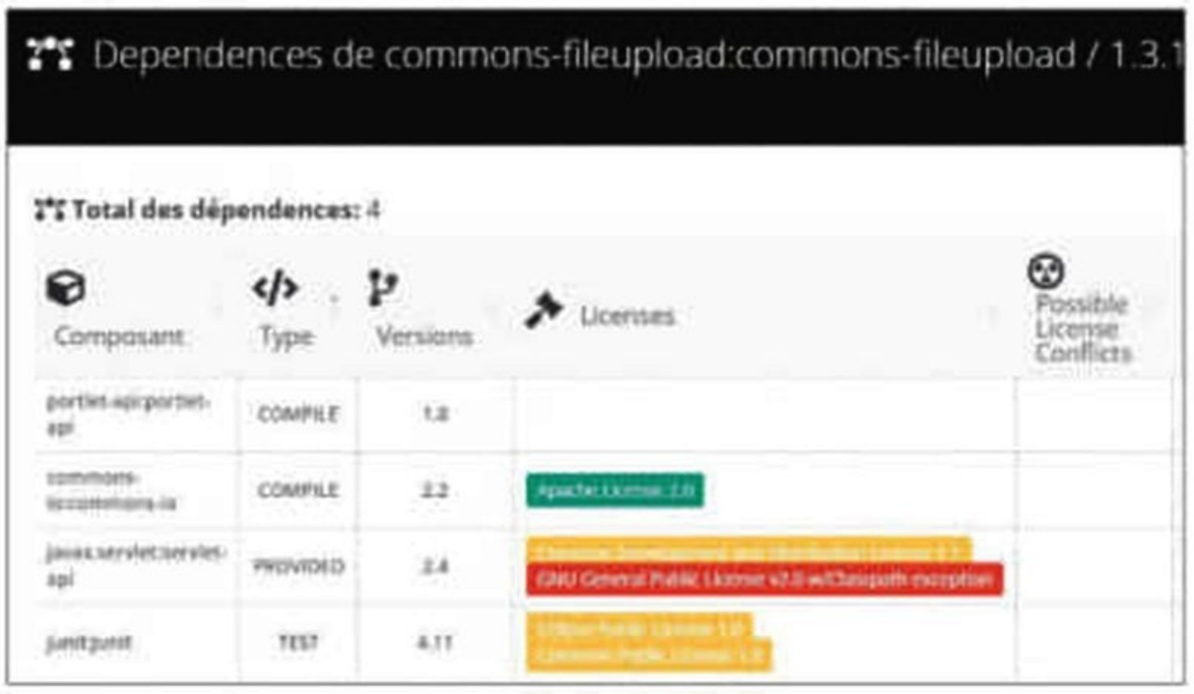


Figure 5

Mettre en œuvre les moyens nécessaires : un jour dans la vie des équipes (du développement à l'exploitation).

Alors que l'Open Source offre de nombreux avantages indispensables au développement moderne d'applications, le fait que ce code ne soit pas développé en interne rend impératif la mise en place une gestion outillée de tout le code Open Source embarqué dans le système d'information tout au long du cycle de vie. Il reste surprenant de voir que même des vulnérabilités bien connues ayant fait les gros titres de la presse en raison de leur niveau élevé de risque et pour lesquelles il existe des solutions perdurent dans le code des applications utilisées par des dizaines de milliers d'entreprises.

En décembre 2021 une vulnérabilité de sécurité cotée 10/10 selon la norme CVSS était détectée dans le composant Log4j. Un composant Open Source très populaire. Deux articles pour vous rappeler que « des centaines de millions d'appareils et de logiciels reposant sur Log4j sont mis en péril » : Bigdata.fr (Log4Shell : tout savoir sur la faille de cybersécurité la plus grave depuis 20 ans (lebigdata.fr)) et le Figaro.fr (Log4shell : cinq questions pour comprendre le vent de panique autour de cette faille critique (lefigaro.fr)). Bien qu'une suite de corrections ait été fournie rapidement, le constat plus d'un an après est édifiant : au 1er décembre 2022 le magazine IT for Business constatait que « 72% des entreprises restent exposées(7) ». Ce chiffre montre bien que si l'Open Source offre la transparence encore faut-il en profiter et connaître quel code Open Source est embarqué dans vos applications.

Pas d'autres solutions pour être efficaces que d'outiller la supply chain logicielle de bout en bout, car la réactivité est essentielle en matière de cyberdéfense face à des attaquants qui sont toujours très réactifs quand une faille est rendue publique (l'article de Bigdata.fr rapporte que : « Quelques heures seulement après l'annonce publique de cette vulnérabilité Log4Shell, des criminels scannaient les systèmes et déployaient des malwares).

Le contrôle de l'Open Source s'organise donc tout au long du cycle de vie (pas seulement du cycle de développement) de chaque application : Développement, Déploiement et Utilisation.

Comment cela se passe-t-il pour le développeur

Une équipe de développement se voit confier la maintenance ou le développement d'une application. Après avoir avancé sur le design fonctionnel et défini les besoins technico-fonctionnels pour atteindre les objectifs de business, l'équipe de développement va prendre la décision de réutiliser des composants Open Source pour accélérer les développements (gestion de l'interface homme machine via un framework MVC, de l'indépendance des données via un ORM, d'éléments fonctionnels génériques – comme le téléchargement de fichier depuis les PC ou Mobile des utilisateurs). Classiquement les développeurs vont rechercher les compo-

(7) Un an après, 72% des entreprises restent exposées à Log4Shell (itforbusiness.fr)

sants dans les dépôts de code Open Source disponibles (Maven, Pipy, Npm etc...). C'est dès ce moment qu'il faut fournir aux développeurs les moyens de choisir judicieusement les composants Open Source qu'ils intègrent dans l'application. Pour impliquer les développeurs dès le début de l'intégration des composants dans le code (Shift left), trois moyens sont classiquement utilisés :

1 Le développeur consulte directement une base de référence des risques lorsqu'il recherche un composant qui conviendra à son besoin dans l'un ou l'autre des dépôts disponibles via par exemple un plug-in dans le browser Internet : c'est ce type d'implémentation qui a été choisie pour CAST Highlight comme le montre la **Figure 6.**

L'ensemble des informations de la base de référence est accessible en un coup d'œil, que ce soit le nombre et la gravité des vulnérabilités connues pour l'ensemble des versions disponibles ou l'une d'entre-elles en particulier, la ou les licences qui protègent la propriété intellectuelle du composant ou encore l'obsolescence d'une version ou le niveau d'activité de la communauté vis-à-vis du composant (ici pas de nouvelle version depuis plus de 12 mois par exemple).

2 Le développeur consulte la base de référence depuis son environnement de développement intégré (IDE) en continu. Un plug-in pour vérifier et valider directement dans son environnement de travail quotidien comme celui proposé par CAST Highlight est directement disponible (ici dans Microsoft Visual Source Code) : **Figure 7**

3 Les différents éléments de code développés par les membres de l'équipe sont intégrés via une chaîne de fabrication automatisée (le pipeline CI/CD). Le contrôle de l'Open Source est intégré au pipeline et le résultat fourni à chaque développeur concerné (il est aussi possible de définir une douane qui fera échouer la fabrication si des vulnérabilités critiques sont détectées). Pour outiller ce type de contrôle, CAST Highlight peut être déclenchée en mode commande en ligne et/ou être provisionner à la volée : CAST Highlight est disponible forme de conteneur Docker. **Figure 8**

Intégrer ce contrôle aux étapes clés du cycle de fabrication est une solution très efficace pour s'assurer que l'assemblage des différentes parties d'une application reste toujours maîtrisé mêmes lorsque ces différentes parties sont sous la responsabilité de plusieurs équipes.

Il s'agit également d'assurer une gestion raisonnée des composants au niveau organisationnel : la mise en place, au sein de l'organisation, d'une fonction de type Open Source Program Office (OSPO) renforce le contrôle de l'Open Source tout au long du cycle de vie des applications. Cela permet d'orchestrer ce contrôle entre toutes les équipes de développement en laissant ces équipes se concentrer sur le développement lui-même tout en harmonisant les politiques de gestion pour l'ensemble d'un patrimoine applicatif après les avoir bien définies.

Ce groupe (OSPO) va pouvoir aussi développer des compétences vis-à-vis de la gestion des licences. Les risques juridiques sont en effet bien réels lors de l'utilisation de l'Open Source et assez peu d'entreprises disposent de l'expertise nécessaire pour les gérer correctement : selon le rapport 2023 State of Open Source d'OpenLogic uniquement 14,16 % des grandes entreprises (plus de 5000 employés) utilisant de l'Open Source ont mis en place un Open Source Office Program. Les industries des Média et du Jeu étant en avance sur le reste du Marché.

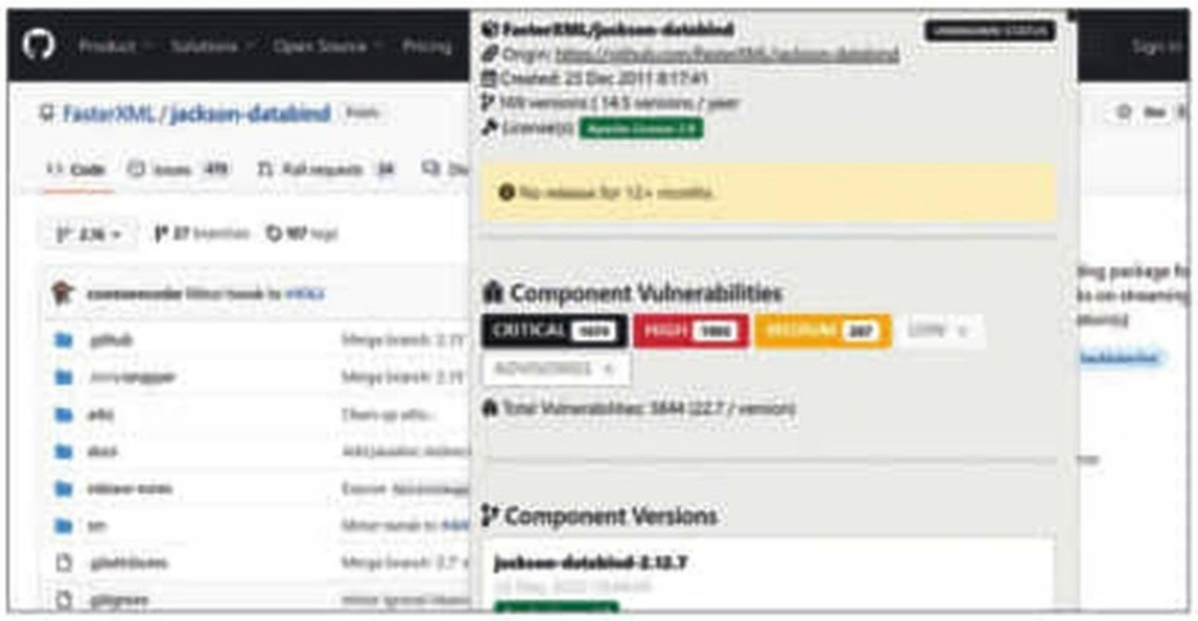


Figure 6

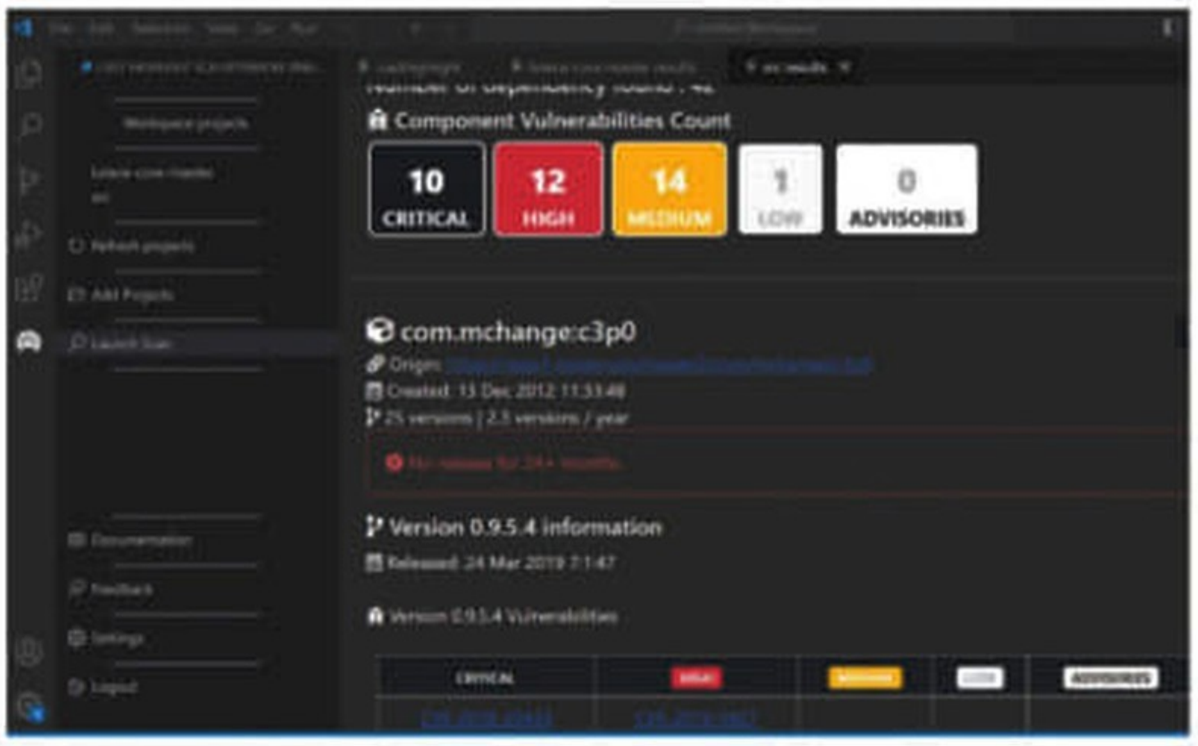


Figure 7

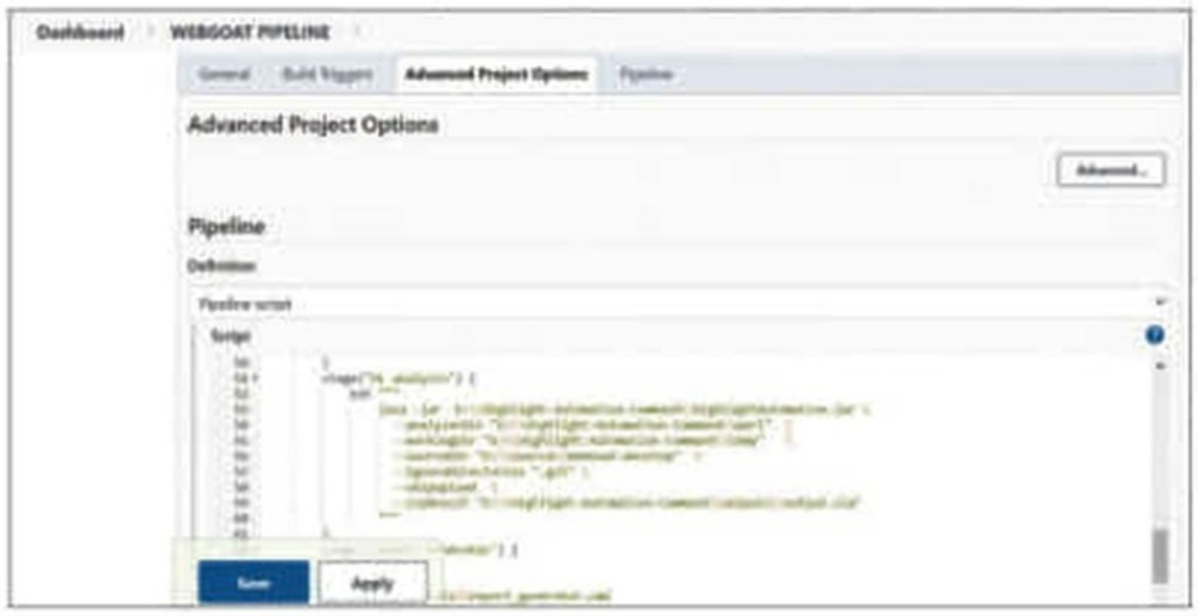


Figure 8

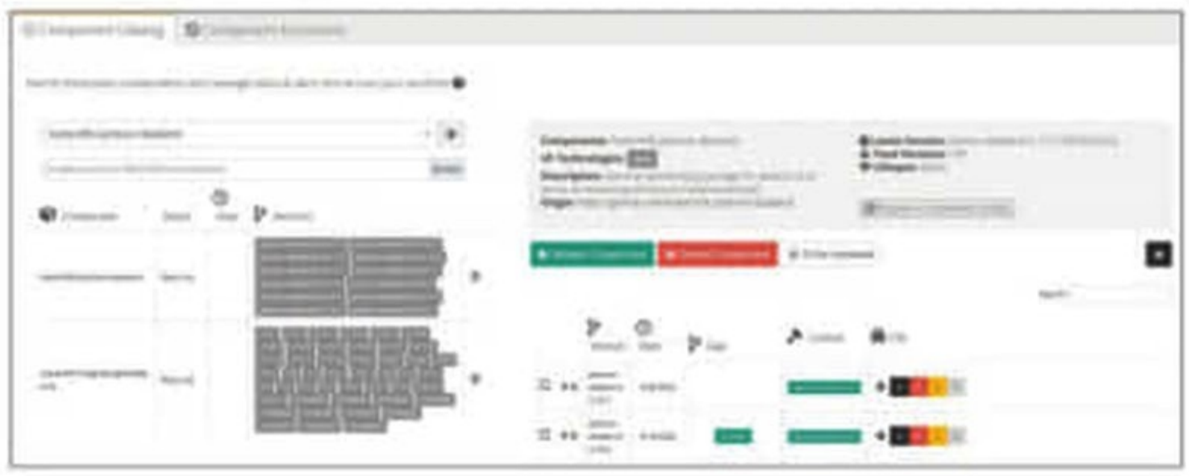


Figure 9

L'Open Source Program Office va utiliser les solutions d'analyse du code Open Source de manière plus avancée en particulier en définissant la politique d'utilisation de ce code sur tous les axes. CAST Highlight a été conçu pour fournir des fonctionnalités supplémentaires destinées à la gestion des politiques Open Source définies par un OSPO. Les composants Open Source pourront être catégorisés comme autorisés ou interdits (Denied list / Allowed list) : **Figure 9**

Dès lors le développeur sera informé au plus tôt qu'il envisa-

ge d'utiliser un composant ou une version de composant interdit. On voit ici que le fond de page du plug-in CAST Highlight pour les navigateurs est devenu jaune indiquant qu'un composant est partiellement interdit d'utilisation au sein de l'entreprise (toutes les versions ne sont pas interdites donc le développeur doit vérifier la politique définie) : **Figure 10**

Si le fond de page était rouge, cela signifierait que le composant, quelle que soit la version, est purement et simplement interdit d'utilisation.

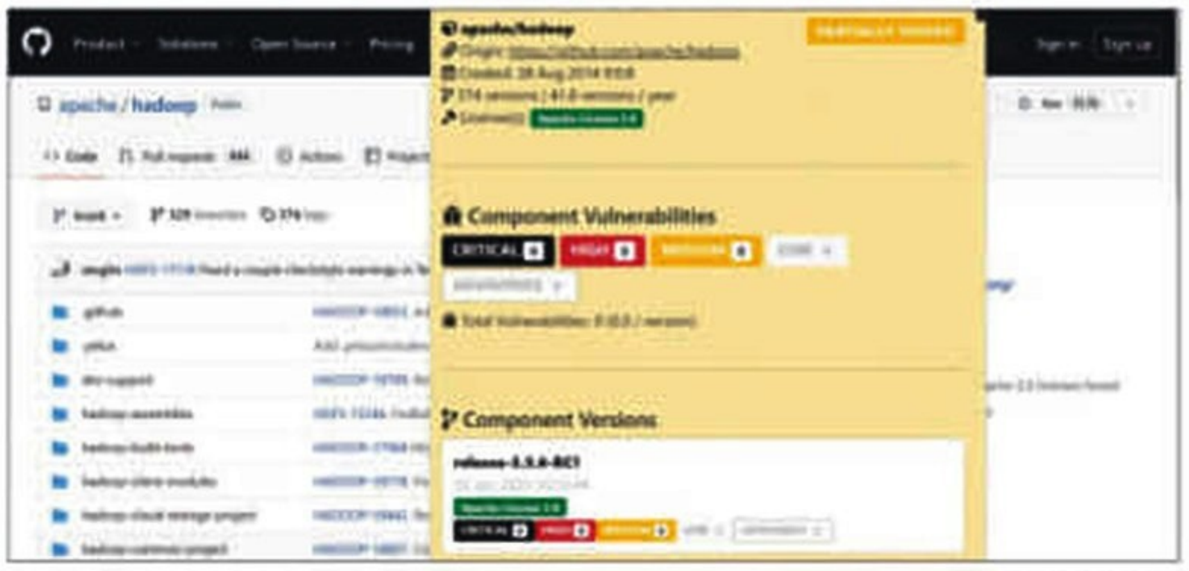


Figure 10

Figure 11

Figure 12

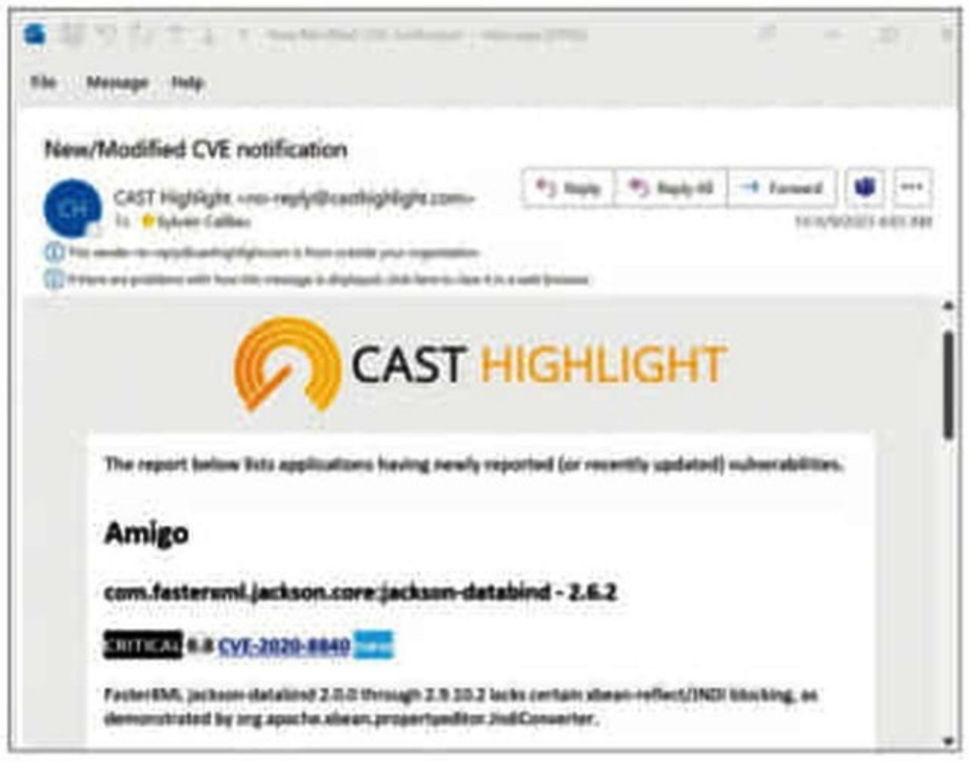


Figure 13

Comment cela se passe-t-il pour l'équipe d'intégration et de déploiement

L'équipe d'intégration peut n'avoir qu'à contrôler le rapport d'analyse de la composition logicielle automatiquement produite par une solution comme CAST Highlight qui aura été intégré au pipeline de déploiement : **Figure 11**

Ainsi les résultats pourront ainsi être consultés avant un déploiement. Si des actions supplémentaires sont réalisées par l'équipe d'intégration ou de déploiement, par exemple préparer un déploiement conteneurisé, l'analyse du code Open Source s'applique à ces tâches.

Si la conteneurisation s'appuie sur le téléchargement de conteneur issu d'une registry Docker publique par exemple, l'équipe d'intégration doit aussi vérifier les risques Open Source embarqués par le ou les conteneurs (CAST Highlight permet de contrôler directement un conteneur).

Comment cela se passe-t-il pour l'équipe d'exploitation

Avoir déployé une application dont tous les composants Open Source embarqués étaient indemnes de vulnérabilité de sécurité au moment du déploiement n'est pas une garantie, malheureusement, que cette situation va perdurer dans le temps. En effet, des failles de sécurité dans des composants Open Source sont découvertes tous les jours et viennent enrichir les bases de vulnérabilités connues.

Ces bases peuvent être publiques comme la National Vulnerability Database gérée par le NIST (Le National Institute of Standards and Technology) ou gérées par les communautés elles-mêmes.

L'équipe d'exploitation va donc devoir assurer le monitoring en continu des applications déployées. La solution CAST Highlight fera partie du dispositif de monitoring. Dès qu'une nouvelle vulnérabilité a été détectée et rendue publique, les applications en production sont potentiellement à risque.

Nous avons vu que les hackers réagissent très vite et tentent d'exploiter toute nouvelle vulnérabilité quelques heures à peine après qu'elle a été rendue publique.

Il faut donc se donner les moyens de réagir et de prendre les bonnes mesures rapidement. Pour se faire, il est crucial d'être informé au plus vite dès qu'une nouvelle vulnérabilité impacte une ou plusieurs de vos applications déployées en production. C'est à cela que CAST Highlight qui dispose d'un mécanisme d'alerte par mail participe au dispositif de monitoring et alerte au plus tôt les responsables d'applications et de l'exploitation.

En effet, dès lors qu'avant d'avoir mis en production une application, vous avez opéré une analyse de composition logicielle avec CAST Highlight et ainsi listé l'ensemble des composants Open Source embarqués par celle-ci (on appelle aussi cela : construire le SBOM – pour Software Bill Of Material – autrement dit la nomenclature logicielle de votre application), CAST Highlight connaît ces composants et dès qu'une nouvelle vulnérabilité est détectée et va alerter par mail de cette occurrence pour permettre ainsi d'agir au plus vite : **Figure 12**

Dès lors qu'une nouvelle vulnérabilité (ici uniquement critique) est connue, un mail comme celui-ci est envoyé : **Figure 13**

Aller plus loin : contrôler les composants Open Source embarqués dans les applications que vous n'avez pas développées

Comme nous l'avons dit en introduction, l'Open Source est partout. Même si votre entreprise s'est organisée pour assurer un haut niveau de contrôle et de gestion de l'Open Source vous ne maîtrisez pas tous les logiciels qui constituent votre système d'information. En effet, lorsque vous installez un progiciel d'éditeur, vous n'avez pas la maîtrise de ce qui est embarqué dans ce progiciel.

De même que la vulnérabilité log4shell avait défrayé la chronique, l'attaque de Solarwind en 2020 avait aussi fait les gros titres de la presse. Voici un article pour vous remémorer cet événement : The SolarWinds Cyber-Attack: What You Need to Know (cisecurity.org). Cette attaque, très sophistiquée, a rendu vulnérable l'ensemble des clients du progiciel et c'est la chaîne de fabrication chez l'éditeur qui a été compromise empêchant les contrôles classiques d'intégrité.

Pour limiter les possibilités d'introduire ou de réintroduire des vulnérabilités au sein de votre système d'information, il est aussi important que vous puissiez vous assurer que les progiciels que vous installez n'embarquent pas de composants porteurs de vulnérabilité. Il existe une méthode pour obtenir cette assurance : exiger de votre fournisseur qu'il vous communique la nomenclature (SBOM) du progiciel qu'il vous livre. Cette nomenclature doit vous être fournie sous forme d'un fichier dans un format standard que vous pourrez facilement exploiter. Aujourd'hui le format standard le plus répandu, pour fournir une nomenclature, est le format CycloneDX. La fourniture d'un SBOM devient par ailleurs une obligation aux États-Unis conformément à l'EXECUTIVE ORDER 14028, IMPROVING THE NATION'S CYBERSECURITY(8). Si votre entreprise est éditeur de logiciel ou fournisseur de code pour d'autres entités que ce soit au sein d'un même groupe ou comme contributeur à des projets collaboratifs, la solution CAST Highlight vous permet de fournir à des tiers la nomenclature de votre logiciel au format CycloneDX entre autres : **Figure 14**

Figure 14

Figure 15

De même la solution CAST Highlight va vous permettre d'exploiter une nomenclature qui vous aura été fournie au format CycloneDX. En effet, même si votre fournisseur vous livre un fichier au format CycloneDX, il vous sera difficile de l'exploiter manuellement. En revanche, en utilisant une solution comme CAST Highlight vous pourrez immédiatement obtenir un rapport de vulnérabilité pour les composants listés dans la nomenclature. Pour ce faire, vous devez créer un conteneur dans Highlight de la même manière que pour toute application que vous souhaitez analyser. Cependant vous n'analyserez pas le code de l'application (qui de toute manière ne vous a pas été livré). En lieu et place, vous importerez le contenu du fichier CycloneDX livré et la solution consultera les bases de références pour vous indiquer immédiatement si des composants embarqués sont vulnérables. La capture d'écran ci-dessous montre l'écran d'import d'un SBOM dans CAST Highlight : **Figure 15**

Une fois le fichier CycloneDX importé, CAST Highlight affiche les vulnérabilités (ici, dans la capture d'écran ci-dessous, 2 vulnérabilités « high » selon la base NVD et 3 vulnérabilités connues de GITLAB non encore répertoriées dans la base NVD) – vous noterez l'icône « component imported » qui vous indique que le composant n'a pas été détecté par Highlight, mais provient de l'import d'un fichier Cyclone DX. Une fois cette nomenclature importée, vous disposez des fonctionnalités d'alerte en cas de détection de nouvelles vulnérabilités. Vous pourrez alors interpeller votre fournisseur pour qu'il mette à jour son progiciel et vous fournisse une nouvelle version « nettoyée » : **Figure 16**

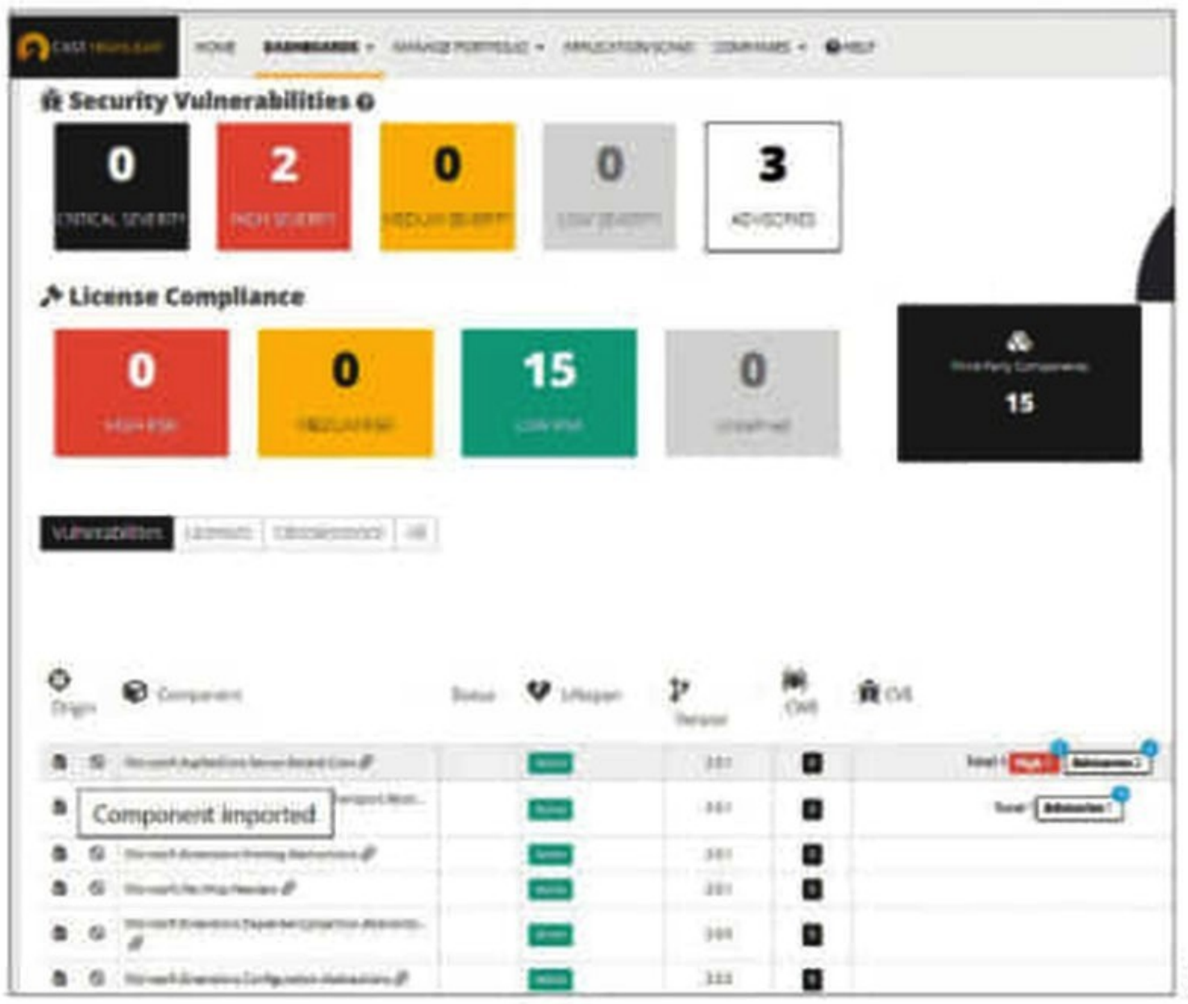


Figure 16

Pour conclure : la sécurité Open Source est une discipline à part entière, j'espère vous en avoir convaincu. Elle nécessite la mise en œuvre d'actions spécifiques tout au long du cycle de vie des applications et exige un outillage approprié comme l'est CAST Highlight. C'est un élément indispensable de votre Cyber-Défense.

(8) Software Security in Supply Chains: Software Bill of Materials (SBOM) | NIST

Des outils Open Source pour la sécurité

À l'instar des principes S.O.L.I.D. pour la programmation orientée objet, je me demandais s'il existait des hypothèses ou fondements en matière de sécurité logicielle. Le projet « OWASP Top 10 Companion Workbook » d'Olivia Dell (www.olivialiddell.com/projects) m'a aidée à mieux comprendre et assimiler le Top 10. Ce nouvel article fait partie intégrante de mon cheminement dans la découverte de la sécurité du cycle de développement.

Romy Alula

Romy est ingénieure logiciel chez Filigran (filigran.io). Elle a rejoint cette startup pour son coeur de métier : traduire la connaissance de l'intelligence de la menace en actions. Les questions qu'elle se posait en matière de cybersécurité ont fait grandir son intérêt pour ce vaste domaine. Guidée par sa passion et sa curiosité, elle partage ses apprentissages et découvertes avec d'autres curieuses et curieux au travers de coding dojos, meetups et autres conférences. Dès qu'elle a du temps libre, elle joue du piano, prend des cours de danse et encourage ses enfants à exprimer leur créativité. @Goumies

À l'époque où je cherchais à lister et expliciter ces principes, je me retrouvais finalement dans des démonstrations d'outils propriétaire. L'open source est plein de projets et d'outils fiables. Année après année, sa robustesse permet de faire face à l'apparition de menaces et l'industrialisation de celles qui sont connues.

La détection de failles de sécurité et d'intrusion sur le réseau

Network Intrusion Detection and Prevention System

Snort - 1998 - https://www.snort.org

Snort est un système de prévention des intrusions. Il a la particularité d'avoir été récompensé par InfoWorld, dans son Open Source Hall of Fame, en 2009, dans la catégorie « meilleur logiciel open source de tous les temps » (source : https://www.infoworld.com/article/2631146/the-greatest-open-source-software-of-all-time.html). **Figure 1**

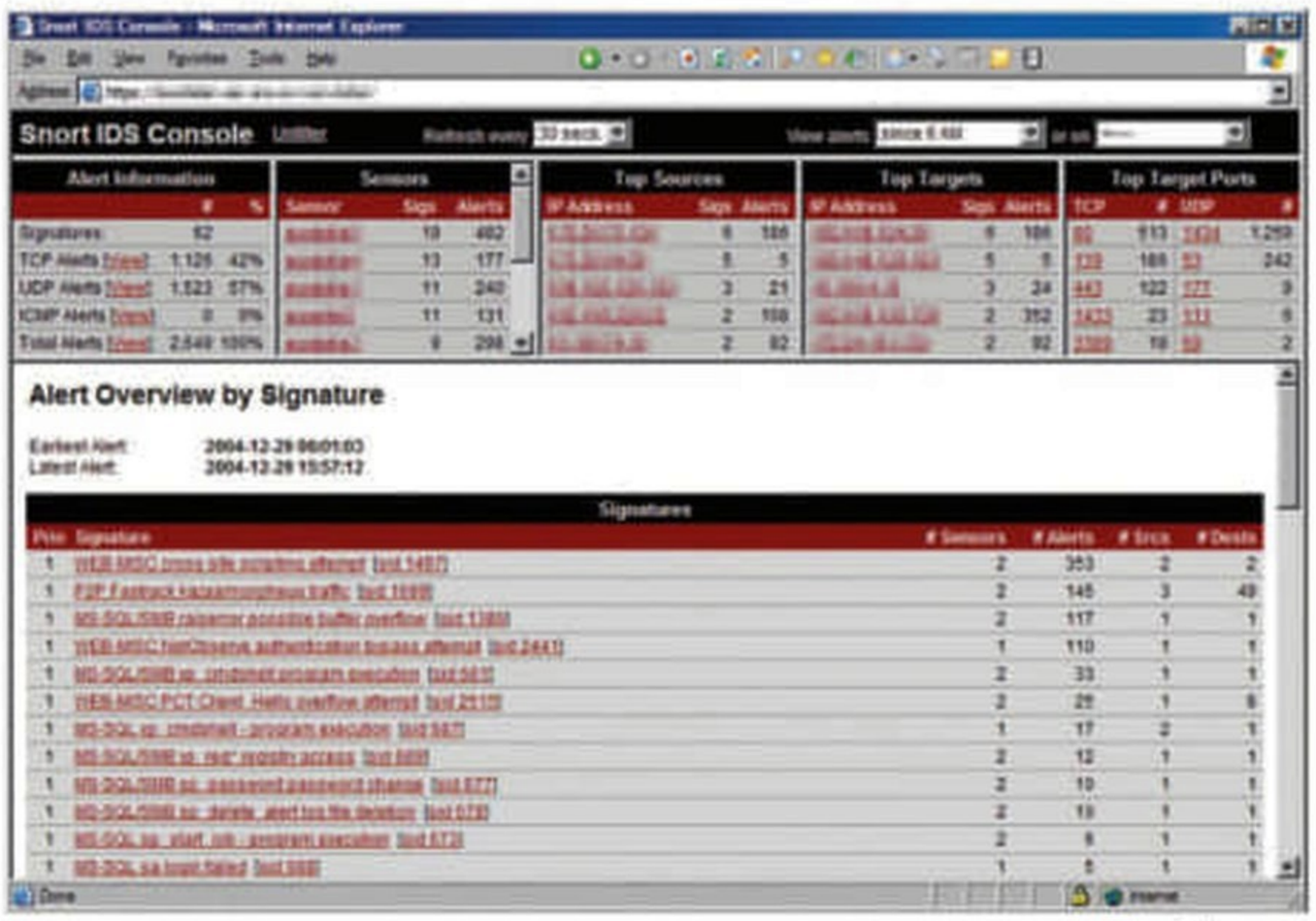


Figure 1

Network Analysis and Threat Detection

Suricata - 2010 - https://suricata.io/

En plus de détecter les intrusions, Suricata a la capacité de les prévenir. **Figure 2**

Host-based Intrusion Detection System

Ce système de détection d'intrusion est capable de monitorer et analyser une machine, mais aussi les paquets circulant sur son réseau.

OSSEC (Open Source HIDS SECurity)

release 2.7 - 24 février 2014 - https://www.ossec.net/about

OSSEC évalue les logs, vérifie l'intégrité des fichiers, surveille le monitoring. **Figure 3**

La gestion de la sécurité du développement des logiciels

Détection d'informations sensibles

Gitleaks - 1.0.0 - 21 juillet 2018 - https://gitleaks.io

Les repos Git sont parcourus pour y déceler toute information sensible comme des mots de passe, des clés d'API ou autres données confidentielles. Gitleaks peut être intégré à la CI. **Figure 4**

Clean Code

SonarQube - 2009 - https://www.sonarsource.com/company/about

Figure 3 - *Source : https://www.linux-magazine.com/Issues/2009/103/Security-Lessons/%28offset%29/3*

Figure 2

SonarQube est une plateforme pour l'inspection continue de la qualité du code. Elle est dotée de plugins permettant d'identifier des vulnérabilités, des code smells ou autres problèmes liés à la qualité. **Figure 5**

Le lien entre la qualité et la sécurité du code réside dans son impact sur la facilité de correction, maintenabilité et évolution du projet. En cas d'incident ou attaque, le temps de réaction pour livrer en production est capital. Encore faut-il pouvoir cibler la ou les zones vulnérable(s).

La surveillance et la gestion des vulnérabilités (scan, patch management)

Vulnerability Scanner

OpenVAS - release 3.0.1 - 26 janvier 2010 - https://openvas.org

L'outil peut scanner un équipement, un ensemble d'équipements ou l'intégralité d'un réseau. **Figure 6**

SCA (Software Composition Analysis)

Ce type d'analyse couvre les dépendances externes à l'application. Elle permet de détecter si elles sont à jour, contient des défauts de sécurité ou des contraintes de licence.

OWASP Dependency-check - 2012

https://owasp.org/www-project-dependency-check

Dependency-check identifie les vulnérabilités connues parmi les dépendances de la solution. Plusieurs langages de programmation sont supportés. Cet outil peut être intégré au processus de build.

OWASP Dependency-Track - 2013

https://owasp.org/www-project-dependency-track

Cet autre outil de SCA peut générer des rapports, suivre les efforts de remédiation et aider à gérer la chaîne d'approvisionnement logiciel. **Figure 9**

Détection des librairies obsolètes

Retire.js - release 1.0.0 - 1er avril 2015

https://github.com/RetireJS/retire.js

Ce scanner de librairies JavaScript aide à repérer les librairies obsolètes et vulnérables. Il existe une interface en lignes de commandes pour une intégration au build. Il peut également se brancher à Grunt ou ZAP. Des extensions sont disponibles pour Chrome et Firefox.

L'évaluation du code des applications

DAST (Dynamic Application Security Testing)

Il s'agit de l'exécution de tests d'intrusion automatisés, dans un environnement isolé. Au lancement, l'outil scanne des réponses à des requêtes avec divers payloads pour trouver des défauts de sécurité.

OWASP ZAP (Zed Attack Proxy) - 6 septembre 2010

https://www.zaproxy.org **Figure 7**

Zap est un scanner d'applications Web très populaire.

Je peux lister quelques fonctionnalités :

- Passive scanning, toutes les requêtes et réponses sont analysées pour détecter les vulnérabilités
- Active scanning, toutes les requêtes sont rejouées pour identifier les plus grandes vulnérabilités
- Fuzzing, les requêtes sont aléatoirement modifiées **Figure 8**

Il existe aussi une CLI pour brancher ZAP à votre usine logicielle (https://www.zaproxy.org/docs/desktop/cmdline).

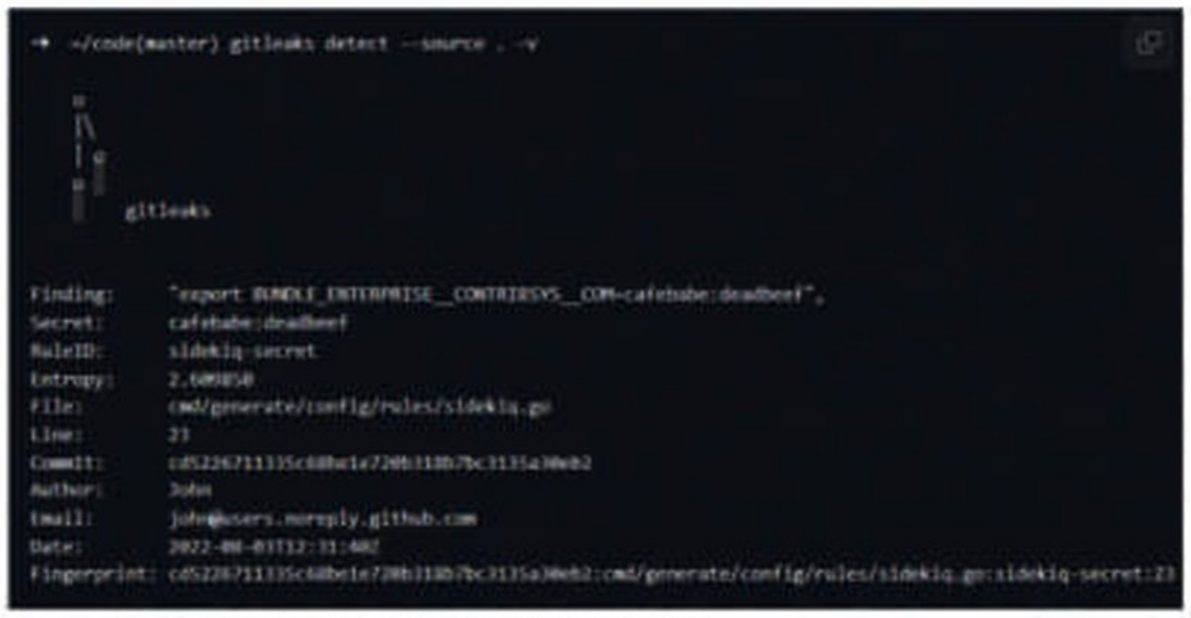


Figure 4 - *Source : https://github.com/gitleaks/gitleaks*

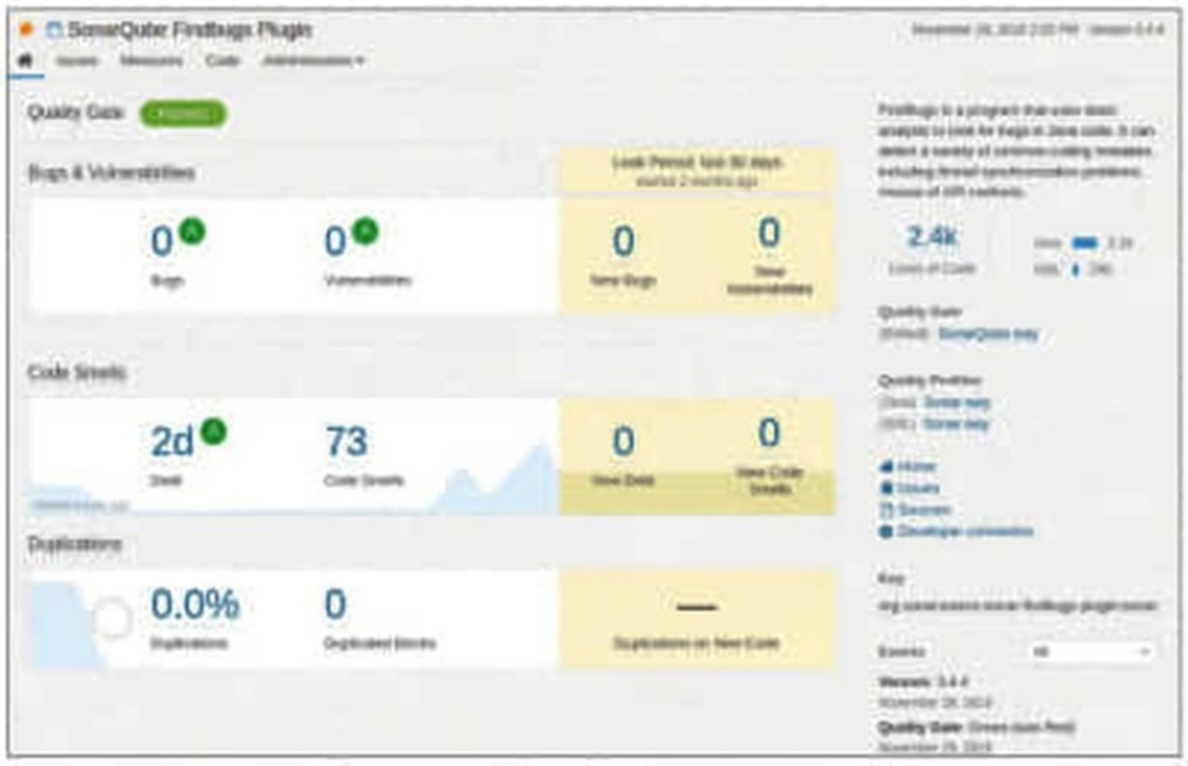


Figure 5 - *Source : Ganncamp — Travail personnel, CC BY-SA 4.0,*

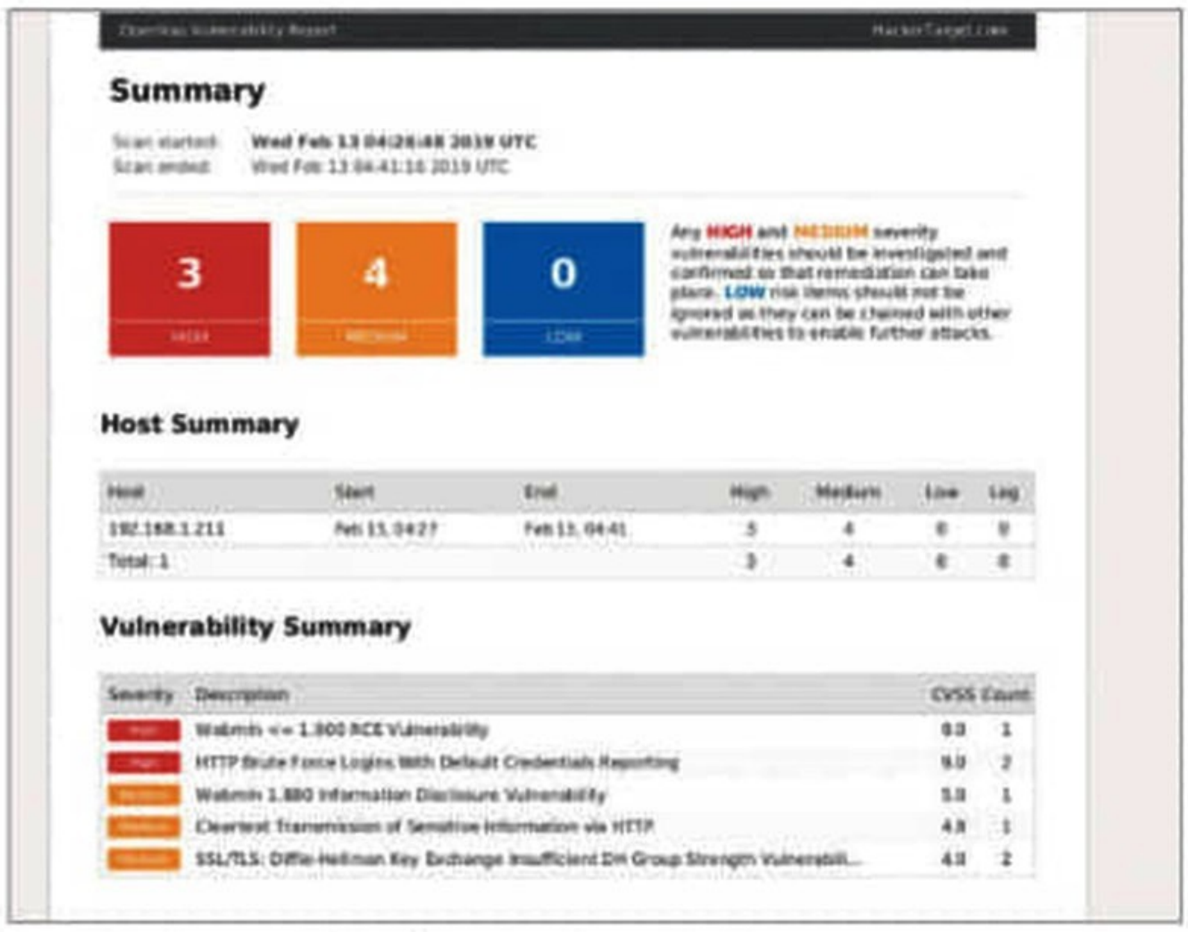


Figure 6 - *Source : https://hackertarget.com/openvas-scan*

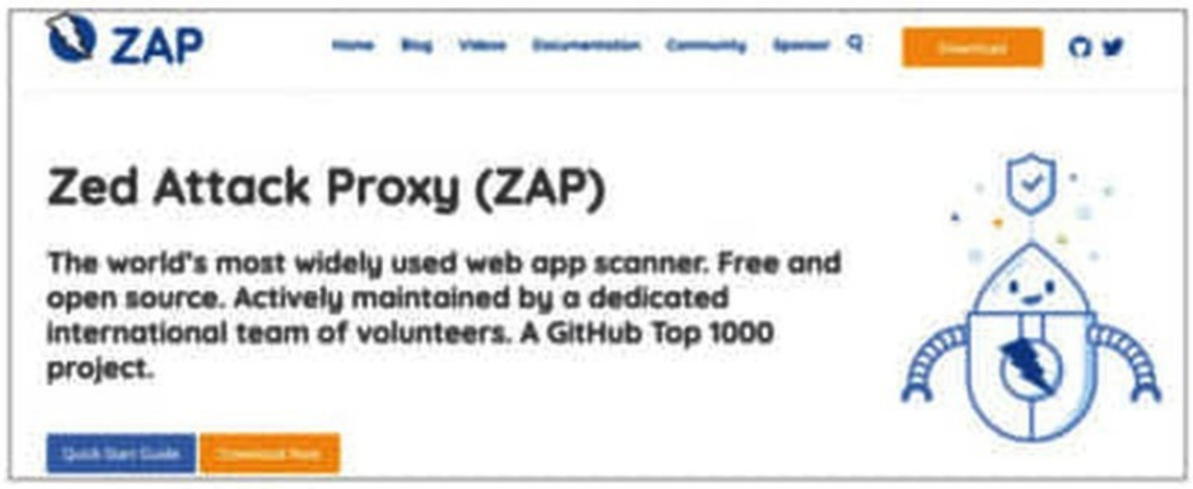


Figure 7 - *Source : https://github.com/zaproxy/zaproxy*

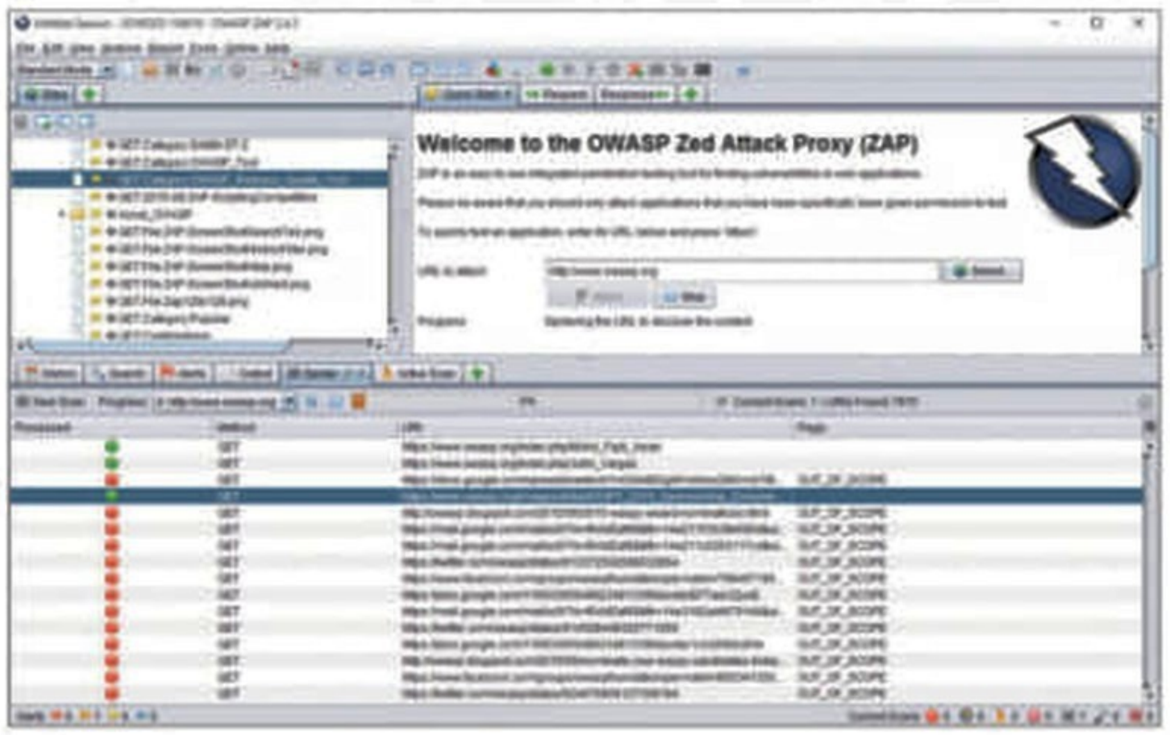


Figure 8 - *Source : Software Informer 2018*

Figure 9 - *Source : https://brakemanscanner.org/docs/quickstart*

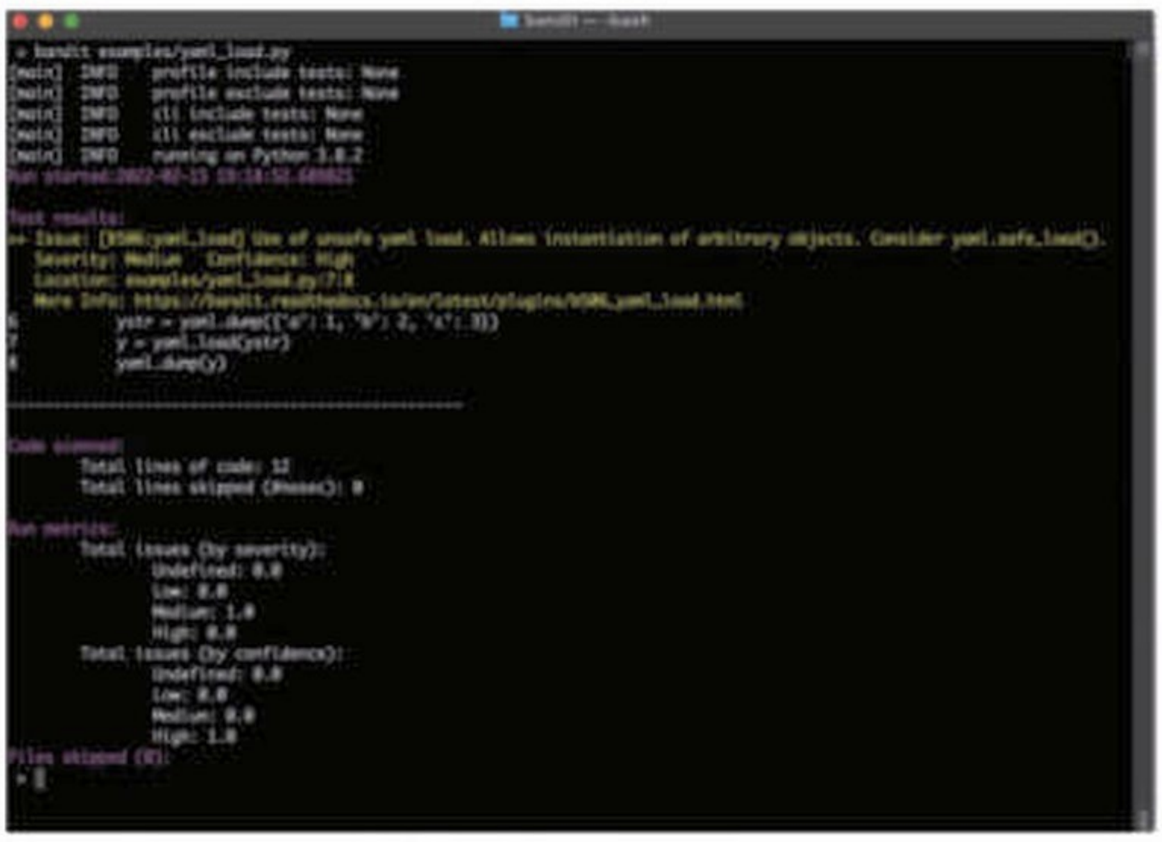


Figure 10 - *Source : https://github.com/PyCQA/bandit*

Figure 11 - *Source : https://github.com/gauntlt/gauntlt/blob/master/examples/generic/generic.attack*

SAST (Static Application Security Testing)

Ce type de test procède à l'analyse statique du code source pour y détecter des patterns de vulnérabilité.

Brakeman - 8 décembre 2011 - https://brakemanscanner.org

Brakeman est spécialement conçu pour les applications Ruby on Rails. Il évalue le code source et alerte sur les vulnérabilités potentielles comme le cross-site scripting (XSS) ou les injections SQL. **Figure 9**

Bandit - janvier 2015

https://github.com/PyCQA/bandit

Bandit est dédié aux applications Python. **Figure 10**

Vous pourrez trouver d'autres outils de SAST : https://owasp.org/www-community/Source_Code_Analysis_Tools

La Cyber Résilience

Cette approche assure :

- Fiabilité
- Intégrité
- Authentification
- Confidentialité et disponibilité des données

Une bonne pratique comme la distinction entre les environnements de développement, de test et de production contribue à la Cyber Résilience.

Ruggedization

Gauntlt - release 1.0.5 - 7 août 2013

https://github.com/gauntlt/gauntlt

« Soyez méchant avec le code », c'est le slogan de Gauntlt. Ce framework de tests de sécurité les automatise dans le cycle de développement. Il s'intègre avec de nombreux outils et frameworks. Facilitant les tests end-to-end, à l'aide de fichiers d'attaque rédiger en Gerkhin. **Figure 11**

Pour en savoir plus : https://insights.sei.cmu.edu/blog/devops-technologies-gauntlt

Formation

La formation au développement sécurisé améliore la résilience des équipes.

OWASP Security Shepherd - release 2.3 - 19 mai 2015

https://github.com/OWASP/SecurityShepherd

L'OWASP propose une plateforme d'entrainement à la sécurité des applications Web et mobile. Elle fournit des applications volontairement vulnérables pour aider les dev à s'exercer dans l'identification et la réparation des failles de sécurité.

Conclusion

L'ensemble des outils présentés sont actifs depuis plusieurs années. Le plus ancien date de 1998. Toujours pertinent, il est la preuve que l'open source apporte des solutions fiables et robustes. Solutions capables de mitiger les attaques toujours plus nombreuses, automatisées et industrialisées. La transparence, inhérente à la nature de l'open source permet aux experts d'appréhender et de scruter le code à l'affût de problèmes ou autres « backdoors ». Cette transparence est un gage de confiance pour les utilisateurs. Elle prévient les risques de compromission. La sécurité est un défi permanent. En tant que dev, nous pouvons construire des solutions avec ces quelques pistes en tête. Pour avancer dans ma quête de principes, j'ai trouvé, entre autres, le guide « Bonnes pratiques de cybersécurité pour le développement logiciel » (https://hal.science/hal-02075159/document).

Sécuriser votre annuaire AD et votre Tenant Microsoft 365 avec Harden AD et Harden 365

Guillaume MATHIEU

Membre fondateur Harden

HARDEN

Secure in minutes

Les annuaires Active Directory sont une cible privilégiée par les attaquants, car compromettre l'Active Directory, c'est prendre le contrôle de toutes les machines membres du domaine. Quand une machine est jointe dans un domaine Active Directory, le groupe Domain Admins est ajouté en tant que membre du groupe Administrators de la base SAM de la machine. Si un attaquant peut s'ajouter dans ce groupe, il devient administrateur local de l'ensemble des machines membres du domaine.

Repartir d'un nouveau domaine Active Directory ne permet pas non plus de bénéficier d'un annuaire sécurisé. Des protocoles à risque comme *LLMNR*, *LDAP BIND SIMPLE*, *NETBT-NS*, *SMB V1* restent activés par défaut alors qu'ils sont activement exploités par des outils d'attaques comme *CRACKMAPEXEC*, *MIMIKATZ*, *RUBEUS* ou *IMPACKET*. Avec un domaine AD en configuration par défaut, les membres du groupe *Domain Admins* peuvent ouvrir des sessions sur toutes les machines du domaine, car aucun modèle de Tiering AD n'est mis en place. En prenant le contrôle d'une seule machine, un attaquant pourrait escalader en tant *Domain Admins*.

Les services Microsoft 365 comme *Exchange Online*, *OneDrive*, *SharePoint Online* et *Teams* sont aussi une cible privilégiée par les attaquants, car ils sont accessibles depuis Internet. Une campagne de phishing est assez simple à mettre en place avec des outils comme *Evilginx* et permet généralement de récupérer un accès distant sur la machine de la victime.

La configuration par défaut de l'antispam *Exchange Online Protection* est trop permissive par défaut pour se protéger contre le phishing, les spams et les logiciels malveillants. La configuration par défaut ne limite pas par exemple le nombre de courriels envoyés par jour par un utilisateur standard. Les protocoles *DKIM* et *DMARC* ne sont pas non plus activés pour les domaines de messagerie de l'entreprise.

Microsoft propose bien des *Preset Security Policies*, mais ces dernières ne sont pas activées par défaut et ne sont pas paramétrables.

https://learn.microsoft.com/en-us/microsoft-365/security/office-365-security/preset-security-policies?view=o365-worldwide

Microsoft propose aussi une configuration sécurisée par défaut pour son annuaire Azure AD via la fonctionnalité *Security Defaults*. Cette dernière permet de désactiver les protocoles à risque comme l'*authentification basique*, *Active Sync* et impose l'utilisation d'un second facteur (MFA) pour tous les comptes Azure AD. La fonctionnalité *Security Defaults* n'est malheureusement pas paramétrable et est donc souvent désactivée par les équipes IT.

https://learn.microsoft.com/en-us/azure/active-directory/fundamentals/security-defaults

Gagner du temps avec Harden AD et Harden 365

Les équipes IT peuvent s'appuyer sur les guides *CIS Benchmark* (https://www.cisecurity.org/cis-benchmarks), la documentation *Microsoft Learn* (https://learn.microsoft.com) ou les guides de l'ANSSI (https://www.ssi.gouv.fr/entreprise/bonnes-pratiques)

Cela nécessite cependant beaucoup de temps pour assimiler plusieurs dizaines de milliers de pages de bonnes pratiques et créer manuellement chaque règle de sécurité requise.

Sécuriser correctement un annuaire AD ou un Tenant Microsoft 365 peut donc nécessiter plusieurs semaines, voire plusieurs mois de travail.

Les membres de la communauté Harden ont analysé ces documents et définis des configurations sécurisées par défaut pour les annuaires AD et les tenants Microsoft en s'appuyant sur les retours d'expériences des différents projets de sécurité AD et Microsoft 365.

D'un point de vue technique, ces configurations standardisées ont été traduites en scripts PowerShell.

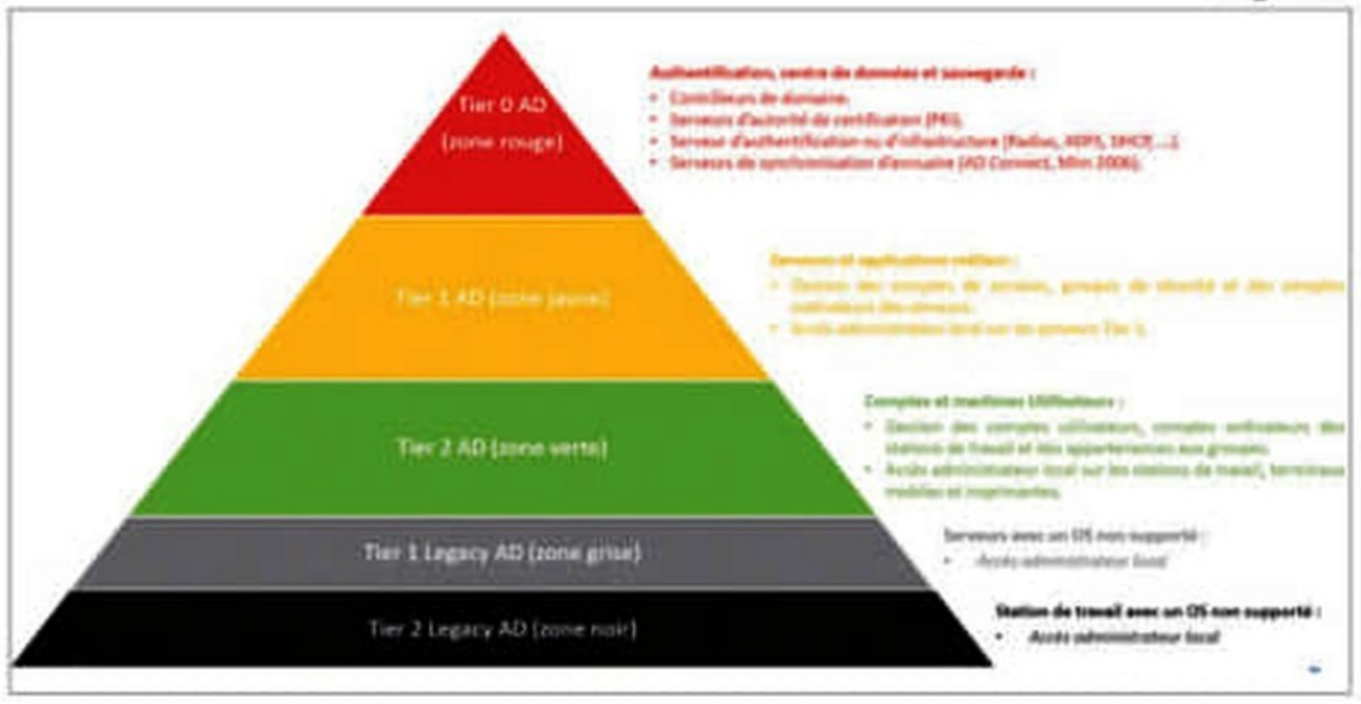
Harden AD et Harden 365 sont livrés avec des fichiers de configuration afin de pouvoir personnaliser la configuration cible déployée sans modifier le code des scripts PowerShell.

Présentation et déploiement de Harden AD

Harden AD segmente l'annuaire AD en 5 Tiers. **Figure 1**

Pour cela, Harden AD crée une unité d'organisation, une

Figure 1



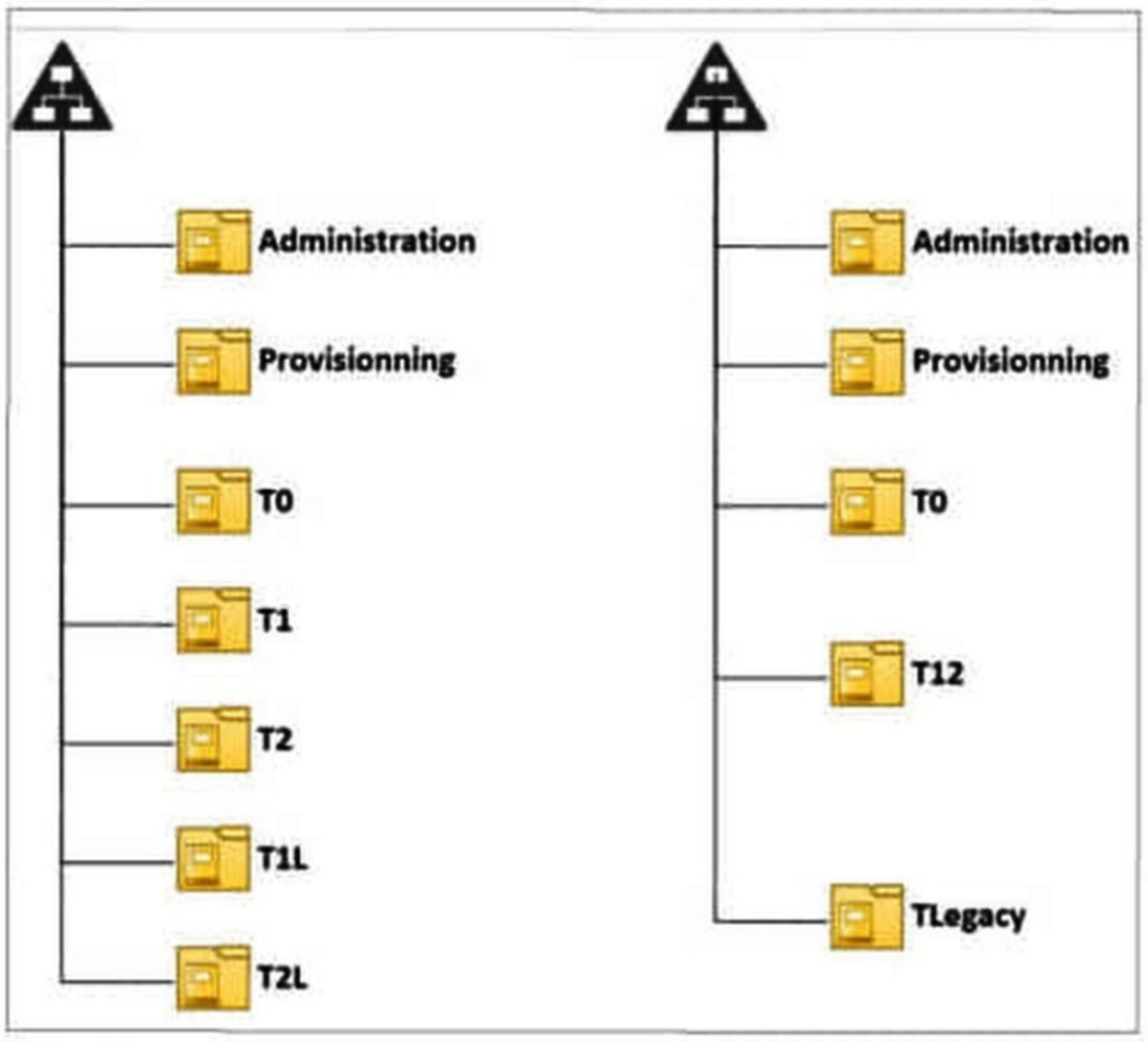


Figure 2

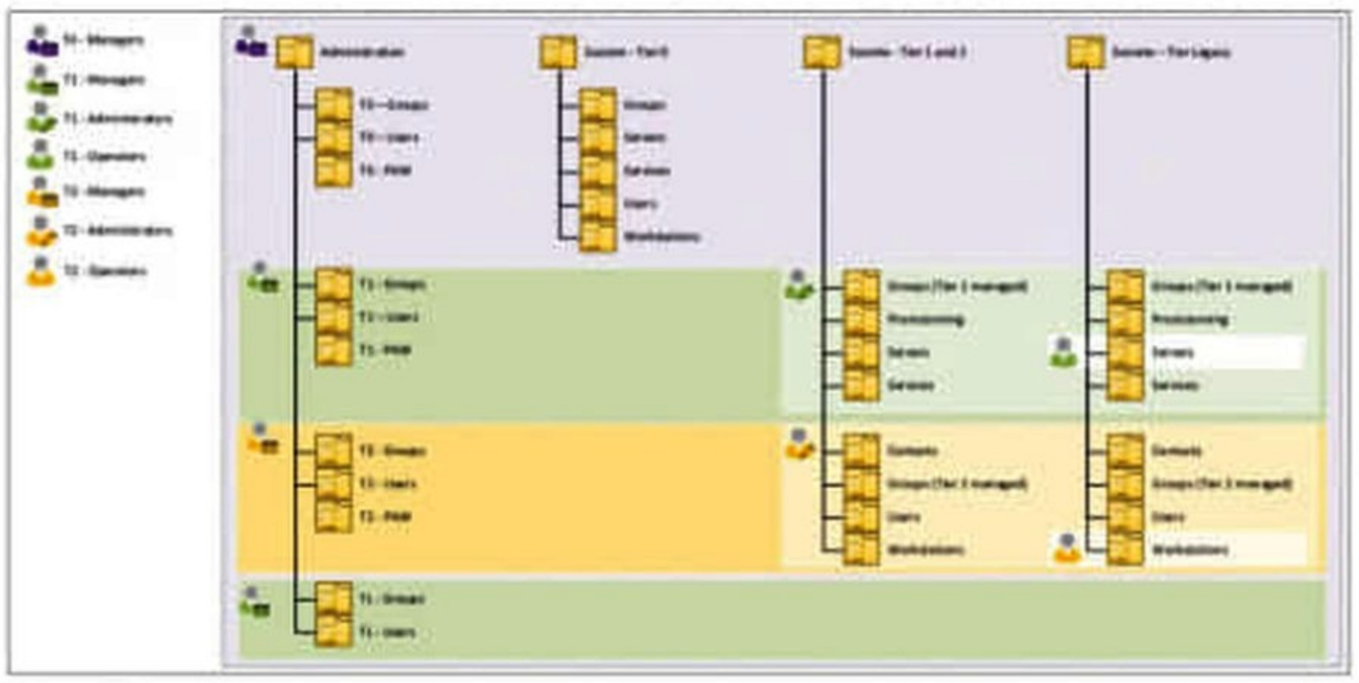


Figure 3

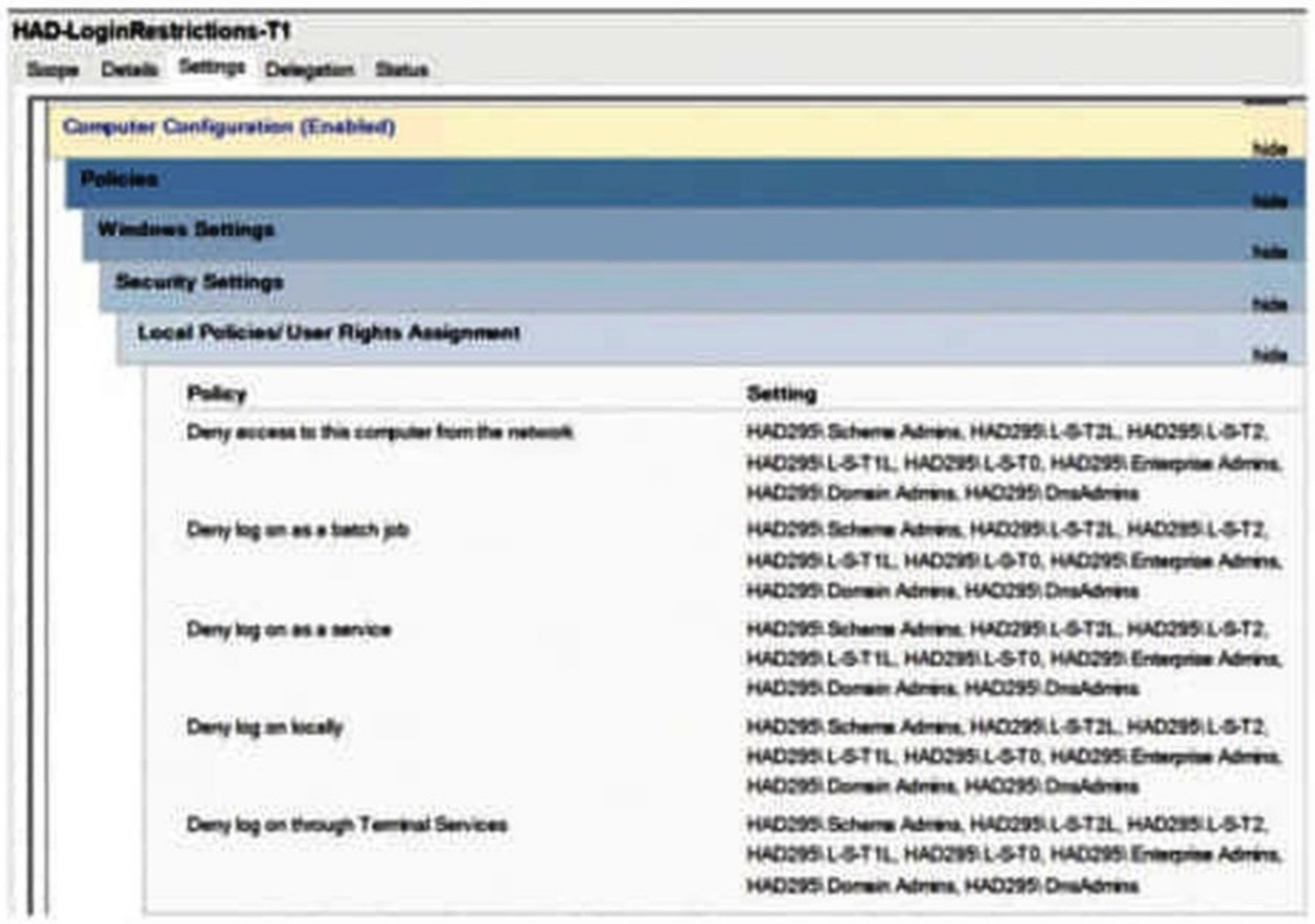


Figure 4

GPO, un compte utilisateur d'administration et un groupe pour chaque Tier AD. 2 variantes sont possibles, une avec 5 unités d'organisation racines, l'autre avec 7 unités d'organisation. Dans la variante à 5 unités d'organisation, les ressources du Tier 1 et du Tier 2 sont rassemblées sous l'unités d'organisation *T12*. Même principe pour les ressources Tier 1 Legacy et Tier 2 Legacy rassemblées sous l'unité d'organisation *TLegacy*. **Figures 2 et 3**

Chaque GPO va interdire les ouvertures de session locale, bureau à distance, service, batch et réseau aux membres des groupes des autres Tier AD en activant les paramètres suivants :

- Interdire l'ouverture d'une session locale (*SeDenyInteractiveLogonRight*)
- Interdire l'ouverture de session par les services Bureau à distance (*SeRemoteInteractiveLogonRight*)
- Interdire l'accès à cet ordinateur à partir du réseau (*SeDenyNetworkLogonRight*)
- Interdire l'ouverture de session en tant que service (*SeDenyServiceLogonRight*)
- Interdire l'ouverture de session en tant que tâche (*SeDenyBatchLogonRight*).

Ce mode de fonctionnement permet aussi de bloquer pour l'utilisation des groupes AD à fort privilèges comme *Domain Admins* pour tous les Tiers AD sauf le Tier 0 AD. **Figure 4**

Un membre de l'équipe IT dispose alors d'un compte utilisateur d'administration pour chaque Tier AD qu'il gère. S'il essaie d'utiliser un compte d'administration du Tier X pour ouvrir une session sur une machine du Tier Y, Windows bloquera l'ouverture de session.

Les nouvelles unités d'organisation vont aussi permettre la mise en place d'une politique de délégation d'administration AD fine pour chaque groupe de Tier AD.

Harden AD va permettre aussi de définir via une GPO de préférence qui est administrateur local d'une machine membre du domaine. Harden AD crée automatiquement par un script un groupe pour chaque machine membre du domaine et une première règle permet d'ajouter ce groupe en tant que membre du groupe Administrators de la base SAM sur les machines membres du domaine.

Harden AD crée aussi un groupe pour chaque Tier et ajoute ce groupe en tant que membre du groupe Administrators de la base SAM. **Figure 5**

Harden AD crée aussi environ 94 GPO et 41 filtres WMI pour déployer les bonnes pratiques de sécurité (désactivation du protocole SSL2, SSL 3.0,...). Par défaut, l'application de ces GPO est filtré par un groupe. Il est donc possible de déployer progressivement ces nouveaux paramètres de sécurité. L'autorisation par défaut *Utilisateurs authentifiés* (*Authenticated Users*) a été retirée au profit d'un groupe dédié pour chaque GPO.

Harden AD déploie aussi *Laps Legacy* (et non *Windows Laps*), car cette version permet de prendre des machines avec des OS non supportés comme Windows 2008 R2 ou Windows 7.

Pour exécuter Harden AD par défaut, il suffit d'exécuter le script PowerShell *HardenAD.ps1*.

QUI SE CACHE DERRIÈRE LES PROJETS HARDEN AD ET HARDEN 365 ?

À l'origine de la communauté Harden, et donc des outils Harden AD et Harden 365, nous retrouvons trois consultants : Guillaume MATHIEU, Jean-François FERNANDEZ et Loïc VEIRMAN.

L'association Harden dispose d'une trentaine de contributeurs actifs et est parrainée par 4 sociétés : INEXSYA, MSSEC, QWANT et SYNETIS.

La communauté Harden recherche des contributeurs pour :

- Proposer de nouvelles mesures de sécurité pour bloquer les outils d'attaque AD et Microsoft 365
- Réaliser des tests d'offensifs sur des annuaires AD et des Tenant Microsoft durcis avec Harden AD et Harden 365

Si cela vous intéresse, n'hésitez pas à contacter l'association via cette adresse de courriel contact@hardenad.net.

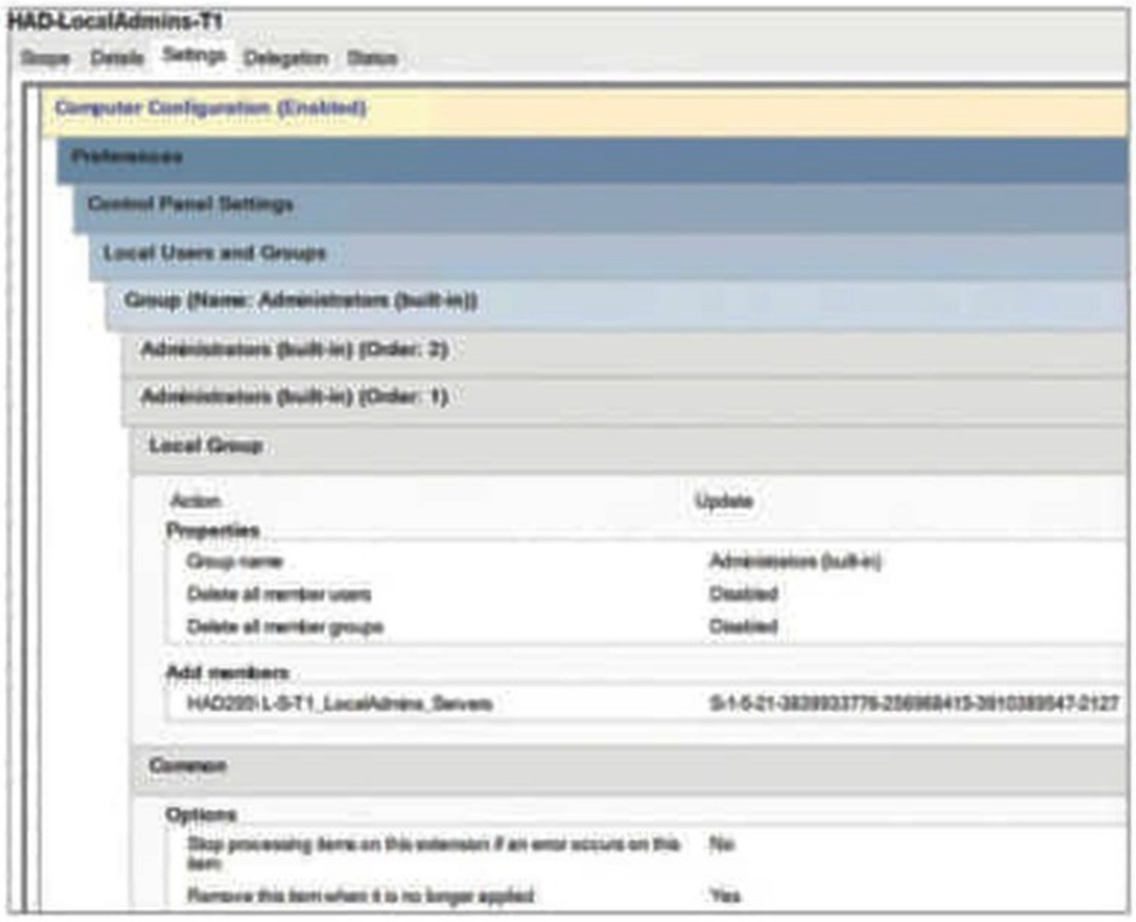


Figure 5

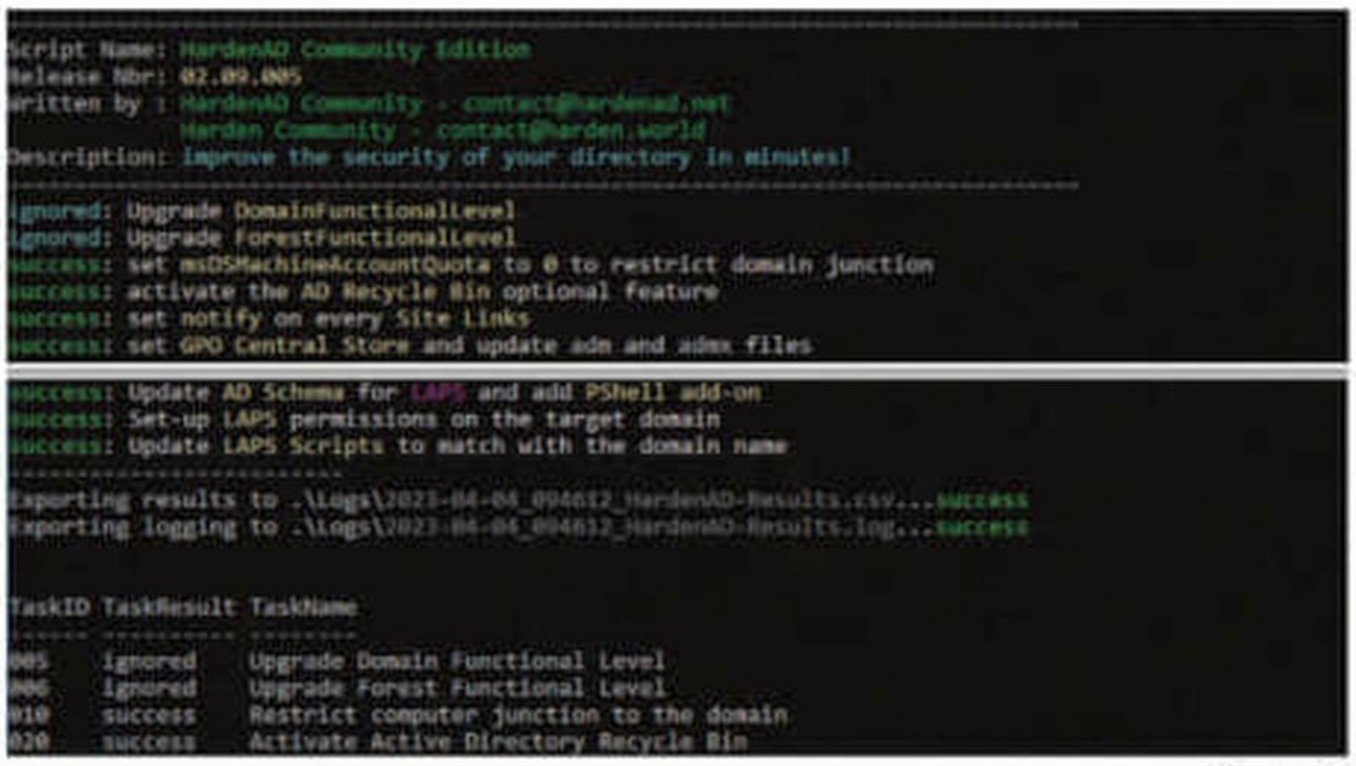


Figure 6

Si vous souhaitez personnaliser votre configuration, vous pouvez modifier le fichier *TasksSequence_HardenAD.xml* qui se trouve dans le dossier *Configs*. **Figure 6**

Présentation et déploiement de Harden 365

Harden 365 permet de faire un diagnostic de sécurité d'un Tenant Microsoft 365 :

- Utilisation d'ORCA pour analyser la configuration *Exchange Online Protection*.
- Revue de la configuration des comptes Azure AD avec privilèges. Une alerte est remontée si le compte est synchronisé ou qu'il n'utilise pas le domaine DNS par défaut du Tenant *(@XXX.onmicrosoft.com)*.
- Lister les boîtes aux lettres configurées pour faire du transfert automatique vers une boîte aux lettres externe (risque de sécurité).
- Lister les permissions définies sur chaque boîte aux lettres. Elles sont très rarement mises à jour lors des mobilités d'utilisateurs entre services par exemple.
- Vérifier que les protocoles DKIM et DMARC sont configurés correctement.

Harden 365 permet d'améliorer le niveau de sécurité du Tenant Microsoft 365 en :

- Créant 2 comptes brise-glace.
- Activant l'authentification à 2 facteurs par SMS en masse (fichier CSV en entrée). Cette action est particulièrement intéressante en cas d'incident de sécurité si l'authentification MFA n'était pas déployée précédemment.
- Créant des règles d'accès conditionnelles pour bloquer les protocoles ActiveSync et l'authentification basique, restreindre les accès des comptes invités. Par défaut l'ensemble des règles créées sont désactivées.
- Déployant toutes les mesures de sécurité recommandées par *Orca* en créant automatiquement des groupes d'exceptions.
- Paramétrant le protocole *DKIM* et *DMARC*.
- Configurant les accès invités pour *SharePoint Online* et *Teams*.
- Créer les profils de configuration Microsoft Intune pour sécuriser les machines jointes à Azure AD.

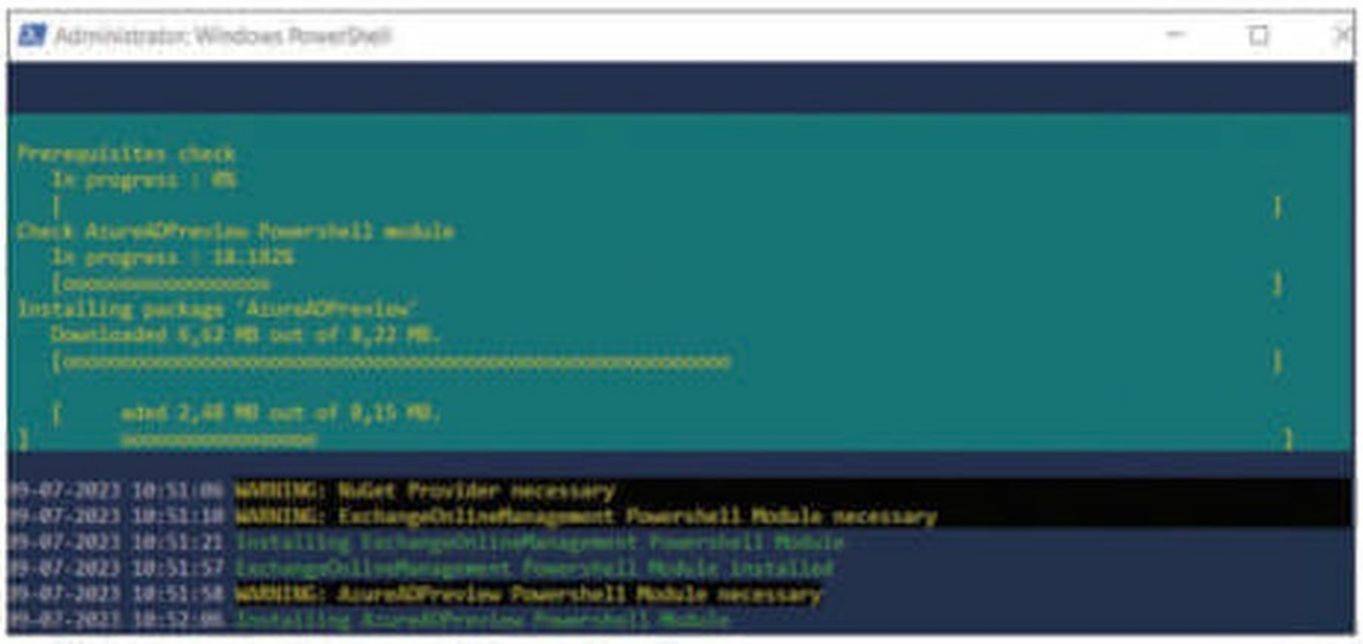


Figure 7

Pour exécuter Harden 365 par défaut, il suffit d'exécuter le script PowerShell *Start-Harden365.ps1*.

Si vous souhaitez personnaliser votre configuration, vous pouvez modifier le fichier *Json* dans le dossier *Configs*.
À la première exécution, Harden 365 va installer tous les modules PowerShell requis. **Figure 7**

Secure by design : à la recherche des fuites mémoires en .NET

Christophe Nasarre

Christophe construit un profiler pour les applications .NET chez Datadog. Il contribue à la communauté des développeurs .NET en tant que Microsoft MVP.

Cela fait presque 12 ans que j'ai développé LeakShell pour m'aider à automatiser la recherche de fuites mémoires en .NET. L'idée était simple : comparer 2 dumps mémoires d'une application avec des fuites afin de lister les types dont le nombre d'instances ne fait que croître. Aujourd'hui, vous pourriez utiliser la fonctionnalité de Memory Usage de Visual Studio pour obtenir la même chose et avec une bien meilleure interface utilisateur ! La fonctionnalité supplémentaire qui tue est la possibilité de voir les chaînes de références expliquant pourquoi un objet qui fuit reste en mémoire.

Comment détecter des fuites de mémoire

La détection de fuites mémoire revient à être capable de savoir quels objets restent en vie, garbage collection après garbage collection :

- implémenter ICorProfilerCallback::ObjectAllocated pour garder la trace de TOUS les objets dans le tas,
- utiliser ICorProfilerCallback::MovedReferences2 afin de modifier les adresses lorsque les objets survivants sont déplacés durant les garbage collections compactantes,
- puisque ICorProfilerCallback::ObjectReferences est appelé pour chaque objet survivant, nettoyer votre liste d'objets vivants dans le tas.

Le premier défaut de cette solution est que pour appeler ces fonctions, la CLR doit désactiver les GC concurrents avec un probable impact sur les performances. Si vous ne pouvez pas trouver une fuite en production qui mène à des crashs en « out of memory », faire tourner une instance de l'application dans ce mode semble parfaitement acceptable. Le second défaut est la complexité de suivre ces objets à la trace après chaque GC compactante.

C'est pourquoi j'ai implémenté en .NET 7 un nouvel ensemble de fonctions dans ICorProfilerInfo13 afin d'imiter ce que vous pouvez faire en C# avec une *weak reference*:

- CreateHandle : crée un weak handle pour encapsuler un objet,
- GetObjectIDFromHandle: récupère l'adresse de l'objet encapsulé ou null si l'objet n'est plus dans le tas,
- DestroyHandle: supprime un weak handle.

Créer un tel weak handle pour les objets alloués nous libère de la complexité de devoir suivre les changements d'adresses suite aux GC compactantes. Cependant, vous ne devriez pas créer un handle pour CHAQUE objet alloué parce que cela ralentirait drastiquement les garbage collections. Ainsi, l'étape suivante est d'écouter les messages AllocationTick envoyés par la CLR et créer un weak handle pour chacune de ces allocations échantillonnées. Même si la distribution statistique de ces échantillons pris tous les 100 KB n'est pas parfaite, les objets qui restent en vie devraient apparaître.

Après chaque garbage collection détectée dans ICorProfilerCallback2::GarbageCollectionFinished ou les événements spécifiques du GC, vous devriez pouvoir nettoyer cette liste des objets alloués en y supprimant ceux pour lesquels **GetObjectIDFromHandle** retourne null.

Prenez le temps de regarder l'implémentation correspondante dans le code du profiler .NET de Datadog.

Reconstruire la chaîne des références jusqu'à la racine

Même s'il est possible de récupérer la pile d'appels qui a mené à l'allocation de l'objet qui fuit grâce aux événements AllocationTick, il serait plus intéressant de connaître pourquoi il reste en mémoire. L'étape suivante est donc de reconstruire la chaîne des références depuis une racine.

Il est possible, pour un objet donné, de récupérer la liste de ses champs et de reconstruire un graphe de dépendances des parents jusqu'à ses enfants. Cependant, vous êtes intéressé par l'opposé : cela nécessiterait de récupérer ces références parents/enfants pour TOUS les objets du tas. Et cela n'est pas possible avec juste l'événement **AllocationTick**...

C'est là que la fonction ICorProfilerCallback::ObjectReferences nous sauve :

```
HRESULT ObjectReferences(
  ObjectID objectId,
  ClassID classId,
  ULONG   cObjectRefs,
  ObjectID objectRefIds[]
);
```

Cette fonction est appelée durant une garbage collection pour tous les objets (par le premier paramètre, **objectId**) toujours en vie en listant ses champs qui référencent des objets sur le tas (par le dernier paramètre **objectRefIds**).

Vous pouvez stocker chaque objet dans un **ObjectNode** :

```
struct ObjectNode
{
public:
  ObjectNode(ObjectID objectId);

public:
  ObjectID instance;
  std::vector<ObjectNode*> rootRefs;
};
```

dans un vecteur qui représente le tas.

Pour chaque champ, recherchez son nœud correspondant dans le vecteur et ajoutez le nœud de son parent (donné par **objectId**) dans son vecteur de parents **rootRefs**. Ainsi, vous reconstruisez une graphe des références inversées : **figure 1**

Les petites flèches bleues montrent les références parent/enfant données par **ObjectReferences** et les violettes plus larges permettent de construire le graphe des références inversées qui vous intéressent.

Vous savez que tous les objets vivants dans le tas ont été énumérés lorsque la fonction ICorProfilerCallback2::GarbageCollectionFinished a été appelée. Il est maintenant temps de reconstituer la chaîne des références pour tous les objets échantillonnés toujours en vie (grâce à la fonction GetObjectIDFromHandle qui ne retourne pas null dans ce cas de figure).
Il est important de comprendre qu'il s'agit bien d'un graphe et non pas un arbre, car les cycles sont possibles en .NET. **Figure 2**

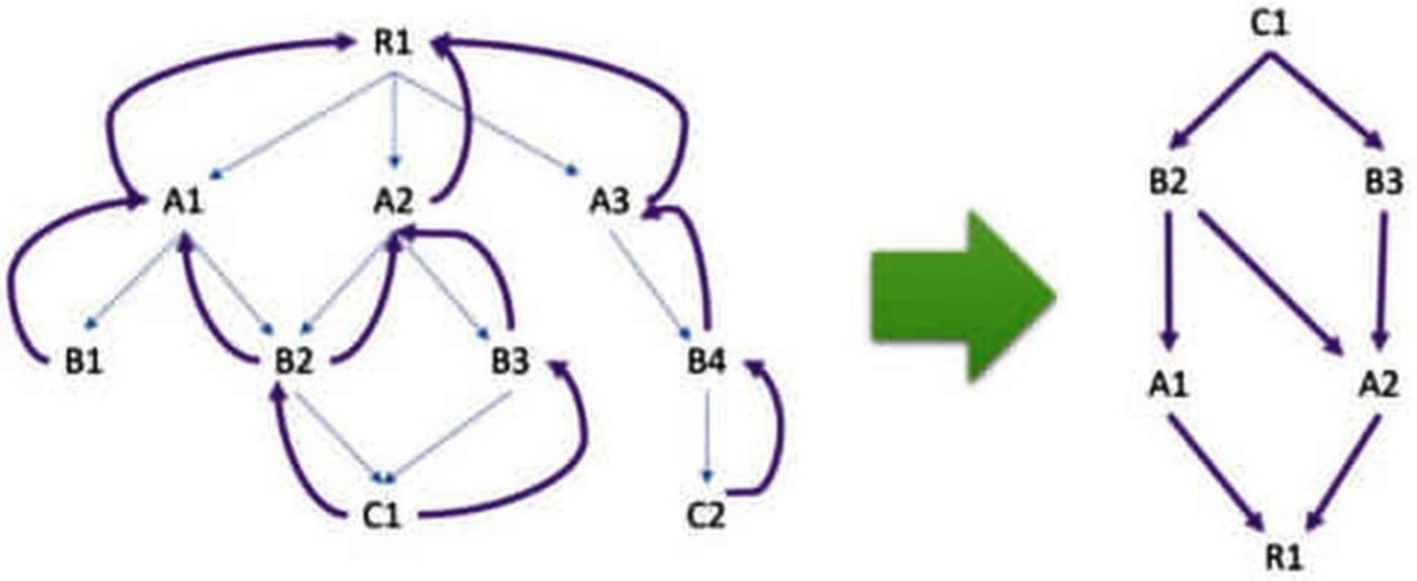


Figure 1

C'est commun dans les situations où les objets ont besoin de garder une référence sur leur « parent ». Cela signifie que ces cycles doivent être détectés au moment de la recherche des listes de référence d'un objet donné afin d'éviter des récursions infinies:

```
bool DumpNode(ObjectNode* node, std::vector<ObjectID>& referenceStack)
```

La fonction de parcours **DumpNode** prend un nœud (c.-à-d. un objet dans le tas) et une pile dans laquelle les parents sont ajoutés les uns après les autres au fur et à mesure que l'on descend dans le graphe.

Code complet sur programmez.com & github

Si un nœud parent est déjà dans la pile, un cycle est détecté et ce chemin du graphe n'est pas suivi. Une fois qu'une racine est atteinte, la pile est affichée comme le montre l'affichage suivant :

```
OnGarbageCollectionFinished: 3859 objects in the heap.

OnRootReferences2: 90/109 roots.
        stack:40
    finalizer:1
       handle:49
        other:0
_____________________

21266c00020 | H - 0 = Object[]
—> 21268c092c8 = NativeRuntimeEventSource
—> 21268c3fbe8 = EventSource.EventMetadata[]
—> 21268c18010 = ParameterInfo[]
—> 21268c17ef0 = RuntimeParameterInfo
—> 21268c0ed58 = RuntimeMethodInfo
—> 21268c0e708 = RuntimeType.RuntimeTypeCache
—> 21268c0e840 = RuntimeType.RuntimeTypeCache.MemberInfoCache<System.Reflection.RuntimeMethodInfo>
—> 21268c14978 = RuntimeMethodInfo[]
—> 21268c10d70 = RuntimeMethodInfo
—> 21268c29720 = Signature
—> 21268c29770 = RuntimeType[]
=======================================
```

Comme on peut le voir, il est possible de fournir les détails du type des racines qui tiennent en vie la chaîne des références grâce à ICorProfilerCallback::RootReferences2:

```
HRESULT RootReferences2(
  ULONG cRootRefs,
  ObjectID rootRefIds[],
  COR_PRF_GC_ROOT_KIND rootKinds[],
  COR_PRF_GC_ROOT_FLAGS rootFlags[],
  UINT_PTR rootIds[]
);
```

Cette fonction est appelée avec 3 tableaux de long **cRootRefs** synchronisés contenant pour chaque racine :

- l'adresse (**rootRefsIds** objectID),
- son type (**rootKind** pour une variable de la stack, un finalizer, un handle et autres)
- et des flags (**rootFlags** pour pinned, weak reference interior ou ref counted).

Ils sont stockés dans un vecteur de **ObjectRoot**:

```
struct ObjectRoot
{
public:
  ObjectID instance;
  COR_PRF_GC_ROOT_KIND kind;
  COR_PRF_GC_ROOT_FLAGS flags;
};
```

Maintenant, quand un nœud est trouvé sans son parent, son **objectId** est utilisé pour trouver la racine correspondante dans le vecteur et en montrer les détails.
Durant mes expérimentations, j'ai noté que plusieurs chaînes peuvent exister et qu'il est bien plus rapide de s'arrêter sur la première.

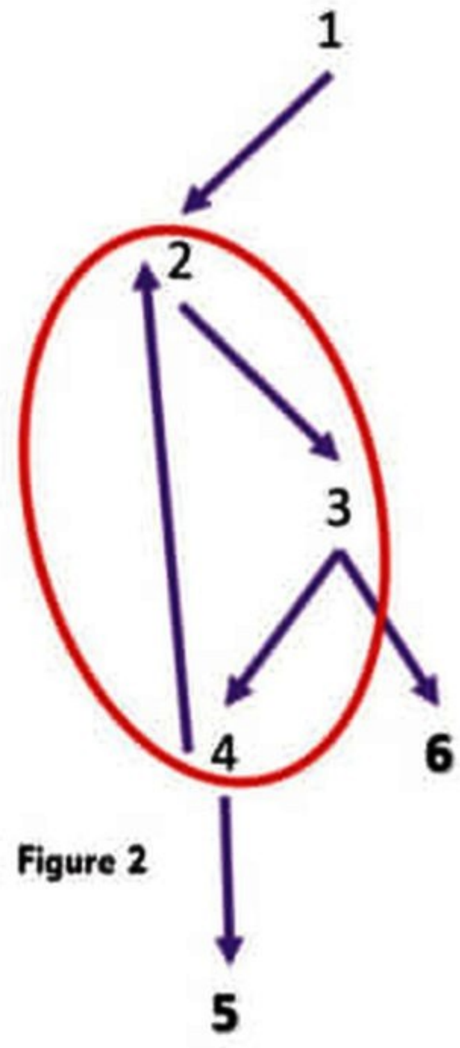


Figure 2

Références

Automate the search of memory leaks with LeakShell
Building your own .NET memory profiler in C#
Introduction au profiling .NET avec ICorProfilerCallback
Pull request dans .NET 7 pour ICorProfilerInfo13 permettant de créer des weak handles
Implémentation du Live Heap Profiler de Datadog

Quelques principes du secure by design côté développeur

Riadh MNASRI

Technical leader, Développeur Sénior passionné, ayant plus de 19 ans d'expérience, Riadh s'intéresse beaucoup aux pratiques Craftsmanship (TDD, BDD, DDD, ...) et au clean code. Il aime bien apprendre et partager ses connaissances, coacher techniquement et surtout concevoir et coder des systèmes robustes et résilients. Il est à fond sur la programmation fonctionnelle autour de Java et de Kotlin. Il a aussi une grande appétence autour de l'infrastructure notamment côté Cloud (GCP et AWS).

https://twitter.com/riadhmnasri

https://www.linkedin.com/in/riadhmnasri/

L'architecture doit être pensée pour prendre en compte en amont les exigences de sécurité. L'écriture d'un code qui respecte les bonnes pratiques devient la suite logique de cette approche. Dans cet article, on va dérouler quelques bonnes pratiques de programmation dit secure by design.

Règles de bases pour écrire un code sécurisé

Écrire un code simple

L'idée derrière cette pratique est d'essayer de toujours d'écrire un code simple et lisible. Plus le code est complexe plus le risque d'avoir des vulnérabilités, et des bugs est important.

Comment écrire un code simple ?

C'est facile à dire mais qu'en est-il si le besoin fonctionnel ou l'algorithme est complexe ? Le mot d'ordre est diviser pour régner. Les règles suivantes peuvent aider à éviter une complexité accidentelle :

- Garder les classes et les méthodes les plus petites possibles
- Limiter les interactions
- Respecter les règles du code propre (principes S.O.L.I.D, KISS, YAGNI...)
- Utiliser les librairies et frameworks testés et utilisés par la communauté (regardez-leur nombre d'étoiles sur Gitab, la fréquence des commits, le nombre de contributeurs, le nombre de tickets « issues » ouverts, etc.) : bref évitez d'utiliser des libs et frameworks trop récents ou peu utilisés
- Assurer la non-obsolescence des librairies : mettre à jour régulièrement les dépendances externes afin d'éviter d'avoir dans ses apps des dépendances avec des failles de sécurité connues.
- Refactorer son code régulièrement et adopter des approches de code qui aide à bien faire ceci. L'approche la plus connue et qui a fait ses preuves, est l'approche TDD « Test Driven Design ».

Déléguer le développement du code relatif à la sécurité

Maîtriser la sécurité pour un développeur est tout sauf simple. Il ne faut pas se leurrer : on maîtrise que ce que l'on sait (bien) faire. Il est prudent et raisonnable de ne jamais coder ses librairies de sécurité soi-même à moins qu'on soit expert dans le domaine et qu'on maîtrise parfaitement. Mon conseil est de privilégier des librairies et des plateformes qui ont fait leur preuve. Coder son propre algorithme cryptographique semble être une aventure qui n'est pas sans risque. En effet, on peut introduire inconsciemment et accidentellement des failles de sécurité aux lourdes conséquences de sécurité.

Limiter la durée de vie des données sensibles ?

Il arrive régulièrement qu'on manipule des données sensibles. Un hacker qui peut exécuter du code au sein du même système pourra facilement avoir accès à ces données.

Il est fortement recommandé de limiter la durée de vie de ces données. Les objets qui contiennent des données sensibles doivent être supprimés dès que leur utilisation est terminée. D'autre part, il ne faut jamais « journaliser » des données sensibles.

Ne jamais stocker une donnée sensible non cryptée côté client

Stocker des données sensibles non cryptées côté client s'est s'exposer. Par exemple, l'utilisation des applications web nécessite le stockage d'informations sensibles dans un fichier texte léger côté client. Ce fichier s'appelle un cookie. Il est créé par le serveur suite à l'authentification du client. Ce fichier sera stocké un certain temps côté client. Si le client souhaite se reconnecter, il transmet ce fichier dans sa requête https, ce qui lui permet d'avoir accès à des ressources côté serveur. Les cookies ne protègent pas contre des attaques XSS (Cross-site scripting). En effet, pour limiter ce risque, il faut limiter la validité de ce fichier et ne jamais stocker des données sensibles en clair. Ces données peuvent être un identifiant, un mot de passe, un numéro de sécurité sociale... Une attaque de type XSS peut être schématisée comme suit : **Figure 1**

Figure 1

Utiliser tant que possible des classes immuables

Coder en utilisant des classes immuables est bénéfique afin d'avoir un code robuste. Un objet en paramètre d'une méthode ne pourra pas être modifié par l'appelant. Si la méthode qui utilise une instance de cette classe a besoin de modifier la donnée portée par cette classe, elle sera obligée de créer une copie. Cette pratique protège les données contre une modification de l'objet reçue par l'appelant :

```
public void process(Query query) {
  var newQuery = new Query(query.name);
  //process newQuery
}
```

Valider les arguments des méthodes sensibles

Les méthodes, qui contiennent du code qui manipule des

données sensibles, doivent obligatoirement contrôler les paramètres en entrée. Elles ne doivent pas autoriser des paramètres null. Il est recommandé de leur donner des valeurs par défaut afin d'éviter d'avoir des effets de bords imprévisibles.

```
public void authenticate(SecurityContext context) {
  if (context == null) {
    throw new IllegalArgumentException("Missing security context");
  }
```

Empêcher l'upload de fichier malicieux

Les applications qui autorisent l'upload de fichiers s'exposent à des risques à stocker et exécuter du code malicieux. Il est primordial de contrôler les fichiers avant de les traiter. Ce contrôle consiste à bien vérifier les métadonnées du fichier, interdire certains types de contenu dangereux. Par exemple : un script shell ou fichier .exe. Il faut également analyser minutieusement le contenu du fichier.

Empêcher l'injection de code

Contrôler les entrées de notre système permet de se protèger contre l'insertion et l'exécution de code malicieux. Ce réflexe est indispensable en partant du principe qu'il ne faut jamais faire confiance à une donnée qui vient de l'extérieur.

Pour ce faire, il faudra systématiquement éviter ou encoder correctement les caractères spéciaux en entrée, et également en sortie. On a souvent tendance à négliger ce point, notamment pour des librairies qui permettent de générer du code interprété.

Par exemple, générer du code HTML côté serveur et l'interpréter code client. Si on ne le fait pas, on s'expose à des attaques de type injection code/XSS (Cross Site Scripting). Heureusement, la plupart des librairies Java nous facilitent la tâche et nous aident à contrôler les paramètres en entrées.

Voici un exemple de code, où on échappe le contenu d'un fichier en entrée :

```
String str = FileUtils.readFileToString(new File("input.txt"));
String results = StringEscapeUtils.escapeJava(str);
System.out.println(results);
```

Les injections les plus courantes sont les injections SQL. Afin de s'y prémunir, il faut éviter de construire soit même sa requête et préférer déléguer ça aux libraires de référence sur le sujet. Voici un exemple, ou on construit une requête SQL nous-mêmes :

```
String query = "SELECT \* FROM accounts WHERE custID='" + request.getParameter
("id") + "'";
```

Par exemple, les ORM (Object Relational Mapping) librairies connues gèrent pour nous les aspects injections SQL en passant par les requêtes paramétrées :

```
String hql = "FROM Customer WHERE id LIKE 'id'";
```

Ne jamais utiliser clone() pour copier des données non sûres

Quand une classe reçoit en paramètre de sa méthode un objet fourni par une classe externe, il est généralement conseillé de faire de la programmation défensive. En effet, il faut éviter de traiter directement l'objet reçu. On sera tenté de l'utiliser la méthode clone() fournie par la classe reçue. Une telle pratique est très risquée puisqu'elle peut causer le contournement des contrôles de sécurité.

La règle est de ne jamais utiliser la méthode done() d'une classe non finale pour faire une copie. Dans un tel cas, la méthode clone peut-être redéfinie par une classe fille et introduire ainsi un code potentiellement dangereux.

Ne pas utiliser Objects.equals() pour comparer des clés cryptographiques

Une erreur fréquente est d'utiliser la méthode equals() pour comparer directement les clés cryptographiques. La plupart des clés ne passent pas cette comparaison. Cette clé cryptographique est un objet composé. Afin d'avoir une comparaison correcte, il faudra comparer les composants de cette clé un par un. Ainsi, cette implémentation est à proscrire :

```
private static boolean areKeysEqual(Key key1, Key key2) {
  if (key1.equals(key2)) {
    return true;
  }
  return false;
}
```

Il faut plutôt privilégier une implémentation de ce style :

```
private static boolean areKeysEqual(Key key1, Key key2) {
  if (key1.equals(key2)) {
    return true;
  }

  if (Arrays.equals(key1.getEncoded(), key2.getEncoded())) {
    return true;
  }

  if ((key1 instanceof RSAPrivateKey) && (key2 instanceof RSAPrivateKey)) {
    if ((((RSAKey) key1).getModulus().equals(
        ((RSAKey) key2).getModulus())) &&
        (((RSAPrivateKey) key1).getPrivateExponent().equals(
          ((RSAPrivateKey) key2).getPrivateExponent()))) {
      return true;
    }
  }
  return false;
}
```

Ne pas utiliser des algorithmes de cryptographiques non sûrs

Les machines modernes ont atteint des capacités de calcul extrêmement impressionnantes. Il devient plus facile de casser des algorithmes de cryptage par « brut force ». DES (Data Encryption Standard) est considéré comme un algorithme vulnérable. Par ailleurs, il faut se renseigner et s'assurer qu'un algorithme est sûr et fiable avant de l'utiliser.

Ne jamais stocker des mots de passe en clair

Une règle élémentaire quand on code est de ne jamais stocker les mots de passe ni en dur dans le code ni en clair dans une base de données par exemple. La technique souvent utilisée est de stocker le mot de la passe en passant par une fonction de hashage.

Quelques techniques de programmation défensive

Minimiser l'accessibilité aux classes et à leurs membres

Il est recommandé de donner par défaut le minimum de niveau d'accessibilité aux classes et aux méthodes. L'objectif de limiter la possibilité à un code malicieux d'appeler notre code sensible. Il faut privilégier le modificateur « private » ou « protected » autant que possible et ne laisser public que le contrat d'interface exposé à l'extérieur.

Il faudra également déclarer les classes finales par défaut afin d'empêcher un code mal-intentionné de créer des classes filles de notre classe mère contenant un code sensible notamment un code qui gère la sécurité.

Annoter les classes thread-safe

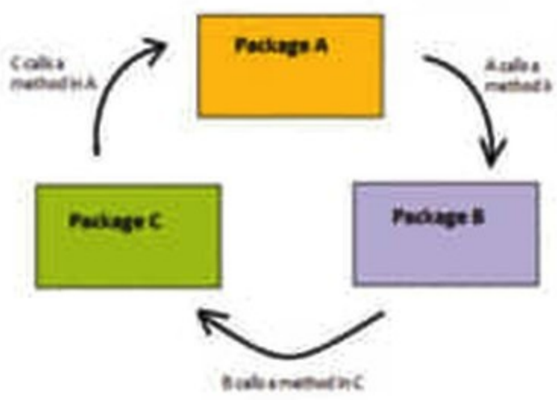
Java offre des annotations qui permettent de donner une étiquette à un comportement important d'une classe ou d'une méthode. Par exemple, on peut ajouter une annotation @ThreadSafe à une classe connue qu'on a codée et qu'on est sûr qu'elle ne pose pas de problème de « thread safety ». Dans le cas contraire, on annote la classe @NotThreadSafe si elle est publique et censée être utilisée dans un contexte multi-threading. On peut également annoter une classe immutable avec l'annotation @Immutable

Ces annotations donnent des informations précieuses aux lecteurs/utilisateurs du code afin de l'orienter sur le bon usage de la classe qu'il manipule.

Eviter les dépendances cycliques

En Java, les dépendances cycliques (ou boucles de dépendances) entre les classes ou les packages sont considérées comme un inconvénient majeur. Une dépendance cyclique se produit lorsque deux, ou plusieurs classes ou packages dépendent les uns des autres de manière circulaire, créant ainsi une boucle. Voici un schéma qui illustre ce que c'est une dépendance cyclique : **Figure 2**

Figure 2



Les inconvénients :

- Les dépendances cycliques rendent le code plus difficile à comprendre et à maintenir, car plus compliqué de suivre le flux logique et les interactions entre les classes.
- Complexité accrue : les dépendances cycliques augmentent la complexité du code et peuvent entraîner des problèmes de conception et plus difficile à résoudre.
- Difficulté à effectuer des modifications : lorsqu'il y a des dépendances cycliques, il devient risquer de faire des modifications dans une classe ou un package, car cela peut entraîner des effets inattendus dans d'autres parties du code.
- Augmentent les risques d'erreurs, car un changement dans une classe peut entraîner des erreurs dans d'autres classes liées par des dépendances cycliques.
- Les dépendances cycliques entraînent un couplage fort entre les classes ou les packages, ce qui rend le code moins flexible et plus difficile à réutiliser. NDLR : le couplage fort est une pratique à proscrire.

Pour éviter les dépendances cycliques, il est essentiel de suivre les principes de conception : principe de responsabilité unique (SRP), principe d'inversion de dépendance (DIP) ou encore principe d'inversion de contrôle (IoC). Ils aident à favoriser une conception modulaire et à réduire les dépendances entre les classes et les packages. Vous pouvez aussi utiliser des modèles de conception tels que l'injection de dépendances (DI).

Prévoir un « graceful recovery » d'une erreur système d'un code sensible

En Java, les exceptions sont classées en 2 catégories Error ou Exception. Les Erreurs sont généralement relatives à des problèmes système (Exemple : OutOfMemoryError). Les Exceptions sont généralement applicatives.

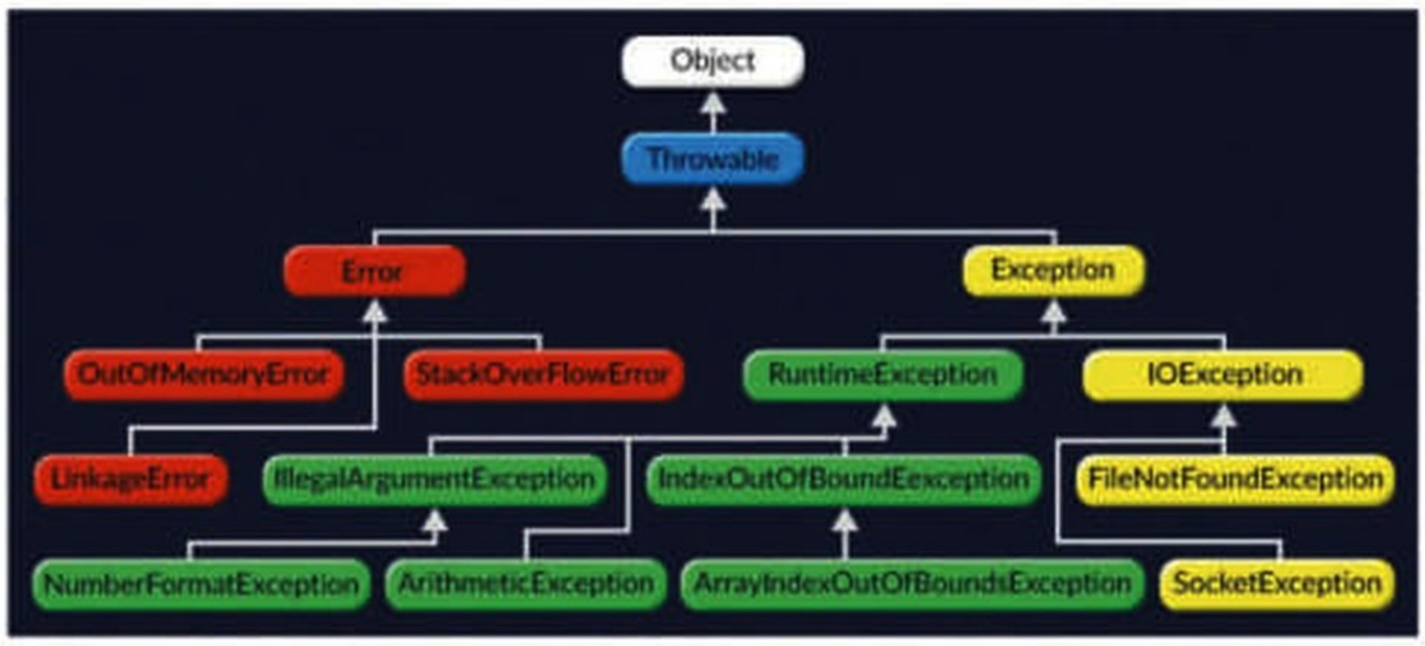
Le schéma 3 illustre cette nomenclature.

Généralement, il est déconseillé de catcher les erreurs, car cette approche rend plus difficile la prise en charge d'exception spécifique. En revanche, s'il y a un risque de fuite de données sensibles, il est autorisé de catcher les erreurs d'une manière similaire au code suivant :

```
try {
  confidentialCodeCall();
} catch (Throwable t) {
  // Forward to handler
} finally {
  // Free cache, release resources
}
```

Ce qui permettra d'éviter un crash violent de la JVM suite à une attaque de type DOS « *Denial of Service* ».

Figure 3



Conclusion

Dans cet article, j'ai listé quelques bonnes pratiques et règles qui nous permettent d'écrire un code Java robuste et sécurisé. Bien entendu, ces conseils sont valables pour la plupart des langages.

En utilisant les mécanismes de sécurité fournis par Java, tels que déléguer au maximum le développement des règles de sécurité usuelles à des librairies, mettre en place les mécaniques de contrôle anti-injection de code, l'authentification appropriée et la validation des entrées, nous pouvons prévenir les attaques courantes telles que les attaques de type DOS « Denial Of Service », les attaques par injection SQL, etc.

De plus, en respectant les normes de codage et en suivant les principes du secure by design, on garantit une base de code solide pour la maintenance et une réduction des vulnérabilités dans les futures évolutions de son app.

A vous de bien coder !

Comment l'IA peut aider les développeurs et le RSSI à produire des logiciels sécurisés ?

Imaginez produire des solutions innovantes et améliorer leurs délais de mise à disposition tout en améliorant votre posture de sécurité et en protégeant vos données sensibles. Nous verrons ici comment cet état idéal devient possible avec l'aide de l'IA.

Grégory Domagala

Grégory est Solution Architect chez Veracode, expert dans le domaine de la sécurité applicative. Avec 22 ans d'expérience professionnelle dans les domaines du développement et de la sécurité , il met ses compétences au service de ses clients afin de les aider à mettre en place des solutions de sécurité applicative intégrées dans leur cycle SDLC/DevOps.

VERACODE

Alors que l'IA est devenue un sujet tendance à l'échelle mondiale, notre mission d'associer le développement et la sécurité dans la conception de logiciels sécurisés nous a amenés à prendre l'IA en considération. Aujourd'hui, alors que nous continuons à être les pionniers de la sécurité logicielle intelligente, nous avons pour mission de partager notre vision de la façon dont l'IA peut aider à sécuriser les logiciels du monde entier.

Avant de regarder en détail la façon dont les équipes de développement et de sécurité peuvent exploiter la puissance de l'IA pour produire des logiciels sécurisés, commençons par un aspect clé de cette conversation : l'éthique et l'IA.

Le rôle de l'IA dans la cybersécurité

Bien que l'IA présente de nombreux avantages potentiels, elle est exploitée par des pirates pour mener des attaques plus sophistiquées et ciblées. Les pirates informatiques malveillants utilisent l'IA pour automatiser les attaques, effectuer des vols de données, échapper aux systèmes de détection et s'engager dans les systèmes sociaux. Par exemple, des algorithmes d'intelligence artificielle sont utilisés pour générer des logiciels malveillants polymorphes qui peuvent muter en temps réel pour éviter d'être détectés par un logiciel antivirus ou d'autres outils de sécurité.

Ces dernières années, les pirates malveillants se sont lancés dans l'ingénierie sociale, où ils utilisent des algorithmes d'intelligence artificielle pour générer des e-mails, des appels vocaux ou des SMS de phishing plus convaincants. Les forums du Dark Web sont occupés par des discussions sur la façon d'utiliser ChatGPT pour générer des e-mails d'hameçonnage. Le comportement en ligne et l'activité des médias sociaux d'une victime sont surveillés afin que les attaquants puissent rédiger des messages qui sont plus susceptibles de réussir à inciter une victime à divulguer ses informations personnelles ou confidentielles.

L'impact sur les entreprises

Dans les entreprises, les réponses automatisées seront la seule ligne de défense.

- Au cours des 5 prochaines années, les coûts mondiaux de la cybercriminalité devraient augmenter de 23 % par an, pour atteindre 23,84 trillions de dollars par an d'ici 2027. (Statista)
- Les prévisions montrent que les entreprises perdront environ 10,5 billions de dollars en 2025, à raison de 19 977 168 dollars par minute en raison de la cybercriminalité. (Cybercrime Magazine)
- Les entreprises publiques perdent environ 8,6 % de leur valeur après une cyber-attaque. (Comparitech)
- 45,5 % des personnes interrogées lors d'un récent sondage ont déclaré que leur organisation avait subi entre 1 et 5 cyberattaques réussies au cours de l'année écoulée. (Statista)

L'Architecture de Transformateur et ChatGPT

L'Architecture de Transformateur est un type de réseau neuronal qui a été introduit dans l'article " Attention Is All You Need " par Vaswani et al. en 2017. Ce modèle est à la base de l'implémentation bien connue, Chat GPT. L'architecture est basée sur le concept d'auto-attention, qui permet au modèle de peser l'importance des différentes parties d'une séquence d'entrée lorsqu'il fait des prédictions.

L'Architecture de Transformateur s'est révélée très efficace pour un large éventail de tâches de traitement du langage naturel, notamment la traduction automatique, la modélisation du langage et la réponse aux questions. Son succès a conduit au développement de nombreux modèles ultérieurs, dont la populaire série de modèles de langage GPT.

Il existe de nombreux types d'IA, et dans cet article, nous nous concentrerons sur Chat GPT, l'architecture Transformateur et les implications qu'ils auront sur l'avenir des tests de sécurité des applications. L'un des principaux avantages de Chat GPT est sa capacité à générer des réponses à consonance très naturelle qui sont souvent impossibles à distinguer de celles écrites par un humain. Cela en fait un outil utile pour les entreprises et les organisations qui cherchent à améliorer les interactions avec leurs clients et à fournir des expériences plus personnalisées.

L'IA et l'avenir de la sécurité applicative

L'IA devient de plus en plus essentielle à la cybersécurité et aux tests de sécurité des applications. Les organisations ont besoin de l'IA pour sécuriser les actifs qui alimentent leur activité. Les cyberattaques étant de plus en plus sophistiquées et répandues et exploitant l'IA, les organisations devront s'appuyer sur des solutions de sécurité alimentées par l'IA pour protéger leurs systèmes et leurs données.

Les organisations de toutes tailles devront accepter l'idée que les logiciels peuvent changer et changeront à chaque étape du cycle de vie du développement logiciel (SDLC). Une approche globale de la sécurité couvrant les phases de planification, de conception, de construction, d'intégration, de livraison et de production du cycle de vie du développement

logiciel sera d'une importance cruciale pour la défense de chaque organisation contre les risques liés à la sécurité.
Les pirates ont adopté l'IA pour lancer des attaques sur les logiciels vulnérables et le feront de plus en plus à l'avenir. Une approche manuelle de la sécurité des logiciels sera intenable et les organisations devront adopter une solution de sécurité automatisée avec l'historique et l'intelligence nécessaires pour identifier et remédier automatiquement aux risques en fonction des décisions politiques.

La nouvelle génération de tests de sécurité des applications

Chat GPT représente une avancée significative dans le domaine du traitement du langage naturel. En outre, il a le potentiel de révolutionner la façon dont les humains interagissent avec les ordinateurs et les assistants numériques. De nombreuses entreprises, comme Veracode, tirent parti de cette technologie pour automatiser la résolution des risques liés à la sécurité des applications.
Veracode Fix est basé sur l'architecture Transformateur qui est un type de modèle d'apprentissage profond utilisé dans le traitement du langage naturel (NLP) qui a été introduit par des chercheurs de Google en 2017. Elle est depuis devenue une architecture largement utilisée dans le NLP, alimentant de nombreux modèles de langage de pointe utilisés aujourd'hui.
L'avenir des tests de sécurité des applications sera profondément ancré dans les réponses de l'IA aux attaques courantes. Alors que les pirates utilisent l'IA pour exploiter les vulnérabilités des applications de plus en plus fréquemment, les organisations doivent utiliser des outils et des technologies qui leur permettent de répondre rapidement, intelligemment et avec un ensemble de règles qui régissent ces réponses. L'implémentation de l'IA par Veracode permet justement d'atteindre cet objectif grâce à Veracode Fix.

Les défis

Les bases de données publiques sur les vulnérabilités, telles que la base de données nationale américaine sur les vulnérabilités (NVD), sont par nature incomplètes, en particulier lorsqu'il s'agit de logiciels libres, qui se comptent par millions. Il existe de nombreuses vulnérabilités connues qui peuvent n'être connues que de l'équipe spécifique du projet open source ou même du développeur individuel.
Le prédicteur de vulnérabilité est formé sur la base des informations disponibles lorsqu'un changement de code est effectué pour corriger une vulnérabilité. Ces informations sont la modification du code, y compris les commentaires et le commentaire de validation. Grâce à ces informations, Veracode peut prédire si la modification est pertinente ou non du point de vue de la sécurité. Ce prédicteur peut fonctionner quotidiennement à grande échelle pour analyser tous les dépôts de logiciels libres utilisés par les clients de Veracode. Les clients de Veracode ont ainsi la possibilité de mettre à jour les paquets pour remédier à leur logiciel, que la vulnérabilité figure ou non dans une base de données publique de vulnérabilités.

Veracode Fix et tests statiques de sécurité

La qualité de presque toutes les IA dépend de l'ensemble de données sur lequel elles ont été formées. De grands modèles linguistiques sont utilisés pour former le Chat GPT. Dans le cas de Veracode, les correctifs proposés pour la sécurité des applications et produits par le modèle ont fait l'objet d'un entraînement poussé, s'appuyant sur les 17 années de données et d'expérience de Veracode en matière d'analyse et de correction des vulnérabilités de sécurité.
Veracode Fix analyse automatiquement le code du développeur à la recherche de vulnérabilités et propose des solutions aux CWE (dictionnaire de vulnérabilités logicielles) et produit des recommandations très précises. Bien que les développeurs puissent coller le code de leur application dans Chat GPT et lui demander d'identifier les risques de sécurité (ce que nous ne recommandons pas !), les résultats sont loin d'être satisfaisants. Le modèle est capable d'identifier les vulnérabilités de base, mais il n'a pas été affiné avec des décennies de connaissances en matière de sécurité, comme c'est le cas pour l'ensemble des données de base de Veracode.
Veracode utilise de larges modèles de langage GPT comme base de notre implémentation de l'IA avec Veracode Fix. GPT signifie "Generative Pre-trained Transformer" et est un type de modèle de langage développé par OpenAI et basé sur l'architecture Transformateur.

CAS D'USAGE :

correction d'un code généré par ChatGPT avec Veracode Fix

Pour illustrer les implications réelles en matière de sécurité du code généré par ChatGPT et l'utilisation de Veracode Fix, considérons un cas d'utilisation réel impliquant une tâche simple : créer une application Java qui enregistre les entrées fournies par l'utilisateur dans un logger standard. Comme prévu, en quelques secondes seulement, ChatGPT génère du code qui répond aux exigences fonctionnelles de notre requête (cf. **figure : 1**)
Pour les habitués du code sécurisé, ceux-ci remarqueront un problème avec le code généré par ChatGPT. Bien qu'il réponde aux exigences fonctionnelles de notre requête, le code contient une faille de sécurité telle qu'écrite ci-dessous :
Le code

```
logger.info("User input : " + userInput);
```

ne fait aucun assainissement de la variable 'userInput' avant de logger, ce qui la rend vulnérable à une injection CRLF via une faille CWE-117.
Ce scénario met en évidence un paradoxe critique : si le code généré par ChatGPT répond aux critères fonctionnels, mais est criblé de vulnérabilités (c'est-à-dire que ChatGPT émule le code défectueux sur lequel il a été entraîné...), il introduira des risques et/ou nécessitera un travail de remédiation fastidieux qui, finalement, annulera le gain de temps apporté par cette génération de code automatique

Veracode Fix et l'IA générative spécialisée pour la correction du code

Nous voyons les limites des outils généralistes comme chatGPT, notamment en matière de sécurité du code.
Des outils de codage spécialisés en IA générative comme Veracode Fix, conçus pour la tâche spécifique de correction sécurisée du code, offrent une solution.

En utilisant un jeu de données propriétaire et supervisé par les experts de Veracode, permettant de relier du code non sécurisé à sa version corrigée, Veracode Fix excelle dans sa tâche primaire de correction de code et offre des résultats fiables sans les problèmes de licence que l'on peut retrouver sur les alternatives formées sur du code open source.

Les compromis sont le temps et les efforts nécessaires pour développer, entraîner et aligner un modèle de données. Cependant, se concentrer sur un cas d'utilisation spécifique permet d'adopter une approche qui met l'accent sur la qualité et la spécialisation plutôt que sur l'étendue et la généralisation. **Figure 2**

En revenant sur l'exemple ci-dessus, prenons le code non sécurisé généré par ChatGPT, et lançons une analyse statique. Sur la **figure 2**, nous voyons que l'analyse statique a bien détecté l'injection CRLF.

Cependant, au lieu d'essayer de réécrire et de corriger manuellement le code (ce qui peut prendre plus de temps que simplement réécrire le code à partir de zéro), nous pouvons utiliser Veracode Fix pour proposer un correctif sur le code généré par ChatGPT afin d'avoir une application fonctionnelle et sécurisée en quelques secondes seulement sans écrire manuellement de code.

Sur la **figure 3**, nous avons invoqué Veracode Fix depuis le terminal, et celui-ci nous a proposé des corrections contextualisées à notre code sous forme de fichier patch, une fois le « fix » choisi, le fichier est modifié et corrigé.

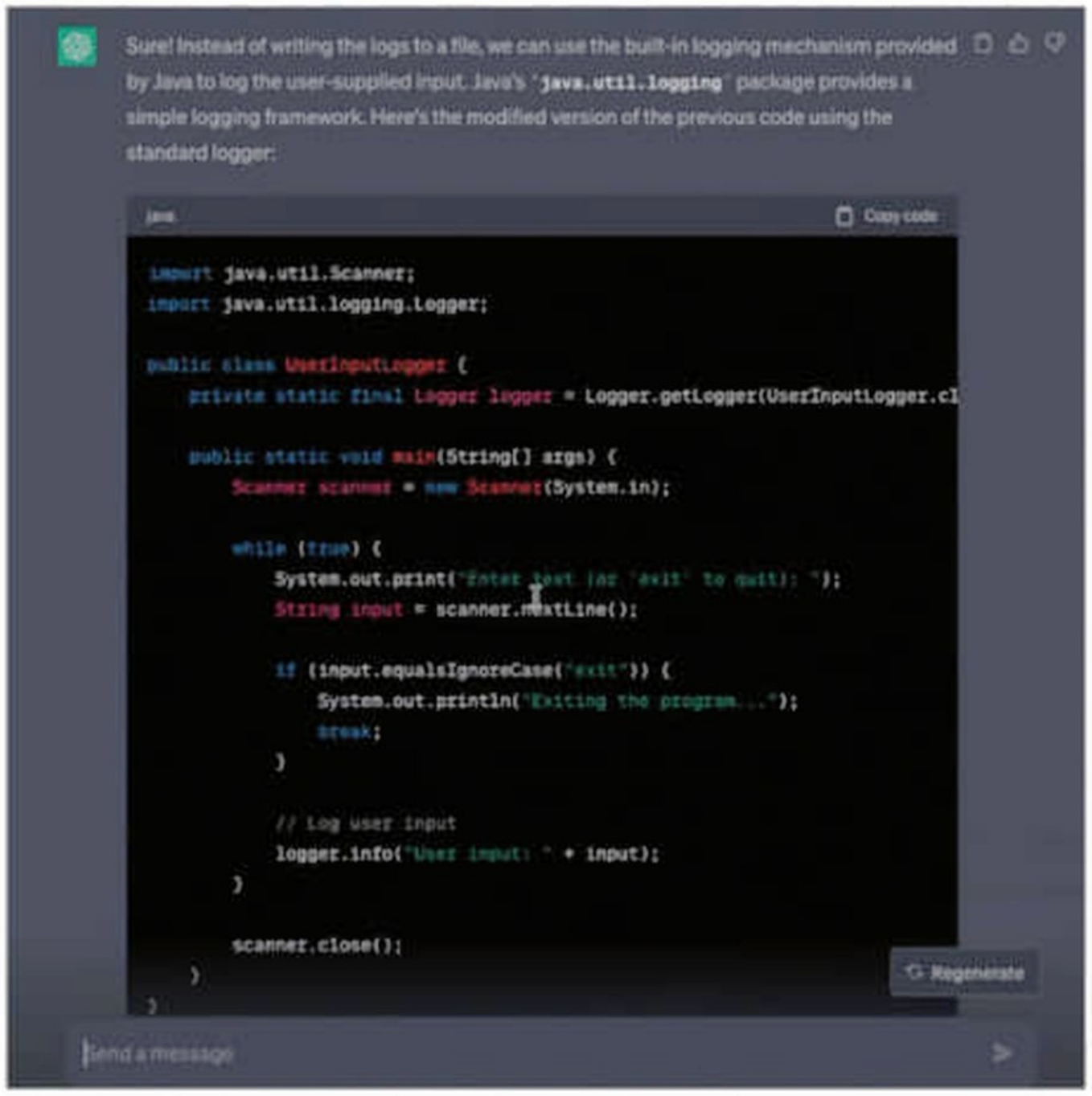


Figure 1 : *Génération par ChatGPT d'un logger standard permettant de logger une entrée utilisateur en Java*

Au-delà de l'analyse statique du code et de la sécurité native dans le Cloud

Dans un premier temps, Veracode Fix aidera les développeurs à corriger les problèmes de sécurité statiques dans tous les principaux langages de programmation. À mesure que le marché de l'IA évolue, Veracode Fix évoluera aussi afin d'offrir une valeur ajoutée sur l'ensemble du SDLC.

L'avenir de la sécurité applicative consistera moins à trouver et à remédier les vulnérabilités, mais plutôt à empêcher que les vulnérabilités atteignent la base de code et les référentiels de code source. Veracode mènera ces avancées technologiques dans les domaines suivants :

1. Empêcher les développeurs d'importer des bibliothèques ou des dépendances transitoires dans des bibliothèques open source dont les vulnérabilités sont connues, ce qui donne aux professionnels de la sécurité l'assurance que de nouvelles vulnérabilités ne sont pas introduites par l'utilisation massive et rapide de logiciels open source.
2. Infrastructure-as-code : l'interprétation intelligente des fragments de code et de leur impact négatif potentiel sur la sécurité sera essentielle pour permettre aux développeurs d'utiliser des fragments de code en toute sécurité.
3. Images de conteneurs : un mécanisme de détection complet et intelligent sera essentiel pour interdire l'adoption d'images de conteneurs qui ne sont pas sécurisées, ce qui peut conduire à des exploits "tout accès" lorsqu'elles sont exécutées en production.

Ces avancées futures constitueront une étape importante pour permettre aux développeurs de coder rapidement et en toute sécurité. En empêchant la consommation et l'inclusion d'OSS, d'images de conteneurs, de systèmes d'exploitation

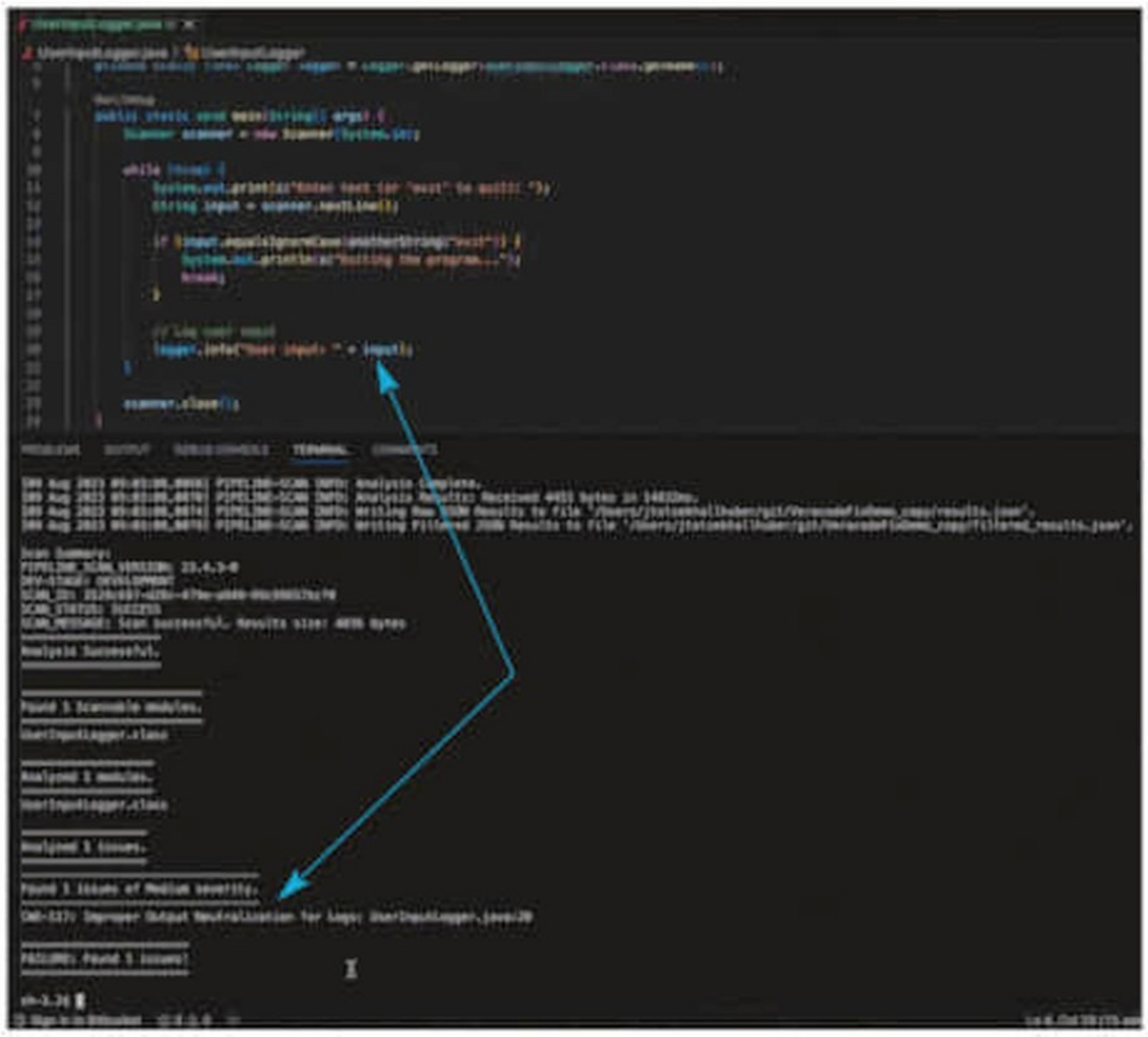


Figure 2 : *L'analyse statique Veracode a identifié la CWE 117 dans le code généré par chatGPT*

de base et de fragments de code IaC qui ne sont pas sécurisés, Veracode Fix empêchera les plus importantes vulnérabilités en matière de sécurité logicielle de se frayer un chemin jusqu'à la base de code d'une organisation.
Il s'agit là d'un grand pas en avant, car les organisations passent de l'analyse, du rapport et de la correction à des pratiques de développement préventif proactives.

En conclusion : l'avenir est prometteur!

L'impact de l'IA sur les tests de sécurité des applications ne peut être surestimé. Avec Veracode Fix, les développeurs et les équipes de sécurité disposent d'un outil puissant qui peut améliorer considérablement la sécurité de leurs applications. En automatisant l'identification et la remédiation des risques de sécurité dans le code, Veracode Fix peut sauver du temps et des ressources tout en garantissant la sécurité des applications dès le début de leur conception.
Il est clair que le futur de l'IA c'est de continuer à révolutionner la façon dont nous approchons la technologie et la sécurité. Cependant, c'est à nous d'exploiter son pouvoir de manière responsable et éthique. Nous devons travailler ensemble pour partager nos points de vue sur l'impact de l'IA sur les entreprises, la société et la réglementation gouvernementale, ainsi que sur les implications potentielles de cette technologie.
Travaillons ensemble pour faire en sorte que l'IA soit utilisée avec responsabilité et bonne foi et que nous puissions tous bénéficier de ses nombreux avantages. L'avenir est prometteur, et tant que les machines ne prendront pas le relais, elles seront sans aucun doute là pour rester.
Adoptons cette technologie et utilisons-la pour créer un monde meilleur et plus sûr. La section suivante aura pour objectif à analyser à quoi ressemble le paysage des menaces à l'ère de l'IA.

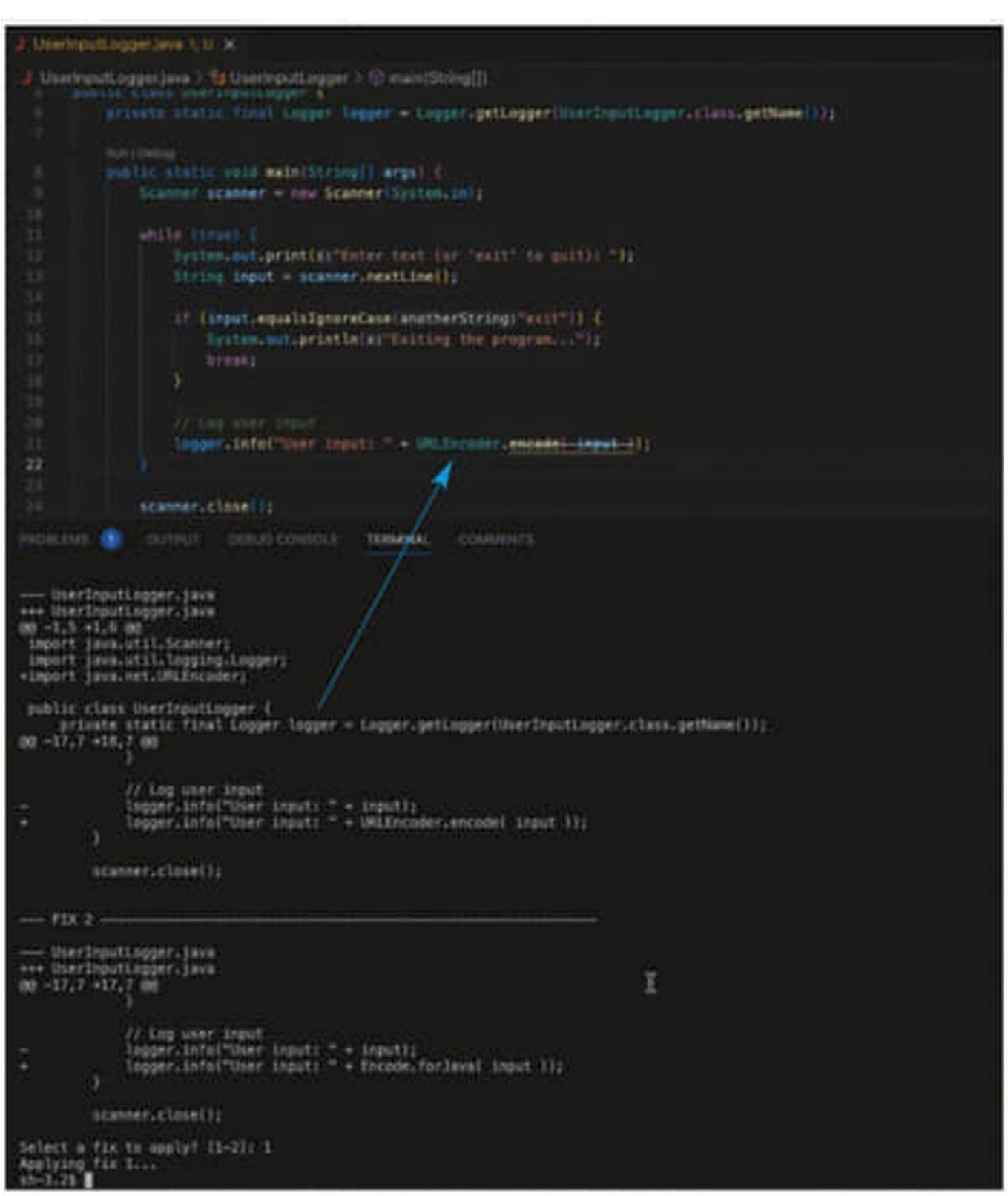


Figure 3 : *Veracode Fix a corrigé la faille CWE-117 en utilisant un URLEncoder pour neutraliser l'entrée fournie par l'utilisateur qui est loggé.*

La sécurité applicative dans une époque où les attaques cyber sont basés sur l'IA

Dans le paysage digital actuel, l'importance de la sécurité des applications ne peut être surestimée, car les entreprises du monde entier sont confrontées à des cybermenaces en constante évolution. Les défenseurs et les attaquants exploitent désormais la puissance de l'intelligence artificielle (IA) à leur avantage. Alors que les attaques basées sur l'IA deviennent de plus en plus sophistiquées, il est crucial pour les organisations d'adopter une approche globale de la sécurité des applications. Dans cette section, nous verrons plus en détail l'importance d'adopter une stratégie de sécurité des applications robuste face aux attaques basées sur l'IA et utiliserons des exemples concrets pour illustrer cela.

L'évolution du paysage des menaces cyber : les attaques alimentées par l'IA

L'IA a transformé de nombreuses industries dans le monde, y compris malheureusement celle de la cybercriminalité. Les pirates utilisent l'IA pour développer des attaques avancées et automatisées capables de contourner les mesures de sécurité traditionnelles. Examinons quelques exemples concrets d'attaques alimentées par l'IA :

1. Les Malwares alimentés par l'IA : les cybercriminels utilisent des algorithmes d'IA pour créer des logiciels malveillants sophistiqués qui s'adaptent et évoluent en réponse à l'évolution des défenses de sécurité. Ces malwares intelligents peuvent contourner les systèmes de détection traditionnels basés sur les signatures, ce qui les rend très difficiles à les identifier et à les fixer efficacement. Par exemple, DeepLocker d'IBM est un type de malware alimenté par l'IA qui dissimule un code malveillant dans des applications apparemment bénignes, ce qui le rend difficile à détecter à l'aide d'un logiciel antivirus conventionnel.
2. Les attaques d'ingénierie sociale : ces attaques alimentées par l'IA sont en augmentation, avec des chatbots et des assistants vocaux imitant les interactions humaines pour tromper les utilisateurs. En analysant de grandes quantités de données, les attaquants peuvent créer des messages hautement personnalisés et convaincants, augmentant considérablement le taux de réussite des tentatives de phishing et des scams grâce à l'ingénierie sociale. Par exemple, les chatbots IA peuvent analyser le profil d'un utilisateur et les conversations précédentes pour créer des messages personnalisés, ce qui les rend plus susceptibles de tromper les utilisateurs.
3. Exploitation automatisée des vulnérabilités : l'IA permet aux pirates d'automatiser l'analyse des vulnérabilités et

d'exploiter les processus de découverte. En tirant parti des algorithmes d'apprentissage automatique, les attaquants peuvent identifier et exploiter les vulnérabilités des applications à une vitesse et à une échelle sans précédent, amplifiant ainsi l'impact potentiel de leurs attaques. Un exemple frappant est le développement de scanners de vulnérabilités pilotés par l'IA comme "AI2" d'OpenAI, qui peut analyser de manière autonome le code et découvrir rapidement les vulnérabilités, dépassant les chercheurs humains.

4. L'IA adverse : l'intelligence artificielle adverse consiste à utiliser des techniques d'apprentissage automatique pour tromper ou manipuler les systèmes d'intelligence artificielle. Les attaquants peuvent créer des entrées malveillantes qui exploitent les vulnérabilités des modèles d'IA, ce qui entraîne des prédictions ou des actions incorrectes. Cela peut avoir de graves conséquences dans divers domaines, notamment la détection des fraudes, la reconnaissance d'images ou le traitement du langage naturel. Par exemple, des chercheurs ont démontré que les systèmes de reconnaissance d'images pouvaient être attaqués en introduisant un bruit imperceptible dans une image, ce qui amène le système d'IA à mal classer des objets tels qu'un panneau d'arrêt ou un panneau de limitation de vitesse.

Une approche globale pour lutter contre les attaques basées sur l'IA

Pour contrer efficacement les attaques pilotées par l'IA, les entreprises doivent compléter leurs pratiques de sécurité des applications par des stratégies spécifiquement conçues pour atténuer ces menaces émergentes. Examinons quelques éléments clés d'une approche globale :

1. Mesures de sécurité renforcées par l'IA : L'adoption de l'IA et des techniques d'apprentissage automatique pour renforcer les défenses de sécurité permettent aux systèmes intelligents d'analyser de grandes quantités de données, de détecter les anomalies et d'identifier les schémas indiquant des attaques pilotées par l'IA. En tirant parti de solutions de sécurité alimentées par l'IA, les organisations peuvent garder une longueur d'avance sur l'évolution du paysage des menaces, comme nous l'avons évoqué précédemment avec Veracode Fix.
2. Systèmes de détection d'intrusion (IDS) basés sur l'IA : la mise en œuvre d'IDS utilisant des algorithmes d'apprentissage automatique permet de détecter les attaques générées par l'IA et d'y répondre. Ces systèmes apprennent le comportement normal du réseau et identifient les activités suspectes, ce qui permet de réagir rapidement en cas d'incident et d'atténuer les dommages potentiels.
3. Partage de l'intelligence sur les menaces : Collaborer avec des pairs de l'industrie, des chercheurs en sécurité et des agences gouvernementales pour partager des renseignements sur les menaces liées aux attaques menées par l'IA permet aux organisations de tirer parti des connaissances partagées et de se tenir au courant des techniques d'attaque et des vulnérabilités émergentes.
4. Développement et test éthiques de l'IA : Encourager des pratiques de développement de l'IA responsables qui donnent la priorité à la sécurité et aux considérations éthiques minimise le risque de vulnérabilités involontaires et d'utilisation abusive de la technologie de l'IA. En intégrant des mesures de sécurité dans le cycle de développement des systèmes d'IA, les organisations peuvent minimiser le risque de vulnérabilités involontaires et d'utilisation abusive de la technologie de l'IA.
5. Formation continue : Il est crucial de rester informé sur les dernières techniques d'attaque pilotées par l'IA et d'investir dans des programmes de formation réguliers pour les employés. En sensibilisant aux risques de sécurité liés à l'IA, les organisations peuvent donner à leur personnel les moyens d'identifier les menaces potentielles et d'y répondre efficacement.

En conclusion

Dans le paysage en constante évolution de la sécurité des applications, la montée en puissance des attaques pilotées par l'IA présente des défis importants. Il est impératif que les organisations adoptent une approche globale qui ne se contente pas de répondre aux préoccupations traditionnelles en matière de sécurité, mais qui prenne également en compte les menaces émergentes posées par l'IA. En mettant en œuvre des mesures de sécurité renforcées par l'IA, en exploitant des systèmes de détection des intrusions basés sur l'IA, en partageant des renseignements sur les menaces, en promouvant des pratiques éthiques de développement de l'IA et en investissant dans l'éducation et la formation continues, les organisations peuvent fortifier leurs défenses de sécurité des applications contre les attaques pilotées par l'IA. Passons maintenant à une analyse plus approfondie des éléments nécessaires à une approche globale de la sécurité et pourquoi.

Pourquoi avez-vous besoin d'une approche globale des tests de sécurité des applications

Avec l'émergence d'attaques alimentées par l'IA et d'autres menaces sophistiquées, il est crucial d'intégrer des tests de sécurité des applications (AST) complets dans le cycle de vie du développement logiciel (SDLC). En tirant parti d'une plate-forme AST efficace qui offre une couverture complète, les organisations peuvent intégrer de manière transparente les tests de sécurité des applications comme une partie inclusive et essentielle du processus de développement. Dans cette section, nous explorerons l'importance d'une plate-forme AST avec une couverture complète pour renforcer la sécurité des applications au sein du SDLC.

Le besoin de tests complets de sécurité des applications (et à quoi cela ressemble)

À mesure que les applications deviennent plus complexes et que les vulnérabilités continuent d'évoluer, des tests complets de sécurité des applications sont essentiels. Les méthodes de test traditionnelles seules sont souvent insuffisantes pour identifier toutes les faiblesses de sécurité potentielles. En adoptant une plate-forme AST avec une couverture complète, les organisations peuvent obtenir les avantages suivants :

- Détection d'un large éventail de vulnérabilités : les plates-formes AST complètes englobent un large éventail de techniques de test de sécurité, notamment l'analyse statique, l'analyse dynamique, les tests interactifs et les tests d'intrusion. Cette approche à multiples facettes permet l'identification de diverses vulnérabilités, telles que les failles au niveau du code, les erreurs de configuration, les problèmes de validation des entrées et les failles logiques.
- Aborder la diversité des menaces : avec la sophistication croissante des attaques, il est essentiel de couvrir un large éventail de menaces potentielles. Une plate-forme AST avec une couverture complète comprend des tests pour les vulnérabilités courantes (par exemple, l'injection SQL, les scripts intersites) ainsi que les menaces émergentes telles que les attaques pilotées par l'IA et d'autres vulnérabilités de type zero-day.
- Intégration dans le SDLC : pour garantir une sécurité efficace des applications, les tests doivent être intégrés de manière transparente dans le SDLC. Une plate-forme AST complète fournit des outils et des intégrations qui permettent d'effectuer des tests de sécurité tout au long du processus de développement, de la création du code au déploiement. Cela permet aux développeurs de résoudre les problèmes de sécurité dès le début, réduisant ainsi le risque d'introduction ou d'omission de vulnérabilités.
- Correction efficace : les plates-formes AST complètes offrent des rapports détaillés et des résultats hiérarchisés qui contribuent à une correction efficace. En fournissant des informations et des conseils exploitables, ces plates-formes permettent aux développeurs de corriger efficacement les vulnérabilités, réduisant ainsi le temps et les efforts nécessaires pour résoudre les problèmes de sécurité.
- Conformité aux normes de sécurité : les organisations opérant dans des secteurs réglementés doivent se conformer à des normes et à des cadres de sécurité spécifiques. Une plate-forme AST avec une couverture complète aide à garantir le respect de ces exigences en facilitant l'identification et l'atténuation des vulnérabilités qui pourraient conduire à la non-conformité.

Le rôle des plates-formes AST dans le SDLC

- Intégration précoce : intégrez les plates-formes AST dès les premières étapes du SDLC pour permettre les tests de sécurité dès que possible. En intégrant la sécurité dans le processus de développement, les entreprises peuvent identifier et traiter de manière proactive les vulnérabilités, réduisant ainsi l'impact potentiel sur l'application finale.
- Automatisation et tests continus : tirez parti des capacités d'automatisation offertes par les plates-formes AST pour permettre des tests continus tout au long du SDLC. L'analyse et les tests automatisés permettent de garantir que la sécurité est évaluée de manière cohérente, même lorsque l'application évolue, réduisant ainsi le risque d'introduction de nouvelles vulnérabilités.
- Collaboration et interfaces ergonomiques pour les développeurs : choisissez les plates-formes AST qui facilitent la collaboration entre les équipes de sécurité et les développeurs. Des interfaces "user friendly" pour les développeurs, des intégrations avec des outils de développement populaires et des conseils de correction clairs contribuent à un flux de travail plus fluide, encourageant les développeurs à s'approprier la sécurité des applications.
- Évolutivité et flexibilité : assurez-vous que la plate-forme AST peut évoluer en fonction des besoins de votre organisation. Il doit s'adapter à la complexité croissante des applications et prendre en charge divers modèles de déploiement, y compris les environnements sur site, cloud et hybrides.

Conclusion

Dans le contexte actuel, des tests complets de sécurité des applications sont essentiels pour protéger les entreprises contre l'évolution des menaces. La bonne plate-forme AST réunira le développement et la sécurité tout en utilisant l'IA. En intégrant une plate-forme AST avec une couverture complète dans le SDLC, les organisations peuvent détecter un large éventail de vulnérabilités, faire face aux menaces émergentes et se concentrer davantage sur l'innovation qui alimente le succès de l'entreprise.

L'ART DE RÉDUIRE LA DETTE DE SÉCURITÉ EN 3 ÉTAPES CLÉS

1 Évaluer et hiérarchiser les risques de sécurité

La première étape pour réduire la dette de sécurité consiste à effectuer une évaluation approfondie des risques de sécurité de votre organisation. Cela implique d'identifier les vulnérabilités, d'évaluer les contrôles de sécurité existants et de comprendre les menaces potentielles. En acquérant une compréhension globale du paysage de sécurité de votre organisation, vous pouvez hiérarchiser efficacement les domaines qui nécessitent une attention immédiate.

2 Mettre en œuvre des contrôles de sécurité robustes

Une fois que vous avez identifié les risques de sécurité critiques, il est temps de mettre en œuvre des contrôles de sécurité robustes pour atténuer ces risques. Cette étape implique l'adoption d'une approche multicouche de la sécurité qui englobe à la fois les solutions techniques et la sensibilisation des employés. La formation de vos employés et développeurs aux meilleures pratiques en matière de cybersécurité est tout aussi essentielle. Organisez des sessions de formation régulières pour sensibiliser aux attaques de phishing, à l'ingénierie sociale et aux habitudes de navigation sécurisées. En favorisant une culture soucieuse de la sécurité, vous donnez à vos employés les moyens d'être la première ligne de défense contre les menaces potentielles.

3 Surveillez et améliorez en continu

La réduction de la dette de sécurité est un processus continu qui nécessite une surveillance et une amélioration continue. La mise en œuvre des bons contrôles de sécurité ne suffit pas ; vous devez également évaluer régulièrement leur efficacité et vous adapter au besoin.

Effectuez régulièrement des audits de sécurité et des tests d'intrusion pour évaluer l'efficacité de vos contrôles de sécurité. Cela vous aidera à identifier les lacunes ou les faiblesses qui doivent être corrigées. Tirez des leçons de tout incident ou violation de sécurité qui se produit et utilisez-les comme des opportunités d'amélioration.

Conclusion

La réduction de la dette de sécurité nécessite une approche stratégique et proactive de la cybersécurité. En suivant les trois étapes clés décrites dans cette section, les organisations peuvent réduire considérablement leur dette de sécurité et améliorer leur posture de sécurité globale.

Devoteam : CyberTrust et SecDevOps

Parmi les entreprises qui se distinguent par leur expertise en matière de transformation numérique et leur agilité, Devoteam émerge comme un pionnier incontesté, mettant en avant ses initiatives innovantes comme son offre SecDevOps.

Laurent Lajugie

Ingénieur cybersécurité, évolue chez Devoteam depuis plus de 7 ans. Ses premières missions portaient sur des sujets offensifs à travers des tests d'intrusion et des audits. Il a ensuite développé, avec son équipe, l'offre Devoteam SecDevOps visant à mettre en place de la sécurité dans l'ensemble du cycle de vie d'une application chez ses clients.

L'expertise en SecDevOps

Dans le paysage numérique actuel, où la vitesse de développement est cruciale, la sécurité ne peut être sacrifiée. C'est là que l'offre SecDevOps, mise en avant par Laurent Lajugie, fait référence à l'intégration de la sécurité dès les premières étapes du cycle de développement logiciel et durant toute sa vie, permettant ainsi de créer des produits et des services robustes et sécurisés dès leur conception. C'est dans ce sens que le terme SecDevOps est plus adapté que DevSecOps pour qualifier cette offre car l'objectif est bien d'appliquer la fonction sécurité au DevOps : Secure(DevOps). Cette fonction s'applique effectivement dans chacune des étapes de la vie d'une application :

- Dès la conception avec du Threat modeling, des abuser stories, l'identification des éléments sensibles…
- Pendant le développement avec de l'analyse du code développé et des différentes sources utilisées, la formation à la mise en place de bonnes pratiques de développement sécurisé…
- Pendant la construction avec des analyses statiques et dynamiques
- Pendant la production avec la sécurité de l'infra et le vulnerability management

Combiné à ces processus et ces formations, il est important de faire évoluer l'outillage mis à disposition des équipes pour que la sécurité soit prise en compte de la manière la plus fluide et efficace possible.

Ci-dessous un schéma représentant des exemples d'implication de mesure de sécurité dans la chaîne CI/CD à mettre en place : **Figure 1**

Notre expertise en SecDevOps repose sur une approche multidisciplinaire. Elle réunit des experts en développement, en sécurité et en exploitation, travaillant de concert pour garantir que la sécurité soit une priorité absolue à chaque étape du processus de développement. Cette approche proactive réduit les risques, accélère la mise sur le marché et garantit la confiance des utilisateurs.

Son objectif final est de transmettre l'expertise aux équipes en question, les processus adaptés et les bons outils pour une autonomie complète dans la démarche SecDevOps sont présentés ci-dessous : **Figure 2**

Le partenariat stratégique gagnant basé sur la complémentarité

Devoteam et Veracode ont créé un partenariat pour redéfinir le paysage de la sécurité numérique sur le marché français, avec des résultats impressionnants à la clé.

Veracode apporte une expertise pointue dans l'analyse de la sécurité des applications, aidant les entreprises à identifier et à résoudre les vulnérabilités avant qu'elles ne soient exploitées par des cybercriminels. Devoteam, quant à elle, possède une profonde compréhension des besoins spécifiques des entreprises françaises et une capacité à intégrer les solutions de cybersécurité dans une vision plus large de la transformation numérique.

L'atout majeur de ce partenariat est de permettre à leurs clients de pouvoir mettre en place une réelle approche SecDevOps au sein de leurs équipes. Concrètement, une fois qu'une roadmap a été définie pour les principales actions, les équipes de Devoteam aident au déploiement des outils SAST et SCA de la plateforme Veracode. Respectivement ces outils permettent d'analyser le code source créé par les développeurs et aussi de s'assurer que les librairies utilisées n'introduisent pas de nouveaux risques dans les applications. Une des forces du SCA Veracode est d'être en mesure de savoir si la fonctionnalité vulnérable de la librairie est bien utilisée dans le code source et aussi d'estimer si la montée de version risque de créer ou pas des problèmes de compatibilité. Tandis que la plateforme Veracode remonte de manière efficace des vulnérabilités et propose des recommandations, les équipes de Devoteam aident à la formation du développement sécurisé et à la priorisation des remédiations à mettre en place. Le vulnerability management est en effet une étape importante pour favoriser l'adoption des équipes à la prise en compte de la sécurité.

Au-delà de la partie outillage, les clients profitent aussi de la proximité des équipes de Devoteam et de Veracode pour être

Figure 1 : *Outillage SecDevOps*

Figure 2 : *Démarche SecDevOps*

ChatGPT

Ce code contient une vulnérabilité SQLi potentielle en raison de la concaténation directe de la chaîne d'entrée utilisateur dans la requête SQL. Pour sécuriser ce code, il est préférable d'utiliser des requêtes paramétrées ou des mécanismes de liaison de variables pour séparer les données de l'instruction SQL, ce qui empêche les attaques SQLi. Voici comment sécuriser le code :

Exemple de code sécurisé :

```python
import sqlite3

# Simule une entrée utilisateur
user_input = "'; DROP TABLE users; --"

# Crée une connexion à la base de données SQLite
conn = sqlite3.connect('ma_base_de_donnees.db')

# Crée un curseur pour exécuter des requêtes SQL
cursor = conn.cursor()

# Utilise une requête paramétrée pour séparer les données de la requête SQL
query = "SELECT * FROM utilisateurs WHERE nom = ?"

# Exécute la requête avec les données sécurisées
cursor.execute(query, (user_input,))

# Récupère les résultats
results = cursor.fetchall()

# Affiche les résultats
```

Figure 3

Bard

La vulnérabilité présente dans le code que vous avez fourni est une injection SQL. Une injection SQL est une attaque dans laquelle un attaquant insère des instructions SQL malveillantes dans une requête SQL. Ces instructions malveillantes peuvent être utilisées pour voler des données, modifier des données ou même exécuter du code arbitraire sur le serveur.

Dans le code que vous avez fourni, la variable user_input est utilisée pour construire la requête SQL. Si l'attaquant contrôle la valeur de user_input, il peut insérer des instructions SQL malveillantes dans la requête. Par exemple, l'attaquant pourrait insérer une instruction DROP TABLE pour supprimer toutes les données de la table utilisateurs.

Pour éviter cette vulnérabilité, vous devez utiliser des paramètres préparés pour construire vos requêtes SQL. Les paramètres préparés permettent de séparer les données de la requête SQL, ce qui empêche l'attaquant d'insérer des instructions SQL malveillantes.

Voici un exemple de code utilisant des paramètres préparés pour éviter l'injection SQL :

Python

```python
import sqlite3

conn = sqlite3.connect('ma_base_de_donnees.db')
cursor = conn.cursor()

query = "SELECT * FROM utilisateurs WHERE nom = ?"
cursor.execute(query, (user_input,))
results = cursor.fetchall()

for row in results:
    print(row)

conn.close()
```

Figure 4

ChatGPT

Quelle est la vulnerabilité ici

```
<?xml version="1.0"?>
<!DOCTYPE chercheur SYSTEM "univdtd">
<chercheur>
<livre>
<titre>Big data</titre>
<auteur>Jean Dupont</auteur>
<ref>Bases de données</ref>
</livre>
<livre>
<titre>Debian 10 administration</titre>
<auteur>Léo Quinville</auteur>
<ref>Tech</ref>
</livre>
<livre>
<titre>Mathématiques financières</titre>
<auteur>Franck Paris</auteur>
<ref>Finance</ref>
</livre>
</chercheur>
```

Dans le code XML que vous avez partagé, il ne semble pas y avoir de vulnérabilité de sécurité particulière. Cependant, il est important de noter que la sécurité des données dépend du contexte dans lequel ces données sont utilisées.

Figure 5

formés sur l'outil et son déploiement et aussi pour faciliter l'accès au support. En effet des points réguliers sont réalisés entre les équipes.

L'IA et son rôle évolutif dans la Sécurité Applicative : Un Avenir Prometteur

À l'heure actuelle, l'IA commence à être utilisée dans la sécurité applicative pour certaines tâches spécifiques. L'une de ses applications les plus courantes réside dans la détection des vulnérabilités et mauvaises pratiques de développement. Elle peut analyser le code source et les configurations des applications pour identifier des vulnérabilités potentielles. Cependant, son efficacité dans ce domaine peut varier en fonction de la complexité des applications et des langages utilisés : c'est là qu'est toute l'importance de bien manipuler cet outil.

Prenons l'exemple ci-dessous avec une injection SQL classique :

```
import sqlite3
conn = sqlite3.connect('ma_base_de_donnees.db')
cursor = conn.cursor()
query = "SELECT * FROM utilisateurs WHERE nom = '" + user_input + "'"
cursor.execute(query)
results = cursor.fetchall
()for row in results:
   print(row)
conn.close()
```

Testons avec Bard et ChatGPT ce qu'ils arrivent à détecter dans ce code : **Figures 3 - 4**

Comme nous pouvons le voir, sur cette vulnérabilité, les outils trouvent bien la bonne vulnérabilité et arrivent à proposer du code source corrigé notamment en indiquant une requête préparée qui permet d'éviter une injection SQL. Très pratique pour un développeur et très utile !

Un autre exemple ci-dessous, basé sur l'OWASP, avec des données simples au format XML

```
<?xml version="1.0"?>
<!DOCTYPE chercheur SYSTEM "univ.dtd">
<chercheur>
 <livre>
  <titre>Big data</titre>
  <auteur>Jean Dupont</auteur>
  <ref>Bases de données</ref>
 </livre>
 <livre>
  <titre>Debian 10 administration</titre>
  <auteur>Léo Quinville</auteur>
  <ref>Tech</ref>
 </livre>
 <livre>
  <titre>Mathématiques financières</titre>
  <auteur>Franck Paris</auteur>
  <ref>Finance</ref>
 </livre>
</chercheur>
```

Analysons les retours de ces deux même IA sur le sujet :

Figure 6

figures 5-6

Nous constatons une divergence forte dans les réponses avec ChatGPT qui estime qu'il n'y a pas de vulnérabilité et Bard qui lui pense qu'il y a une XXE... Qui doit croire le développeur dans ce type de situation ?

Un des éléments de réponse pour expliquer cette divergence est le fait que les IA sont entraînées pour répondre, même si elles n'ont pas d'éléments pertinents à fournir. Nous pouvons supposer ici que Bard, qui a détecté du XML et à qui j'ai demandé quelle était la vulnérabilité, se sent obligée de dire qu'il y a bien une vulnérabilité de type XXE même s'il n'y en a pas.

Nous pouvons bien voir ici que la vigilance de l'expert en sécurité reste primordiale, même en présence de l'intelligence artificielle.

Bien que l'IA puisse jouer un rôle crucial dans la détection des menaces et la prévention des vulnérabilités, elle n'est pas infaillible. Les experts en sécurité doivent continuellement superviser et vérifier les résultats produits par les systèmes d'IA. Cela inclut l'examen des alertes générées par l'IA pour s'assurer qu'elles ne sont pas des faux positifs ou des faux négatifs. De plus, les experts en sécurité sont nécessaires pour interpréter le contexte des menaces détectées, prendre des décisions éclairées et mettre en œuvre des actions correctives appropriées. Que ce soit pour de la détection de vulnérabilités ou pour de la proposition de corrections, il est aujourd'hui important que l'humain reste maître du code produit. C'est d'ailleurs ce que propose Veracode avec le Veracode Fix : proposer des recommandations grâce à une IA qui doivent être validées par un humain avant d'être intégrées dans le code.

Cette collaboration entre l'IA et les experts en sécurité humaine est la clé tant pour garantir une défense robuste contre les menaces informatiques que pour faire grandir l'IA.

CHANEL TECH MISE SUR LES PERFORMANCES ET LA SÉCURITÉ

On ne présente plus la célèbre marque de luxe, Chanel. L'entité IT, CHANEL TECH, a mené un vaste chantier de modernisation pour améliorer l'expérience des clients, tout en misant sur les performances et la sécurité. Dans ce cadre, CHANEL TECH a souhaité atteindre un niveau d'Elite Performer tel qu'il est décrit dans le framework Accelerate.

Nous savions en outre que nous souhaitions utiliser une approche dite de "Security by Design" qui consiste à inclure la sécurité dans toutes les phases du développement logiciel et ainsi résoudre les vulnérabilités "just-in-time". C'est-à-dire au moment où le développeur introduit cette vulnérabilité.

Quand une vulnérabilité est détectée trop tard dans le cycle de développement, le développeur qui a écrit le code peut se demander : qui a écrit ça. Le "context switching" entraîne une perte de productivité.

Une des pratiques consiste à utiliser un outil de SAST (Static Analysis Security Testing) pour scanner le code. Ce scan se réalise directement dans l'IDE sans attendre une validation du pipeline CI/CD.

Celle-ci est indispensable pour avoir des garde-fous sur l'introduction de vulnérabilité et des métriques sur l'ajout de vulnérabilités et la capacité des équipes à corriger ces vulnérabilités et ainsi s'assurer du niveau de qualité du logiciel produit. Pour le SAST, nous avons opté pour la solution Verocode.

La plateforme Veracode nous a permis de déployer des scans de sécurité sur les pipelines de CI/CD et de prendre conscience de notre "dette de sécurité" : aucune équipe ne peut produire du code sans aucune vulnérabilité. Il fallait alors définir une stratégie pour résoudre cette dette technique ou a minima, la contenir.

Pour réduire cette dette, il fallait résoudre les vulnérabilités identifiées dans des librairies tierces puis nous focaliser sur le code produit par nos équipes.

Aujourd'hui, nous monitorons les vulnérabilités le plus tôt possible pour réagir au plus tôt : corriger et valider la correction.

Il existe un réel problème de compétences autour de ces pratiques. Elles sont difficiles à recruter et les profils experts sont rapidement recrutés. Pour relever ce défi, nous déployons des outils de génération de codes tels Github Copilot, Codéium, Amazon Code Whisperer ou Tabnine. Ils sont là pour aider les développeurs et accélérer les développements.

Une des interrogations autour de ces usages est l'entraînement des modèles : uniquement sur les codes internes ou sur une base plus large. C'est un compromis entre la confidentialité et la performance du modèle. Plus ce volume de données est important et plus le modèle LLM sera pertinent.

Il est intéressant de profiter du plus grand volume de données possible afin que ces assistants au développement soient capables d'aider les équipes le plus efficacement possible. Peu de développeurs ont des compétences de sécurité avancée. Il est donc très intéressant de pouvoir les aider avec des LLM dédiés à la sécurité.

Github Copilot, Codéium, Amazon Code Whisperer ou Tabnine devraient pouvoir aider les équipes de développement et assister à la résolution des vulnérabilités identifiées avec l'outil en proposant plusieurs résolutions possibles aux équipes de développement qui feraient alors leur choix.

◆ *Remi LAVEDRINE*

DevSecOps : de la sécurité dans mon DevOps

Adrien Pessu

Je suis Field Security Specialist chez GitHub mais avant tout je suis un développeur passionné avec un intérêt particulier pour la sécurité.

Le DevSecOps est la contraction de Development, Security et Operations. Est-ce une simple tendance ? Une autre manière de parler de DevOps ? ou une nouvelle pratique de création de logiciel ? Pourquoi en est-on venu à une telle popularité du terme ? Tout d'abord revenons à la base, le DevOps.

Cela fait plus de 10 ans que les créateurs de logiciels essaient d'appliquer les principes du DevOps mais pourquoi ?

Avant les développeurs développaient le logiciel dans leur coin. Une fois fini, ils envoient le livrable aux administrateurs systèmes (Operations ou Ops) pour mettre en production et maintenir l'application en condition opérationnelle.

Ce système fonctionnait jusqu'à ce qu'à l'arrivée d'un bug en production, les Ops ne connaissant pas l'application et les devs ne connaissant pas l'hébergement de cette application, il était difficile de résoudre ce bug. Les deux équipes se parlent peu et se renvoient la responsabilité du problème de production.

C'est pour résoudre ce problème qu'est arrivé le DevOps. L'idée est de réunir les Devs et les Ops, pour mettre en production plus rapidement et de déployer les nouvelles versions fréquemment. Pour cela, les Ops mettent en place des outils pour permettre le maintien en condition opérationnelle des applications par les développeurs (et/ou en collaboration avec eux). Ce maintien en condition opérationnelle, qui comprend entre autres, le déploiement, le monitoring, le passage à l'échelle, devient donc la responsabilité des développeurs.

Mais une développeuse ou un développeur, en plus de savoir gérer l'application en production, doit connaitre les langages de programmation frontend, backend, mobile, l'accessibilité, la gestion des bases de données, le CSS, mais aussi la sécurité du logiciel.

Malheureusement, la sécurité passe souvent au second plan. En tant que développeur, la sécurité deviendra un sujet une fois le logiciel terminé, voir même pas du tout un sujet parce qu'on ne se sent pas exposés aux attaques ou alors on verra une fois qu'une faille de sécurité est trouvée en production par un acteur malveillant. Dans ce cas, les conséquences peuvent être graves.

Une équipe de sécurité pourrait alors ajouter des contrôles à chaque étape du cycle de développement : valider l'architecture technique avant de passer au développement, revoir le code avant les tests, faire un test de pénétration avant toutes mises en production. Mais on se retrouve dans notre situation pre-DevOps, avec différentes équipes et responsabilités, une qui essaie de livrer de nouvelles fonctionnalités rapidement avec agilité, les DevOps et la sécurité qui bloquent en ajoutant des barrières et des contrôles.

L'idée du DevSecOps, est la même que celle du DevOps mais pour la sécurité. Il s'agit d'ajouter des outils et pratiques de sécurité le plus automatisé possible au cycle de développement et le plus tôt possible dans celui-ci.

Comme tout bug, le plus tard il est découvert, le plus cher il coûtera. Si une vulnérabilité est découverte en production, il faudra repasser par toutes les étapes du cycle de développement et déploiement du logiciel pour le corriger. À cela, s'ajoute le coût de l'exploitation de cette faille si celle-ci a été découverte par une personne malveillante.

Le *shift left* est une idée qui vise à apporter de la sécurité dans la partie plus à gauche des schémas de cycles de développement — au moment de l'écriture de code.

Le DevSecOps ne devrait pas être une notion à part entière et ses principes devraient déjà être appliqués dans le DevOps. Mais force est de constater que ce n'est pas le cas dans bon nombre d'organisations. Si la popularité du DevSecOps est aussi grande, c'est parce que la sécurité a été oubliée lors du mouvement vers le DevOps.

L'automatisation comme solution

Sans sécurité dans le DevOps, le cycle de développement d'un logiciel se retrouve bloqué dans chacune de ses étapes. Il faut faire valider l'architecture par l'équipe de sécurité avant de commencer à coder. Il faut que l'équipe de sécurité valide le code avant de commencer à tester. Il faut faire un test de pénétration avant d'envoyer en production.

Si on découvre un souci de sécurité à l'une de ces étapes, il faut recommencer depuis le départ. Mais surtout les équipes de sécurité sont bien souvent sous dimensionnées pour la charge de travail. Alors ajouter des étapes bloquantes pour la sécurité lors du cycle de développement, rend l'agilité impossible, ajoute de la défiance envers la sécurité et oblige à trouver des moyens de contournement. Cette organisation pousse donc dos-à-dos, le développement en mode DevOps et l'équipe de sécurité.

Il nous faut donc arrêter d'être bloquant mais plutôt de donner les outils aux développeurs pour leur permettre de développer de manière sécurisée.

La première automatisation à mettre en place est l'analyse des dépendances.

SCA (SOFTWARE COMPOSITION ANALYSIS)

Un SCA est un outil d'analyse de la composition des dépendances de votre application. L'outil lit le fichier décrivant vos dépendances, pom.xml, gradle.build, package.json ou package.lock ,... et construit un arbre de dépendances contenant les dépendances directs et indirectes (les dépendances de ces dépendances). Pour chaque élément de l'arbre, l'outil cherchera s'il existe une vulnérabilité connue et alerter les développeurs.

La recherche de vulnérabilités se basent sur des bases de données de vulnérabilités, la plus connues est la NVD (National Vulnerability Database) et est maintenue par la NIST (National Institute of Standards and Technology) une

agence du département du Commerce des États-Unis. Chaque vulnérabilité identifiée par un numéro de CVE (Common Vulnerabilities and Exposures), sera décrite par son impact, sa sévérité, sa version cible mais surtout avec des conseils de résolution. Il existe d'autres bases de données de vulnérabilités maintenu par des éditeurs comme celle de GitHub (GitHub Advisory Database)

La plus-part du temps, la résolution sera d'utiliser la dernière version de la dépendance vulnérable, où la vulnérabilité a été corrigée. Mais il est aussi possible aussi de paramétrer votre logiciel pour désactiver la partie de la dépendance qui est vulnérable et dont votre logiciel pourrait se passer, avec une variable d'environnement ou l'ajout d'une propriété.

Pourquoi le SCA serait la première automatisation à mettre en place ? Parce que dans un logiciel, les développeurs écrivent entre 10 et 30 % du code qui est mis en production, le reste vient des bibliothèques et des Frameworks.

Il faut donc apporter une attention particulière aux lignes de code que l'on n'écrit pas et que l'on ne connait que très peu. L'exemple récent le plus marquant est certainement la faille log4shell. En décembre 2021, une équipe de chercheur en sécurité d'Alibaba Cloud divulgue une vulnérabilité, dîtes « Zero days » (non connue), sur la bibliothèque log4j. Log4j est un outil très répendu de journalisation pour l'écosystème Java. La faille divulguée concerne toutes les version 2.x et permet d'exécuter du code arbitrairement en utilisant ce qui pouvait paraitre comme une fonctionnalité : la recherche « Java Naming and Directory Interface ». Les organisations qui surveillent leurs dépendances ont pu être alerté dès la sortie de la CVE, ou lors de l'ajout de la dépendance vers log4j, si la version vulnérable est ajoutée par inadvertance, le SCA préviendra l'équipe de développement.

L'utilité de scanner les dépendances lors de la modification du fichier de description de celle-ci est évidente, mais il est primordial d'analyser tout l'arbre des dépendances régulièrement (au moins hebdomadaire), parce que les vulnérabilités apparaissent rarement lors de l'ajout d'une dépendance mais plus souvent plusieurs mois après.

Pour aller un peu plus loin dans la gestion des dépendances, il est intéressant d'avoir dans son système d'information un SBOM (Software supply chain). Que l'on soit dans une entreprise qui développe beaucoup d'applications en interne, dans une entreprise qui intègre beaucoup d'outils open-source ou une entreprise qui achète beaucoup de SaaS (Software as a Service), la gestion de la composition de notre système d'information est complexe.

Reprenons notre exemple de log4shell. La résolution de la vulnérabilité ne s'arrête pas à la mise à jour de la dépendance dans l'application pour les clients. Qu'en est-il de l'application d'intégration continue ou de mesure de la qualité de code ? Ces applications peuvent avoir accès au code source ou à l'infrastructure. C'est pour cela qu'ont émergé des standards d'export de SBOM, comme Software Product Data Exchange (SPDX), pour pouvoir importer automatiquement l'ensemble des composants et repérer rapidement ceux qui sont vulnérables. La sécurité du système d'information est comme une chaine, aussi forte que son maillon le plus faible. Une fois avoir sécurisé le code des composants que l'on n'écrit pas, passons à ceux que l'on écrit.

SAST (STATIC APPLICATION SECURITY TESTING)

Un SAST est un outil d'analyse de code static, c'est-à-dire qu'il a accès en lecture au code source sans l'exécuter. Il peut avoir cet accès depuis votre IDE, souvent par un plug-in. Il peut aussi avoir cet accès dans l'intégration continue, ce qui a l'avantage d'ajouter ce contrôle dans une « Pull request » et favoriser la résolution et la collaboration.

L'équipe de sécurité de peut pas être derrière chaque Pull-request pour revoir chacune des modifications de code. L'équipe de développement ne peut pas attendre non plus l'aval de l'équipe de sécurité pour avancer dans la construction de nouvelles fonctionnalités.

Il existe plusieurs types d'outils d'analyse statique de code. Ici, nous allons nous intéresser qu'aux outils de sécurité. On omettra donc l'analyse de qualité, les linters, les types checkers et les formaters (https://github.com/readme/guides/formatters-linters-compilers)

Dans les outils d'analyse du code orienté sécurité, l'interprétation du code va au-delà de la recherche avec des expressions régulières. Des outils plus performants sont apparus ces dernières années, ils parcourent le code pour l'extraire sous plusieurs formes.

Abstract Syntax Tree (AST)

Dans l'AST, chaque nœud correspond à un élément du code (une instruction, une condition, ...). Le terme abstract ici est important car l'AST ne représente pas la totalité du code mais plus la structure du code.

Voici un exemple avec le code suivant

```
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }
}
```

L'AST devrait être représenté comme ceci : **figure 1**

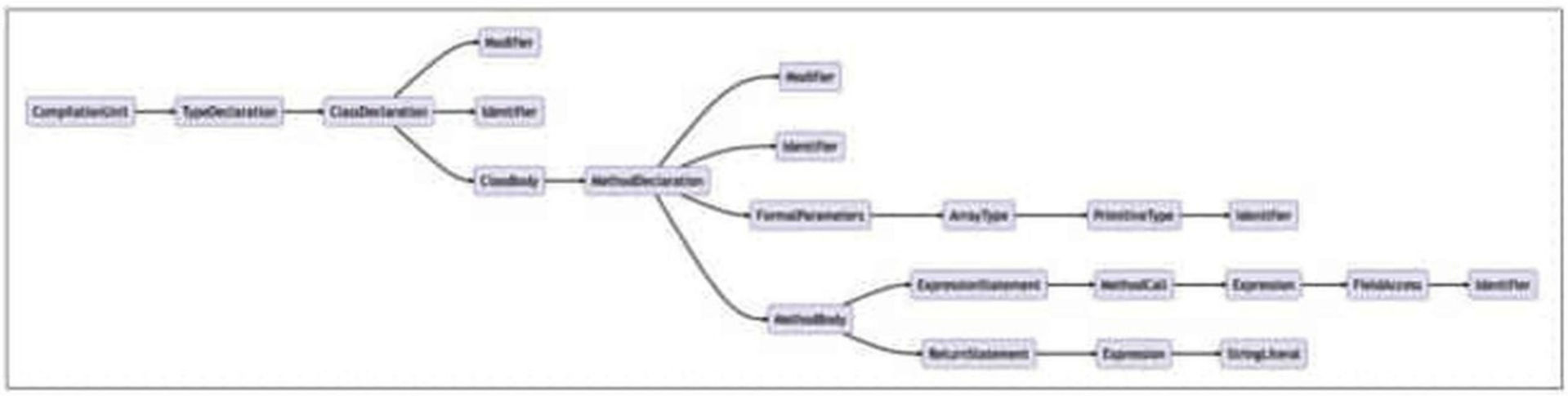


Figure 1

Control-flow graph

Un Control-flow graph est une représentation des chemins d'exécution du code.
Par exemple, pour le code suivant :

```
if (a > 10) {
 if (b > c) {
   c = b;
 } else {
   a = c;
 }
}
System.out.println(a);
System.out.println(b);
System.out.println(c);
```

Figure 2

Data Flow Diagram

Comme son nom l'indique, ce diagramme est une représentation des flux de données. Cette représentation permettra de suivre les données entrées par l'utilisateur pour détecter par exemple des injections SQL ou XSS.
En voici un exemple pour le code :

```
int target = input();

String targetAsString = "Input: " + target

System.out.println(targetAsString)
```

Figure 3

Enfin, à partir de ces représentations, le code sera disséqué avec un analyses dite « Taint » (contamination) pour savoir si une donnée est considérée comme contaminé, c'est-à-dire, si en fonction de sa source (ex : une entrée utilisateur), de son traitement dans le code et sa destination (ex : une requête de base de données), une vulnérabilité peut être exploitée.
Les langages compilés vont avoir une particularité par rapport aux langages interprétés. La raison est simple. Lors de la compilation, il est souvent possible d'apporter des changements aux codes comme l'ajout de méthodes, d'attribues, d'implémentations, ... Donc pour que l'analyse soit la plus pertinente possible, le SAST doit accéder au même code que celui qui sera exécuté en production.
A partir de toutes ces représentations, l'outil d'analyse du code va construire un modèle structuré pour permettre de rechercher des comportements du code qui amèneraient à une vulnérabilité.

Figure 2

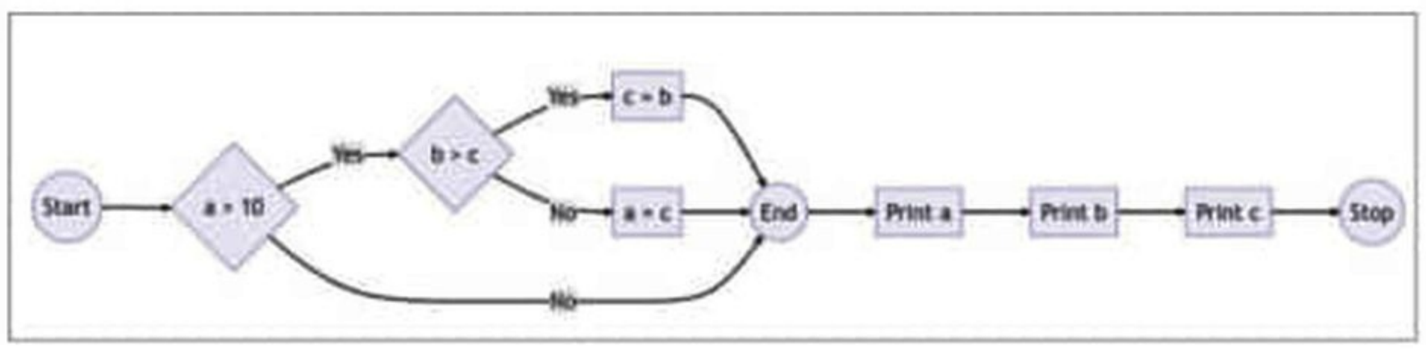
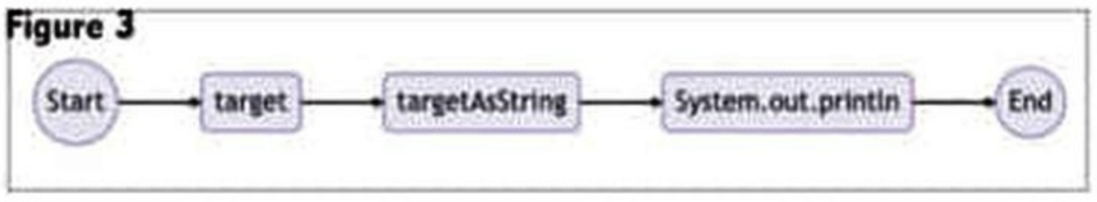


Figure 3



En pratique

Il est utile pour un SAST de pouvoir être configurable. Il est bien sûr primordial d'avoir un certain nombre de règles préconfigurés, mais il aussi important de pouvoir adapter son outil à son utilisation du langage de programmation ou du framework.
Prenons par exemple le code Angular suivant :

```
<div text [innerHTML]="'greeting.translate.key' | translate:{name: name}"></div>
```

Ce code permet de d'afficher le message correspondant à la clé greeting.translate.key dans la langue de l'utilisateur en utilisant le paramètre name. Ce paramètre est très probablement une variable qui vient d'une entrée utilisateur lors de son inscription. L'utilisation de la propriété innerHTML est donc potentiellement un risque car si l'utilisateur a inséré du code HTML, voir JavaScript dans son nom, la variable name sera interprété par la propriété innerHTML. Il y a donc un risque d'injection HTML ou XSS. Ce système a été mis en place pour permettre aux personnes fonctionnelles du projet de modifier la valeur du texte correspondant à chaque clé dans toutes les langues proposées sans connaissance technique. Ce système permet donc de ne pas avoir à solliciter les développeurs pour changer du texte ou mettre un mot en gras. En tant que responsable sécurité, doit-on bloquer ce système car il est susceptible d'introduire une vulnérabilité, sachant qu'interdire ce système diminue l'agilité du reste de l'équipe ? Et bien non, avec l'approche DevSecOps, on va fournir un outil qui pourra détecter si la variable name représente un risque ou non et cet outil va avertir le développeur qui prendra la responsabilité de cette alerte.
Pour améliorer la sécurité d'un volume de code déjà conséquent, toute l'équipe va chercher à améliorer progressivement son projet, parce qu'il est difficilement réalisable de traiter l'intégralité des alertes sur une base de code créée il y a plusieurs années et où il n'y a peu ou pas eu de correction de vulnérabilités. Le nombre d'alertes seraient trop conséquent. L'approche va être de tout d'abord faire en sorte que le code modifié soit sans vulnérabilité, puis de chercher à corriger en priorité les erreurs critiques. Enfin, les erreurs qui se répètent, pourront être traitées par une solution technique commune, une formation des développeurs ou l'ajout de documentations.
Avoir un SAST qui donnera les informations nécessaires pour corriger la vulnérabilité en toute autonomie est primordiale pour favoriser l'autonomie des développeurs
Enfin dans le choix et la configuration de l'outil, pour plus d'adoption et d'efficacité dans la prise en compte des alertes, on va chercher à réduire au maximum le nombre de faux-positifs.

Le cas particulier de l'analyse des secrets

Les repositories Git publiques sont automatiquement scannés par des robots. Si vous laissez des accès AWS dans votre code, vous pouvez vous faire hacker en moins de 20 minutes. Il est donc intéressant même pour un dépôt git même privé d'avoir un outil de surveillance pour éviter à tout prix d'avoir

des secrets au sein du code. Dans ce cas la remédiation est simple, il faut invalider le secret. Le supprimer du repository Git est inutile, ce code a pu être récupéré sur n'importe quel poste de développeur y ayant accès.

DAST (DYNAMIC APPLICATION SECURITY TESTING)

C'est un outil qui scanne votre application Web via l'interface Web. C'est un test en boîte noire car contrairement aux outils de test de sécurité des applications statiques, les DAST n'ont pas accès au code source et détectent donc les vulnérabilités en effectuant effectivement des attaques et en repérant des indices de failles, comme la détection de version dépréciée de dépendances frontend ou de certificat SSL.
De par leur nature, les DAST ne sont pas personnalisés à votre application et les résultats sont donc peu pertinents. Mais ce sont des outils peu coûteux et facile à mettre en place. Ils peuvent ressortir des erreurs grossières qui doivent être corrigés rapidement.

IAST (INTERACTIVE APPLICATION SECURITY TESTING)

Les IAST sont des DAST mais un peu plus avancées, car les outils ont accès au code source et au monitoring de l'application (mémoire, CPU, ...). Les solutions IAST scrutent les applications en déployant des agents et des capteurs en cours d'exécution et en analysant en permanence toutes les interactions d'application initiées par des tests manuels, des tests automatisés ou une combinaison des deux pour identifier les vulnérabilités en temps réel. Les IAST sont, en général, utilisés sur des environnements de tests car leur exécution ajoute de la charge sur l'infrastructure.

WAF (WEB APPLICATION FIREWALL)

Un WAF est une évolution d'un pare-feu. Il analyse les paquets HTTP(S) d'un serveur. Il peut donc être configuré finement en fonction des technologies du serveur à protéger.

RASP (RUN-TIME APPLICATION SECURITY PROTECTION)

Un RASP est une évolution d'un WAF. Un RASP s'exécute dans l'application par l'installation d'un agent. Il va analyser les requêtes (URL, paramètres, en-tetes, ...) à la recherche d'attaques. Le RASP agit sur la sémantique de la requête.

Infrastructure analysis

L'infrastructure-as-code est une description des éléments de l'infrastructure sous forme de code. Cela tombe dans le giron du SAST. On peut cependant y a jouter une petite différence. Avant d'appliquer le code d'infrastructure certains outils proposent d'afficher les différences entre le code et l'infrastructure réelle. Cela permet à l'outil d'alerter, par exemple si l'on ouvre un port 22 sur une machine virtuelle ou n'importe quelle vulnérabilité liée à un problème de configuration.
Il existe aussi des outils à qui on donne un accès en lecture seule à notre infrastructure cloud qui va l'analyser régulière pour nous alerter de configuration qui pourrait amener une faille de sécurité, comme par exemple, est-ce que notre bucket compatible S3 est ouvert au public alors qu'il devrait être privé.

Enfin, l'utilisation de conteneur pose un peu les mêmes soucis de sécurité. Le fichier de description du conteneur doit être vérifier de la même manière que le code. Le conteneur lors de son exécution doit être vérifié comme le peut être une machine virtuelle. Le système de déploiement de conteneur (Kubernetes) doit être vérifié comme on vérifie la configuration de notre infrastructure cloud.

SIEM (Security information and event management)

Un SIEM est un agrégateur d'évènements de sécurité. L'idée est la suivante : les outils précédemment présentés envoient des alertes à leurs utilisateurs directes, les analyseurs de code aux développeurs et les analyseurs de l'infrastructures aux administrateurs systèmes. Mais pour que l'équipe sécurité puisse avoir une vision globale de la posture de sécurité du système d'information, il faut un outil qui va agréger et guider l'équipe sécurité pour concentrer l'aide là où il est le plus nécessaire.

L'organisation comme solution

La documentation

Particulièrement si vous pouvez faire du télétravail, avoir une documentation à jour est très important. La documentation sera la béquille sur laquelle pourront se reposer les développeuses et les développeurs en toute autonomie. Cette documentation doit bien sûr être à jour. En termes de sécurité, il est utile de documenter les bonnes pratiques, l'échelle de classification des données (confidentielle, privée, publique, ...), mais aussi la liste des vulnérabilités que l'on a déjà détecté lors de précédents tests de pénétration. En plus de la transparence, cela permet de ne pas réitérer les erreurs du passé.

La Formation

Lors de l'arrivé de nouveaux collaborateurs dans l'équipe, il est intéressant de les former aux bonnes pratiques de sécurités du projet. Le contenu de la formation n'a pas besoin d'être très conséquent mais au minimum ce qui a été mis dans la documentation et de donner aux formés l'occasion de poser des questions. Un recyclage annuel n'est pas superflu, car on ne voudrait pas que les bonnes pratiques s'oublient avec le temps.

Les security champions

Dans une organisation repartie en équipe par fonctionnalité (feature team), on donne à une développeuse ou un développeur par équipe la casquette de la sécurité. Cette personne ne fera pas partie de l'équipe sécurité mais aura la charge de reposer la question de la sécurité lors de chaque étape du cycle de vie du logiciel (lors de la revue du backlog, lors de décisions techniques, ...).
De plus, une personne de l'équipe de sécurité va réunir tous les security champions régulièrement pour partager les actualités autour de la sécurité et les questions que chaque équipe a pu se poser.

Risk assessment et Thread Modeling

Le risk assessment et le threat modeling sont deux sessions de travail qui produisent un document chacune, idéalement

stockés dans le code (en markdown ou AsciiDoc). Ces cessions de travail se font idéalement juste avant la conception technique.
Le risk assessment permet de d'établir quelles sont les données que l'on manipule et de les classifier. Ensuite, on note le risque associé à la manipulation de ces données. Y-a-t'il un risque pour la réputation du client ou la nôtre, y-a-t 'il un risque financier ? ,... Il existe plusieurs méthodes de risk assessment. Mozilla propose gratuitement sa méthode rapide pour commencer (https://infosec.mozilla.org/guidelines/risk/rapid_risk_assessment.html). Il n'y a pas besoin d'avoir un profil technique, c'est d'ailleurs intéressant que la ou les personnes ayant un profil fonctionnel puissent se rendre compte des questions de sécurité que doivent se poser l'équipe
Le threat modeling est une réponse au risk assessment. On y note les actions techniques que l'on a prévu pour mitiger ces risques. Il existe beaucoup de méthodes, les plus connues sont STRIDE et PASTA.

KPI (Key Performance Indicator)

L'objectif principal du DevSecOps est d'améliorer notre posture de sécurité et pour mesurer cette amélioration, il faut des indicateurs. Pour la sécurité, il peut y en avoir beaucoup de possibilités. Il est cependant intéressant de mesurer le temps moyen de détection et le temps moyen de résolution.
Avoir des indicateurs est aussi utile pour justifier l'achat de logiciel et le temps passer à les mettre en place auprès de la direction.

Conclusion

L'approche DevSecOps nous amène à apporter de la sécurité plus près des développeurs et le plus automatiquement possible mais cette approche vient aussi avec un changement dans la manière de travailler et implique aussi du changement humain.
Cependant, le DevSecOps ne signifie pas enlever l'équipe de sécurité. Cette équipe continuera d'améliorer la posture de sécurité de l'entreprise et la gestion des incidents en production, mais elle apportera aussi sa connaissance aux équipes de développement. L'équipe sécurité étant souvent sous-dimensionné, la gestion en amont et l'automatisation est un atout.
Cette approche est oubli dons l'application du DevOps et existe par nécessiter.

Un peu de Forensic en Python

La mémoire vive contient toutes les données qui sont traitées lors du démarrage de l'ordinateur. Elle agit comme une sorte de disque dur à accès rapide, plus sa capacité est importante plus les performances de la machine sont élevées en termes de rapidité d'exécution des programmes.

Franck Ebel

Enseignant à l'uphf (université polytechnique des hauts de France), lieutenant-colonel réserviste de la gendarmerie nationale en cyberdéfense.

Co-créateur de la licence CDAISI (cyber défense et anti intrusion des systèmes d'information), intervenant en Master Cyber et chez Oteria Cyber school.

Auteur de divers ouvrages chez ENI Edition sur le Python et le Hacking.

Organisateur et créateur d'épreuves pour diverses challenges de Hacking.

Contrairement au disque dur, la mémoire RAM est sollicitée pour chacune des actions réalisées sur le système et contient donc plus de données sensibles telles que des mots de passe saisis, historique, des données non maîtrisées et non inscrites sur le disque dur...

Il existe de nombreux moyens pour dumper la mémoire, ce qui en fait une cible privilégiée. La compromission d'un système par l'altération directe de la mémoire RAM est plus discrète que celle réalisée par l'altération des fichiers du disque dur.

Contrairement à certains disques durs, le contenu de la mémoire RAM n'est pas chiffré et facilite ainsi l'accès au système ou sa manipulation même si le disque dur est chiffré.

La solidité d'une chaîne est conditionnée par celle de son maillon le plus faible. C'est de cette manière que le vol d'un seul ordinateur portable peut entraîner des pertes colossales: des données sensibles sont stockées sur ces PC.

Les experts en sécurité informatiques recommandent le chiffrement des données, donc du disque dur. Dans ce cas, la sécurité des données repose sur le fait que la mémoire vive soit effacée dès qu'elle n'est plus sous tension. La seule façon de récupérer alors la clé de cryptage est de contrôler la machine. Il existe de nombreux logiciels permettant l'extraction de la mémoire RAM lorsqu'une session Windows est démarrée et contrôlée. Ils nécessitent tous les droits Administrateur pour s'exécuter. Leur exécution peut être réalisée à distance via un accès obtenu licite (accès Netbios ou VNC par exemple) ou illicite (via une faille identifiée et exploitée) et ne nécessite pas d'avoir obligatoirement un accès physique à la machine. Ils peuvent tous être placés sur un média amovible afin de réaliser une extraction depuis celui-ci sans qu'aucun outil ou librairie n'ait besoin d'être installé sur le système.

Dans la suite, nous devons avoir des droits d'administration. Nous partons du principe que Windows est compromis.

Dump mémoire

La première chose à effectuer est un dump mémoire, c'est à dire : récupérer le contenu entier de la mémoire avant de l'exploiter grâce au logiciel, en Python, Volatility que nous verrons dans le point suivant.

Comment effectuer un dump mémoire ? Nous allons voir un moyen, il en existe beaucoup et de nombreux logiciels existent. Depuis 2011, la suite « MoonSol 's Windows Memory Toolskit » de Matthieu Suiche propose un exécutable permettant de dumper la mémoire à partir d'une clé usb : Dumpit.exe

Le fonctionnement en est très simple : cliquer sur l'exécutable et suivre les indications. **Figure 1**

Il suffit donc ensuite de lui donner la destination du Dump, sur la clé usb ici et un nom : IUT_270_MODELE-20120115-095924.raw **Figure 2**

Nous avons donc maintenant un Dump mémoire que nous pouvons utiliser pour la suite.

Analyse du Dump mémoire.

Volatility est un framework développé en Python qui permet d'analyser une empreinte de mémoire physique (dump mémoire). Il existe deux méthodologies pour réaliser les analyses forensics, la méthode à chaud (live forensics) et la méthode à froid (dead forensics). Volatility est utilisé dans le cadre de la méthode à chaud.

Il suffit ensuite de le décompresser et d'avoir au préalable installé python. Volatility fonctionne avec des plugins, pour voir ceux-ci, la commande suivante vous renseignera sur ceux-ci et vous fournira d'autres renseignements :

```
Python vol.py -h
```

Figure 3

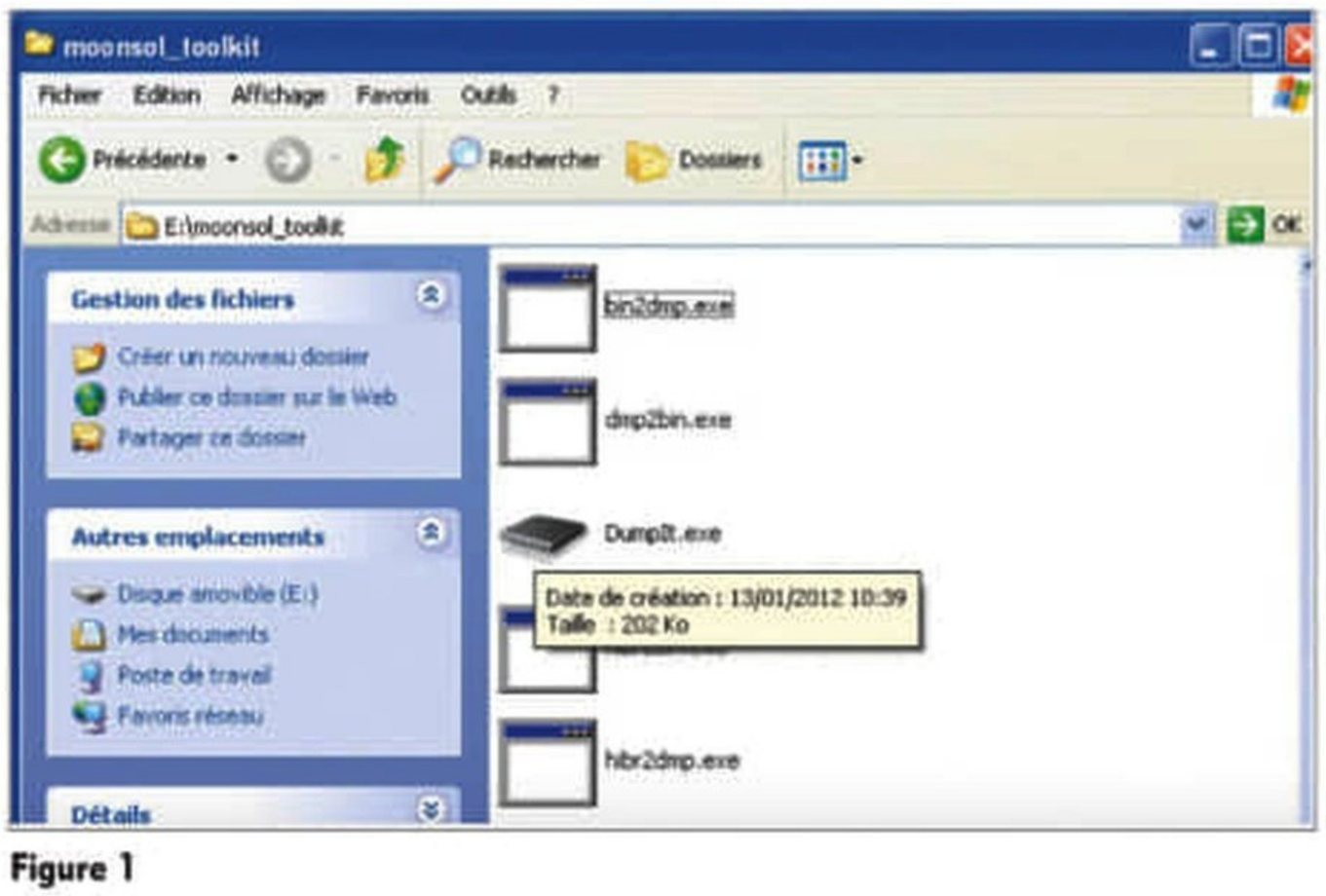


Figure 1

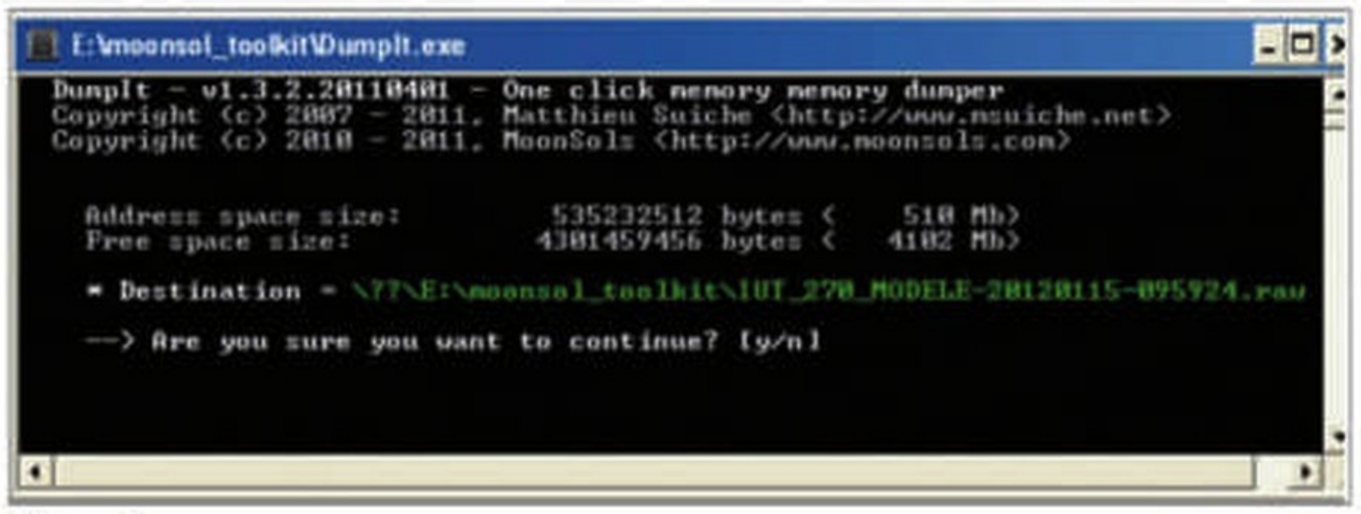


Figure 2

```
/volatility-2.0 $ ./vol.py --info
Volatile Systems Volatility Framework 2.0

PLUGIN_COMMANDS
---------------
bioskbd       - Reads the keyboard buffer from Real Mode memory
connections   - Print list of open connections [Windows XP Only]
connscan      - Scan Physical memory for _TCPT_OBJECT objects (tcp connections)
crashinfo     - Dump crash-dump information
dlldump       - Dump DLLs from a process address space
dlllist       - Print list of loaded dlls for each process
driverscan    - Scan for driver objects _DRIVER_OBJECT
filescan      - Scan Physical memory for _FILE_OBJECT pool allocations
getsids       - Print the SIDs owning each process
handles       - Print list of open handles for each process
hashdump      - Dumps passwords hashes (LM/NTLM) from memory
hibinfo       - Dump hibernation file information
hivedump      - Prints out a hive
hivelist      - Print list of registry hives.
hivescan      - Scan Physical memory for _CMHIVE objects (registry hives)
imagecopy     - Copies a physical address space out as a raw DD image
imageinfo     - Identify information for the image
inspectcache  - Inspect the contents of a cache
kdbgscan      - Search for and dump potential KDBG values
kpcrscan      - Search for and dump potential KPCR values
lsadump       - Dump (decrypted) LSA secrets from the registry
memdump       - Dump the addressable memory for a process
```

Figure 3

Download

Volatility-2.0: tar.gz / zip / standalone EXE / EXE (python installed) / md5 / sha1
Volatility-1.3_Beta: tar.gz zip md5 sha1 gpg tar.gz gpg zip gpg_key
Volatility-1.1.2: tar.gz zip md5 sha1 gpg tar.gz gpg zip gpg_key
Volatility-1.1.1: tar.gz md5 sha1 gpg gpg_key

Figure 4

```
./vol.py -h
Volatile Systems Volatility Framework 2.0
Usage: Volatility - A memory forensics analysis platform.

Options:
  -h, --help            list all available options and their default values.
                        Default values may be set in the configuration file
                        (/etc/volatilityrc)
  --conf-file=/home/rezor/.volatilityrc
                        User based configuration file
  -d, --debug           Debug volatility
  --info                Print information about all registered objects
  --plugins=PLUGINS     Additional plugin directories to use (colon separated)
  --cache-directory=/home/rezor/.cache/volatility
                        Directory where cache files are stored
  --no-cache            Disable caching
  --tz=TZ               Sets the timezone for displaying timestamps
  -f FILENAME, --filename=FILENAME
                        Filename to use when opening an image
  -k KPCR, --kpcr=KPCR  Specify a specific KPCR address
  -g KDBG, --kdbg=KDBG  Specify a specific KDBG virtual address
  --output=text         Output in this format (format support is module
                        specific)
  --output-file=OUTPUT_FILE
                        write output in this file
  -v, --verbose         Verbose information
  --dtb=DTB             DTB Address
  --cache-dtb           Cache virtual to physical mappings
  --use-old-as          Use the legacy address spaces
  -w, --write           Enable write support
  --profile=WinXPSP2x86
                        Name of the profile to load
  -l LOCATION, --location=LOCATION
                        A URN location from which to load an address space
```

Figure 5

```
python vol.py -f ../../dumps-memory/dump_xp_dom1.dmp imageinfo
Volatile Systems Volatility Framework 2.0
Determining profile based on KDBG search...

          Suggested Profile(s) : WinXPSP3x86, WinXPSP2x86 (Instantiated with WinXPSP2x86)
                     AS Layer1 : JKIA32PagedMemory (Kernel AS)
                     AS Layer2 : WindowsCrashDumpSpace32 (/datas/courssecu/failles-physiques/dumpm
emoire/dumps-memory/dump_xp_dom1.dmp)
                     AS Layer3 : FileAddressSpace (/datas/courssecu/failles-physiques/dumpmemoire/
dumps-memory/dump_xp_dom1.dmp)
                      PAE type : No PAE
                           DTB : 0x39000
                          KDBG : 0x8054cde0
                          KPCR : 0xffdff000L
            KUSER_SHARED_DATA : 0xffdf0000L
          Image date and time : 2011-03-30 14:13:49
    Image local date and time : 2011-03-30 14:13:49
        Number of Processors : 1
                  Image Type : Service Pack 3
```

Figure 7

Il est téléchargeable directement sur le site suivant : https://www.volatilesystems.com/default/volatility

La dernière version est la 2.31 et est capable d'analyser des dump mémoire sur xp sp1-2, vista sp0-1-2, seven sp0-1, 2003 sp0-1-2,2008 sp1-2

Figure 4

La commande suivante vous affichera l'aide sur la commande « vol.py » ainsi que les différents plugins : **Figure 5**
Si vous désirez une aide sur le fonctionnement spécifique d'un plugin, il suffit d'appeler l'aide sur celui-ci de la façon suivante :

```
Python vol.py <plugin> -h
```

Figure 6

Vous commencerez par demander les informations inhérentes au dump mémoire concerné.

```
Python -f <chemin du dump> imageinfo
```

Figure 7

Vous savez maintenant que votre dump concerne un windows xp sp3, il faudra préciser ce profil pour chaque plugin utilisé. Pour lister les processus présents en mémoire au moment du dump :

```
Python –profile=WinXPSP3x86 -f <chemin du dump> pslist
```

Figure 8

Le plugin « psscan » fait la même chose avec un peu plus de détails :

```
Python –profile=WinXPSP3x86 -f <chemin du dump> psscan
```

Figure 9

Les hashes du sytème windows en cours peuvent être retrouvés dans le dump mémoire. Pour les afficher, il faut connaître les adresses physiques des emplacements en mémoire de la

```
python vol.py hashdump -h
Volatile Systems Volatility Framework 2.0
Usage: Volatility - A memory forensics analysis platform.

Options:
  -h, --help            list all available options and their default values.
                        Default values may be set in the configuration file
                        (/etc/volatilityrc)
  --conf-file=/home/rezor/.volatilityrc
                        User based configuration file
  -d, --debug           Debug volatility
  --info                Print information about all registered objects
  --plugins=PLUGINS     Additional plugin directories to use (colon separated)
  --cache-directory=/home/rezor/.cache/volatility
                        Directory where cache files are stored
  --no-cache            Disable caching
  --tz=TZ               Sets the timezone for displaying timestamps
  -f FILENAME, --filename=FILENAME
                        Filename to use when opening an image
  -k KPCR, --kpcr=KPCR  Specify a specific KPCR address
  -g KDBG, --kdbg=KDBG  Specify a specific KDBG virtual address
  --output=text         Output in this format (format support is module
                        specific)
  --output-file=OUTPUT_FILE
                        write output in this file
  -v, --verbose         Verbose information
  --dtb=DTB             DTB Address
  --cache-dtb           Cache virtual to physical mappings
  --use-old-as          Use the legacy address spaces
  -w, --write           Enable write support
  --profile=WinXPSP2x86
                        Name of the profile to load
  -l LOCATION, --location=LOCATION
                        A URN location from which to load an address space
  -y SYS_OFFSET, --sys-offset=SYS_OFFSET
                        SYSTEM hive offset (virtual)
  -s SAM_OFFSET, --sam-offset=SAM_OFFSET
                        SAM hive offset (virtual)
```

Figure 6

syskey et de la base SAM. On affiche dans un premier temps les adresses registres présents en mémoire :

```
Python –profile=WinXPSP3x86 -f <chemin du dump> hivelist
```

Figure 10

Il suffit ensuite de demander l'affichege des hashes à l'aide du plugin hashdump en lui précisant l'adresse en mémoire de la syskey et de la base SAM :

```
Python –profile=WinXPSP3x86 -f <chemin du dump> hashdump -y
0xe1035b60 -s 0xe1490b60
```

Figure 11

Si on se place maintenant dans le cadre d'une intrusion sur un système, l'analyse du dump peut permettre de donner des indices sur celle-ci. Le plugin « printkey » va permettre de s'intéresser aux valeurs des clés et sous clés du registre et également aux dernières dates de modification de celles-ci.
Il suffit de lancer la commande suivante :

```
Python -f <chemin dump> printkey
```

Figure 12

On peut également s'intéresser à une clé plus spécifiquement. Après avoir affiché les offset des clés de registre à l'aide du plugin « hivelist » vu précédemment on lance la commande suivante :

```
Python -f <chemin dump> -o <offset clé recherchée> printkey
```

Figure 13

À ce niveau, on s'aperçoit que devant chaque sous clé est écrit un « S » ou un « V » pour « Stable » et « Volatile ».
Les données volatiles (en mémoire) nous intéressant. Nous allons explorer celles-ci à l'aide du commutateur « K ».

```
Python -f <chemin dump> -o <offset clé recherchée> printkey -K
<nom sous clé de registre>
```

Figure 14

Le dump de la clé LSA (service d'autorité locale de sécurité pour Local Security Authority) peut également être intéressant (mot de passe). On utilise alors le plugin « lsadump » en précisant l'offset de la clé « system » et celui de la clé « Security »

```
Python -f <chemin dump> -o <offset clé recherchée> lsadump -y
<offset clé system> -s <offset clé Security>
```

Figure 15

Le plugin « hivedump » permet également de dumper un registre. Le dump sera redirigé dans un fichier « .txt » pour relecture facile car assez long.

```
Python -f <chemin dump> -o <offset clé registre> hivedump
```

Figure 16

On peut également vouloir s'intéresser aux connections ouvertes. Le plugin « connections » permet de lister les connections en cours. Celui-ci est spécifique à XP, pour 7,

```
python vol.py -f ../../dumps-memory/dump_xp_dom1.dmp pslist
Volatile Systems Volatility Framework 2.0
 Offset(V)  Name                    PID    PPID   Thds   Hnds   Time
---------- -------------------- ------ ------ ------ ------ -------------------
0x81fcaa00 System                    4      0     70   2277 1970-01-01 00:00:00
0x81bd13a8 smss.exe                596      4      3     19 2011-03-30 14:02:44
0x81be96b0 csrss.exe               664    596     11    484 2011-03-30 14:02:46
0x81b521c8 winlogon.exe            688    596     19    485 2011-03-30 14:02:46
0x81b68da0 services.exe            732    688     15    281 2011-03-30 14:02:47
0x81b6fda0 lsass.exe               744    688     19    353 2011-03-30 14:02:47
0x81b68568 svchost.exe             920    732     17    201 2011-03-30 14:02:48
0x81b87660 svchost.exe             984    732     11    297 2011-03-30 14:02:48
0x81c34020 svchost.exe            1072    732     63   1216 2011-03-30 14:02:48
0x81718458 SCTSvc.exe             1096    732      9    167 2011-03-30 14:02:48
0x81af8c20 svchost.exe            1304    732      6     89 2011-03-30 14:02:50
0x81b08978 svchost.exe            1356    732     10    155 2011-03-30 14:02:50
0x81b09458 AvastSvc.exe           1416    732     57    744 2011-03-30 14:02:50
0x813d5da0 spoolsv.exe            1744    732      9    108 2011-03-30 14:03:01
0x813dc610 svchost.exe            1864    732      4    107 2011-03-30 14:03:17
0x8163da20 ApplicationUpda        1904    732      3    117 2011-03-30 14:03:17
0x81acdbe0 jqs.exe                2044    732      5    187 2011-03-30 14:03:18
0x812ecda0 ngctw32.exe             232    732      5    116 2011-03-30 14:03:18
0x81638da0 OcsService.exe          268    732      5     69 2011-03-30 14:03:19
0x81358b28 alg.exe                1288    732      6    106 2011-03-30 14:03:24
0x812d75b8 GoogleUpdate.ex        1832    936      4    146 2011-03-30 14:03:30
0x81bfd5e0 explorer.exe            900    472     16    475 2011-03-30 14:06:30
0x81267b38 hkcmd.exe              2288    900      3     93 2011-03-30 14:06:37
0x81385ac0 igfxpers.exe           2372    900      4    101 2011-03-30 14:06:37
0x813cc360 smax4pnp.exe           2284    900      3     90 2011-03-30 14:06:37
0x81e01da0 AdobeARM.exe           2204    900      8    218 2011-03-30 14:06:37
0x812aa5b0 AvastUI.exe            2324    900      8    114 2011-03-30 14:06:37
0x81a8a960 winampa.exe            2252    900      1     31 2011-03-30 14:06:37
0x81e2fda0 jusched.exe            1964    900      5    135 2011-03-30 14:06:38
0x81e0e020 ngtray.exe             2040    900      2     58 2011-03-30 14:06:38
0x81365c68 Bubble.exe             2444    900      4    111 2011-03-30 14:06:38
0x8132b928 ctfmon.exe             2440    900      1     72 2011-03-30 14:06:38
0x8122fbd0 unsecapp.exe           2876    920      3    103 2011-03-30 14:06:44
0x81329638 wuauclt.exe            3368   1072      7    134 2011-03-30 14:11:40
0x81228020 cmd.exe                2708    900      1     34 2011-03-30 14:12:31
0x8138ac18 win32dd.exe            3932   2708      1     23 2011-03-30 14:13:47
```

Figure 8

```
python vol.py -f ../../dumps-memory/dump_xp_dom1.dmp psscan
Volatile Systems Volatility Framework 2.0
 Offset     Name                 PID    PPID   PDB        Time created             Time exited
---------- ---------------- ------ ------ ---------- ------------------------ ------------
----
0x01628020 cmd.exe              2708    900 0x05563000 2011-03-30 14:12:31

0x0162fbd0 unsecapp.exe         2876    920 0x1a3bb000 2011-03-30 14:06:44

0x01667b38 hkcmd.exe            2288    900 0x1a75c000 2011-03-30 14:06:37

0x016aa5b0 AvastUI.exe          2324    900 0x1bfbc000 2011-03-30 14:06:37

0x016d75b8 GoogleUpdate.ex      1832    936 0x1a7af000 2011-03-30 14:03:30

0x016ecda0 ngctw32.exe           232    732 0x16b10000 2011-03-30 14:03:18

0x01729638 wuauclt.exe          3368   1072 0x1c714000 2011-03-30 14:11:40

0x0172b928 ctfmon.exe           2440    900 0x14504000 2011-03-30 14:06:38

0x01758b28 alg.exe              1288    732 0x17f9a000 2011-03-30 14:03:24

0x01765c68 Bubble.exe           2444    900 0x0412a000 2011-03-30 14:06:38

0x01785ac0 igfxpers.exe         2372    900 0x17fa1000 2011-03-30 14:06:37

0x0178ac18 win32dd.exe          3932   2708 0x08c10000 2011-03-30 14:13:47
```

Figure 9

```
python vol.py -f ../../dumps-memory/dump_xp_dom1.dmp hivelist
Volatile Systems Volatility Framework 2.0
Virtual     Physical    Name
0xe1d7db60  0x13988b60  \Device\HarddiskVolume1\Documents and Settings\LocalService\NTUSER.DAT
0xe1c10b60  0x131ceb60  \Device\HarddiskVolume1\Documents and Settings\NetworkService\Local Settin
gs\Application Data\Microsoft\Windows\UsrClass.dat
0xe1bec6a0  0x12d886a0  \Device\HarddiskVolume1\Documents and Settings\NetworkService\NTUSER.DAT
0xe1436b60  0x0f7eab60  \Device\HarddiskVolume1\WINDOWS\system32\config\software
0xe1436bb8  0x0f7ea6b8  \Device\HarddiskVolume1\WINDOWS\system32\config\default
0xe1490b60  0x0f81eb60  \Device\HarddiskVolume1\WINDOWS\system32\config\SAM
0xe17c9758  0x0f2bb758  \Device\HarddiskVolume1\WINDOWS\system32\config\SECURITY
0xe133c288  0x02b90288  [no name]
0xe1035b60  0x02887b60  \Device\HarddiskVolume1\WINDOWS\system32\config\system
0xe1022758  0x0284e758  [no name]
0x8068fd58  0x8068fd58  [no name]
0xe1c93008  0x196a6008  \Device\HarddiskVolume1\Documents and Settings\Administrateur\Local Settin
gs\Application Data\Microsoft\Windows\UsrClass.dat
0xe12e9008  0x0028e008  \Device\HarddiskVolume1\Documents and Settings\Administrateur\NTUSER.DAT
0xe1d62b60  0x19db8b60  \Device\HarddiskVolume1\Documents and Settings\LocalService\Local Settings
\Application Data\Microsoft\Windows\UsrClass.dat
```

Figure 10

```
python vol.py -f ../../dumps-memory/dump_xp_dom1.dmp hashdump -y 0xe1035b60 -s 0xe1490b60
Volatile Systems Volatility Framework 2.0
Administrateur:500:8fefcb557a836e500f91f0644acd94b4:7bed448fe2fbf1a9f1d6cb4bf21b0443:::
Invit●:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:f263a33284ae64116598c48b0d8e68c1:919796b7725438862b6c40a7413941d9:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:5b3852b687894b0bd20ba5a56b765722:::
igi:1003:aad3b435b51404eeaad3b435b51404ee:9cab1178283c1293b149c5e0cf2066c9:::
```

Figure 11

```
python vol.py -f ../../dumps-memory/dump xp dom1.dmp printkey
Volatile Systems Volatility Framework 2.0
Legend: (S) = Stable   (V) = Volatile

----------------------------
Registry: \Device\HarddiskVolume1\Documents and Settings\LocalService\Local Settings\Application
Data\Microsoft\Windows\UsrClass.dat
Key name: S-1-5-19_Classes (S)
Last updated: 2010-05-03 09:57:15

Subkeys:

Values:
----------------------------
Registry: \Device\HarddiskVolume1\Documents and Settings\NetworkService\NTUSER.DAT
Key name: $$$PROTO.HIV (S)
Last updated: 2010-05-03 09:54:57

Subkeys:
  (S) AppEvents
  (S) Console
  (S) Control Panel
  (S) Environment
  (S) Identities
  (S) Keyboard Layout
  (S) Printers
  (S) Software
  (S) UNICODE Program Groups
```

Figure 12

```
python vol.py -f ../../dumps-memory/dump xp dom1.dmp printkey -o 0xe1035b60 -k ControlSet001
Volatile Systems Volatility Framework 2.0
Legend: (S) = Stable   (V) = Volatile

----------------------------
Registry: User Specified
Key name: ControlSet001 (V)
Last updated: 2011-03-30 14:02:32

Subkeys:

Values:
REG_LINK        SymbolicLinkValue : (V) \Registry\Machine\System\ControlSet001
```

Figure 14

```
python vol.py -f ../../dumps-memory/dump xp dom1.dmp -o 0xe1498b60 hivedump >SAM.txt
Volatile Systems Volatility Framework 2.0

cat SAM.txt
Last Written         Key
2010-05-03 11:20:49  \SAM
2010-05-03 11:20:49  \SAM\SAM
2010-05-03 11:20:49  \SAM\SAM\Domains
2010-11-26 08:24:47  \SAM\SAM\Domains\Account
2010-05-03 09:49:48  \SAM\SAM\Domains\Account\Aliases
2010-05-03 09:49:53  \SAM\SAM\Domains\Account\Aliases\000003E9
2010-05-03 09:49:53  \SAM\SAM\Domains\Account\Aliases\Members
2010-05-03 09:49:53  \SAM\SAM\Domains\Account\Aliases\Members\S-1-5-21-1229272821-1364589140-16069
80048
2010-05-03 09:49:53  \SAM\SAM\Domains\Account\Aliases\Members\S-1-5-21-1229272821-1364589140-16069
80048\000003EA
2010-05-03 09:49:48  \SAM\SAM\Domains\Account\Aliases\Names
2010-05-03 09:49:48  \SAM\SAM\Domains\Account\Aliases\Names\HelpServicesGroup
2010-05-03 11:20:49  \SAM\SAM\Domains\Account\Groups
2010-11-26 08:24:48  \SAM\SAM\Domains\Account\Groups\00000201
2010-05-03 11:20:49  \SAM\SAM\Domains\Account\Groups\Names
2010-05-03 11:20:49  \SAM\SAM\Domains\Account\Groups\Names\Aurun
2010-11-26 08:24:48  \SAM\SAM\Domains\Account\Users
```

Figure 16

```
python vol.py -f ../../dumps-memory/dump xp dom1.dmp printkey -o 0xe1035b60
Volatile Systems Volatility Framework 2.0
Legend: (S) = Stable   (V) = Volatile

----------------------------
Registry: User Specified
Key name: $$$PROTO.HIV (S)
Last updated: 2011-03-30 14:02:32

Subkeys:
  (S) ControlSet001
  (S) ControlSet002
  (S) LastKnownGoodRecovery
  (S) MountedDevices
  (S) Select
  (S) Setup
  (S) WPA
  (V) CurrentControlSet

Values:
```

Figure 13

```
python vol.py -f ../../dumps-memory/dump xp dom1.dmp lsadump -y 0xe1035b60 -s 0xe17c9758
Volatile Systems Volatility Framework 2.0
9015040C600011D38

0000   00 00 02 00 00 00 00 00                           ........

L$RTMTIMEBOMB_13201530-8DA3-4e8e-827B-0D888223A588

0000   80 C6 76 8A F3 48 CB 01                           ..v..H..

_SC_Alerter


SAC

0000   8F 00 00 00 02 00 00 00 88 00 00 00 14 4C 87 BA   .............L..
0010   FD A5 06 48 6C C7 2A 7A DA 18 FA D6 16 1C D8 3F   ...Hl.*z.......?
0020   CF 1A 90 46 6F 7A ED 6F 0E F6 3F 8F 31 68 E5 B3   ...Foz.o..?.1h..
0030   1D 19 5F 13 7C 23 01 B4 DF 13 6D 71 51 EB F5 A5   .._.|#....mqQ...
0040   AA DB 16 4F 39 A7 37 AC BD 8D EB A8 E1 C2 4B C8   ...O9.7.......K.
0050   1E 7C 88 A5 EF D8 6D B8 4B 1D 48 6A AD B4 C5 51   .|....m.K.Hj...Q
0060   48 70 92 23 58 28 68 5F 8D 22 26 AD 3A 76 64 EC   Hp.#X(h_."&.:vd.
0070   B4 60 7E 00 24 48 90 B9 41 77 28 65 32 7B 81 4C   .`~.$H..Aw(e2{.L
0080   D3 68 81 F3 CA DE B0 56 D9 CC DD 77 70 6D 72 AA   .h.....V...wpmr.
0090   7A 58 CD BA 50 00 00 00 1B EC 37 C0 C8 6E 93 90   zX..P.....7..n..
00A0   CD 77 A0 AE 9C 83 9E C8 06 41 68 57 AC 38 67 63   .w.......AhW.8gc
00B0   19 09 A5 B1 73 EB D9 52 F9 BF D8 D7 A8 22 AC C6   ....s..R.....".."
00C0   5C 1F 9F 94 37 A6 E4 65 27 93 3C 53 00 1E 2A D2   ....7..e'.<S..*.
00D0   AE 80 A0 C2 B9 1D 07 88 92 19 BA 84 4B E1 C3 FD   ............K...
00E0   6D 47 BC 83 74 E1 6F BC                           mG..t.o.

_SC_RpcLocator


0083343a-f925-4ed7-b1d6-d95d17a0b57b-RemoteDesktopHelpAssistantAccount
```

Figure 15

```
python vol.py -f ../../dumps-memory/dump xp dom1.dmp connections
Volatile Systems Volatility Framework 2.0
 Offset(V)  Local Address             Remote Address            Pid
---------- ------------------------ ------------------------ ------
0x8130a698 127.0.0.1:5152           127.0.0.1:1215             2044
0x81299558 192.168.23.89:1251       92.123.3.235:80            1072
```

Figure 17

```
python vol.py -f ../../dumps-memory/dump xp dom1.dmp connscan
Volatile Systems Volatility Framework 2.0
 Offset     Local Address             Remote Address            Pid
---------- ------------------------ ------------------------ ------
0x01350680 192.168.23.89:1239       193.51.224.14:80           1964
0x01354d68 192.168.23.89:1219       195.221.189.248:3128       1416
0x01355b48 127.0.0.1:1146           127.0.0.1:12080            3488
0x0162d4c0 192.168.23.89:1115       195.221.189.248:3128       1416
0x01654e68 19.0.0.0:8833            8.44.245.129:51287      16777235
0x01665cb0 127.0.0.1:12080          127.0.0.1:1228             1416
0x01666280 127.0.0.1:12080          127.0.0.1:1220             1416
0x01669d38 127.0.0.1:12080          127.0.0.1:1146             1416
0x01699558 192.168.23.89:1251       92.123.3.235:80            1072
0x0169e9b8 192.168.23.89:1249       75.125.246.130:80          1416
0x016cb948 0.0.0.0:0                0.0.0.0:0               2167191904
0x0170a698 127.0.0.1:5152           127.0.0.1:1215             2044
0x01754e68 0.0.0.0:0                232.164.55.129:0        2167754368
0x01758e68 192.168.23.89:1149       195.221.189.248:3128       1416
0x01778688 127.0.0.1:1148           127.0.0.1:12080            3488
0x0179ae68 127.0.0.1:12080          127.0.0.1:1225             1416
0x017c1008 127.0.0.1:12080          127.0.0.1:1221             1416
0x01a1fdc0 192.168.23.89:1156       195.221.189.248:3128       1416
0x01a34950 192.168.23.89:1212       92.123.3.235:80            1072
0x01a36be0 127.0.0.1:1221           127.0.0.1:12080            3488
0x01ac3e68 0.0.0.0:0                0.0.0.0:0               2171354752
0x01b293d0 192.168.23.89:1214       195.221.189.248:3128       1416
```

Figure 18

vista, 2003 et 2008, il faudra utiliser le plugin « netscan ». La commande est la suivante :

```
python vol.py -f <chemin du dump> connections
```

Figure 17

Le plugin « connscan » nous donnera plus d'informations sur les connections. On vérifiera que le pid associé avec la connexion est présent dans le résultat de « pslist », dans le cas contraire cela pourrait signifier la présence d'un rootkit. La commande est la suivante :

```
python vol.py -f <chemin du dump> connscan
```

Figure 18

Le plugin « socket » permettra de s'intéresser aux sockets ouverts afin de vérifier qu'ils ne sont pas suspects.

```
python vol.py -f <chemin du dump> sockets
```

Figure 19

Il peut également être intéressant de connaître les SID des processus en cours pour avoir une idée des utilisateurs. Un SID (identificateur de sécurité unique) est une valeur unique de longueur variable qui est utilisée pour identifier une entité de sécurité ou d'un groupe de sécurité dans les systèmes

Windows. En se référant au « Well-known security identifiers in Windows operating systems », on peut identifier facilement chaque SID révélé

```
python vol.py -f <chemin du dump> getsids
```

Figure 20

Le plugin dlllist permet de lister ou toutes les dll chargées en mémoires ou celles spécifiques à un processus.

```
python vol.py -f <chemin du dump> dlllist -p <PID>
```

Il est également possible, à l'aide du plugin memdump, de dumper l'espace mémoire adressé par un processus pour l'analyser ensuite avec des outils de reverse engeenering. Reprenons l'écran où nous avons utilisé le plugin psscan qui nous donne l'offset mémoire et le pid de chaque processus et lançons la commande suivante :

```
python vol.py -f <chemin du dump> memdump -o <offset processus>
-p <PID>
```

Figure 21

On peut par exemple appliquer la commande strings sur le dump réalisé, ici « 2876.dmp »

Figure 22

On peut également afficher les modules chargés dans le noyau à l'aide du plugin « modules » :

```
python vol.py -f <chemin du dump> modules
```

En regardant de près, on s'aperçoit que deux modules intéressants sont chargés en mémoire. Le premier est ioperm.sys et le deuxième win32dd.sys respectivement de cmospwd et de la suite moonsol. Le plugin memdump permet de dumper le module chargé en noyau pour une analyse ultérieure :

```
python vol.py -f <chemin du dump> moddump -o <adresse base> -D
<répertoire où écrire le dump>
```

Le nom du dump sera de la forme driver.<adresse hexa>.sys

Figure 23

De la même façon qu'un processus peut être caché pour être discret, un module peut être caché. Le plugin modscan va nous aider à afficher un éventuel module caché :

```
python vol.py -f <chemin du dump> modscan > <chemin et nom du
fichier.txt à créer>
```

Le nom du dump sera de la forme driver.<adresse hexa>.sys

Conclusion

Nous venons de voir comment récupérer des informations importantes en mémoire après un Dump. Il faut avoir compromis la machine via des méthodes de hacking différentes telles que les exploits, les failles firewire, un boot avec des outils spécialisés nous permettant d'être administrateur. Ce travail de forensic est utilisé, par exemple, lors de saisies d'ordinateurs pour trouver des preuves contre le criminel. Vous pouvez retrouver mes vidéos sur python, scapy, le forensic en python sur ma chaine youtube :
http://www.youtube.com/c/franckbel

```
python vol.py -f ../../dumps-memory/dump xp dom1.dmp sockets
Volatile Systems Volatility Framework 2.0
 Offset(V)  PID    Port   Proto                Address         Create Time
---------- ------ ------ ------------------ --------------- -------------------
0x817bf418    232   1252     17 UDP              0.0.0.0         2011-03-30 14:13:44
0x812ac008   1072   1251      6 TCP              0.0.0.0         2011-03-30 14:13:42
0x8124f6a8   1356   1900     17 UDP              192.168.23.89   2011-03-30 14:04:02
0x8124f6a8   1416  12143      6 TCP              127.0.0.1       2011-03-30 14:02:53
0x8170f1b8   1072   1250     17 UDP              127.0.0.1       2011-03-30 14:13:42
0x81620a28      4    445      6 TCP              0.0.0.0         2011-03-30 14:02:42
0x81ad1478   1416  12465      6 TCP              127.0.0.1       2011-03-30 14:02:53
0x81ad1478    984    135      6 TCP              0.0.0.0         2011-03-30 14:02:48
0x81b551c8      4    139      6 TCP              192.168.23.89   2011-03-30 14:02:42
0x81e9a6b8    232   1346     17 UDP              0.0.0.0         2011-03-30 14:03:19
0x81b2b458   1416  12993      6 TCP              127.0.0.1       2011-03-30 14:02:54
0x81b2b458      4    137     17 UDP              192.168.23.89   2011-03-30 14:02:42
0x8126c6b0   1072    123     17 UDP              127.0.0.1       2011-03-30 14:03:30
0x81b29958   1416  12563      6 TCP              127.0.0.1       2011-03-30 14:02:54
0x81b2c278   1416  12110      6 TCP              127.0.0.1       2011-03-30 14:02:53
0x8127fd68   1288   1030      6 TCP              127.0.0.1       2011-03-30 14:03:24
0x8127fd68      4    138     17 UDP              192.168.23.89   2011-03-30 14:02:42
0x81bac200   2204   1070     17 UDP              127.0.0.1       2011-03-30 14:06:42
0x81c41470   1416  12025      6 TCP              127.0.0.1       2011-03-30 14:02:53
0x81323268   1356   1900     17 UDP              127.0.0.1       2011-03-30 14:04:02
0x81323268   1072    123     17 UDP              192.168.23.89   2011-03-30 14:03:30
0x8128be98   2044   5152      6 TCP              127.0.0.1       2011-03-30 14:03:19
0x8128be98      4    445     17 UDP              0.0.0.0         2011-03-30 14:02:42
0x81620b98   1416  12080      6 TCP              127.0.0.1       2011-03-30 14:02:53
0x81b4f450   1416  12995      6 TCP              127.0.0.1       2011-03-30 14:02:53
0x81ad1270   1416  12119      6 TCP              127.0.0.1       2011-03-30 14:02:53
```

Figure 19

```
python vol.py -f ../../dumps-memory/dump xp dom1.dmp getsids
Volatile Systems Volatility Framework 2.0
System (4): S-1-5-18 (Local System)
System (4): S-1-5-32-544 (Administrators)
System (4): S-1-1-0 (Everyone)
System (4): S-1-5-11 (Authenticated Users)
smss.exe (596): S-1-5-18 (Local System)
smss.exe (596): S-1-5-32-544 (Administrators)
smss.exe (596): S-1-1-0 (Everyone)
smss.exe (596): S-1-5-11 (Authenticated Users)
csrss.exe (664): S-1-5-18 (Local System)
csrss.exe (664): S-1-5-32-544 (Administrators)
csrss.exe (664): S-1-1-0 (Everyone)
csrss.exe (664): S-1-5-11 (Authenticated Users)
winlogon.exe (688): S-1-5-18 (Local System)
winlogon.exe (688): S-1-5-32-544 (Administrators)
winlogon.exe (688): S-1-1-0 (Everyone)
winlogon.exe (688): S-1-5-11 (Authenticated Users)
services.exe (732): S-1-5-18 (Local System)
services.exe (732): S-1-5-32-544 (Administrators)
services.exe (732): S-1-1-0 (Everyone)
services.exe (732): S-1-5-11 (Authenticated Users)
lsass.exe (744): S-1-5-18 (Local System)
lsass.exe (744): S-1-5-32-544 (Administrators)
lsass.exe (744): S-1-1-0 (Everyone)
```

Figure 20

```
python vol.py -f ../../dumps-memory/dump xp dom1.dmp -o 0x0162fbd0 -p 2876 memdump -D test
Volatile Systems Volatility Framework 2.0
************************************************************************
Writing unsecapp.exe [  2876] to 2876.dmp
```

Figure 21

```
strings test/2876.dmp | more
ys comm
mSpec=C
_HOST_CH
cuments c
inistrateur
LOGONSERVER=\\IUT_270_MODELE
NUMBER_OF_PROCESSORS=1
OS=Windows_NT
Path=C:\Python27\;C:\Python27\Scripts;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\D
ocuments and Settings\prof\Application Data\Python\Scripts
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.py;.pyw
PROCESSOR_ARCHITECTURE=x86
PROCESSOR_IDENTIFIER=x86 Family 15 Model 2 Stepping 9, GenuineIntel
PROCESSOR_LEVEL=15
PROCESSOR_REVISION=0209
```

Figure 22

```
python vol.py -f ../../dumps-memory/dump xp dom1.dmp moddump -o 0x00eeb49000 -u -D test
Volatile Systems Volatility Framework 2.0
Dumping ioperm.sys, Base: eeb49000 output: driver.eeb49000.sys

ls test
2876.dmp  driver.ee5be000.sys  driver.eeb49000.sys
```

Figure 23

L'équipe des Solutions Engineer France de Checkmarx

La BOLA

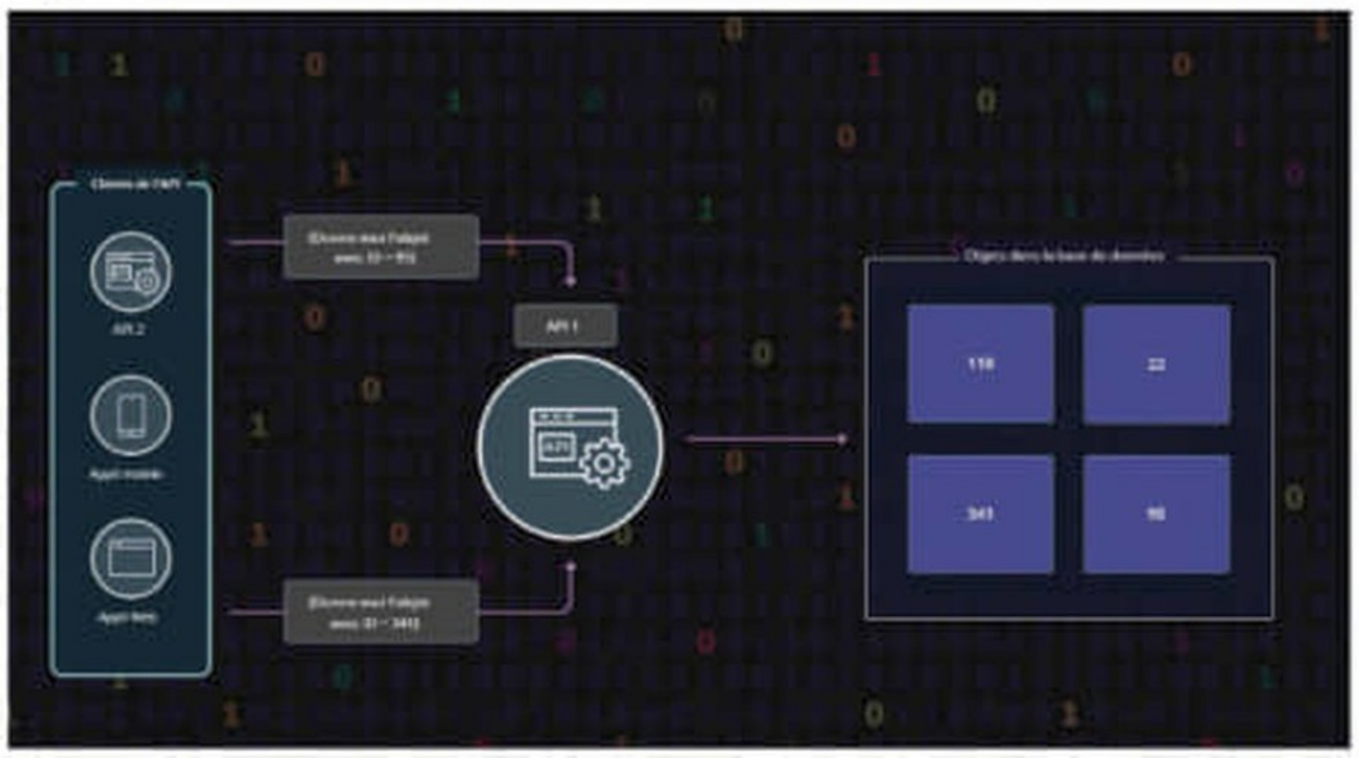
La BOLA (Broken Object Level Authorization, ou Autorisation défaillante au niveau de l'objet) est largement représentée dans le 'top 10' OWASP, étant effectivement la première vulnérabilité à prendre en compte et celle qui est le plus fréquemment exploitée dans les API.

Il est important de suivre plusieurs étapes pour parvenir à la résoudre : **Figure 1**
Nous vous proposons d'approfondir votre compréhension à travers une application Java/Spring.

Description de la vulnérabilité

Habituellement, la vulnérabilité BOLA est la conséquence d'une autorisation mal configurée ou absente.
Les API reçoivent généralement un identifiant d'objet que l'utilisateur de l'application emploie pour ses opérations de lecture, de modification ou de suppression. Lorsque l'application ne parvient pas à vérifier que l'utilisateur authentifié a un niveau d'autorisation suffisant pour accéder à l'objet associé à l'ID spécifié, elle est vulnérable aux attaques BOLA.

Figure 2



Les vulnérabilités BOLA se manifestent également lorsqu'un point de terminaison d'API renvoie une liste d'objets à l'utilisateur, mais que ce dernier n'est autorisé à accéder qu'à une partie de cette liste. Cela signifie qu'un contrôle d'autorisation est non seulement nécessaire pour les opérations sur les objets associés à un ID déterminé par l'utilisateur, mais également pour les listes d'objets exposés par les points de terminaison. **Figure 2**

Le contexte d'exploitation

La société fictive **Bizent** a développé une plate-forme pour héberger des applications d'achat en ligne destinées à des petites et moyennes entreprises. Chaque application d'achat a son propre administrateur et sa propre base d'utilisateurs. Les utilisateurs ne peuvent pas accéder aux méthodes que l'API met à la disposition de l'administrateur. L'interface d'administration de ces applications d'achat en ligne est implémentée de manière que toute requête qui lui est destinée contienne l'UUID de l'appli en question. Dans l'exemple de requête ci-dessous, l'UUID est en surbrillance.

```
1 GET /api/admin/shops/314d8548-77a1-4339-a9b1-56c9366da9e4 HTTP/1.1
2 Host: bizent.me
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:83.0) Gecko/20100101
  Firefox/83.0
4 Accept: */*
5 Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 Connection: close
```

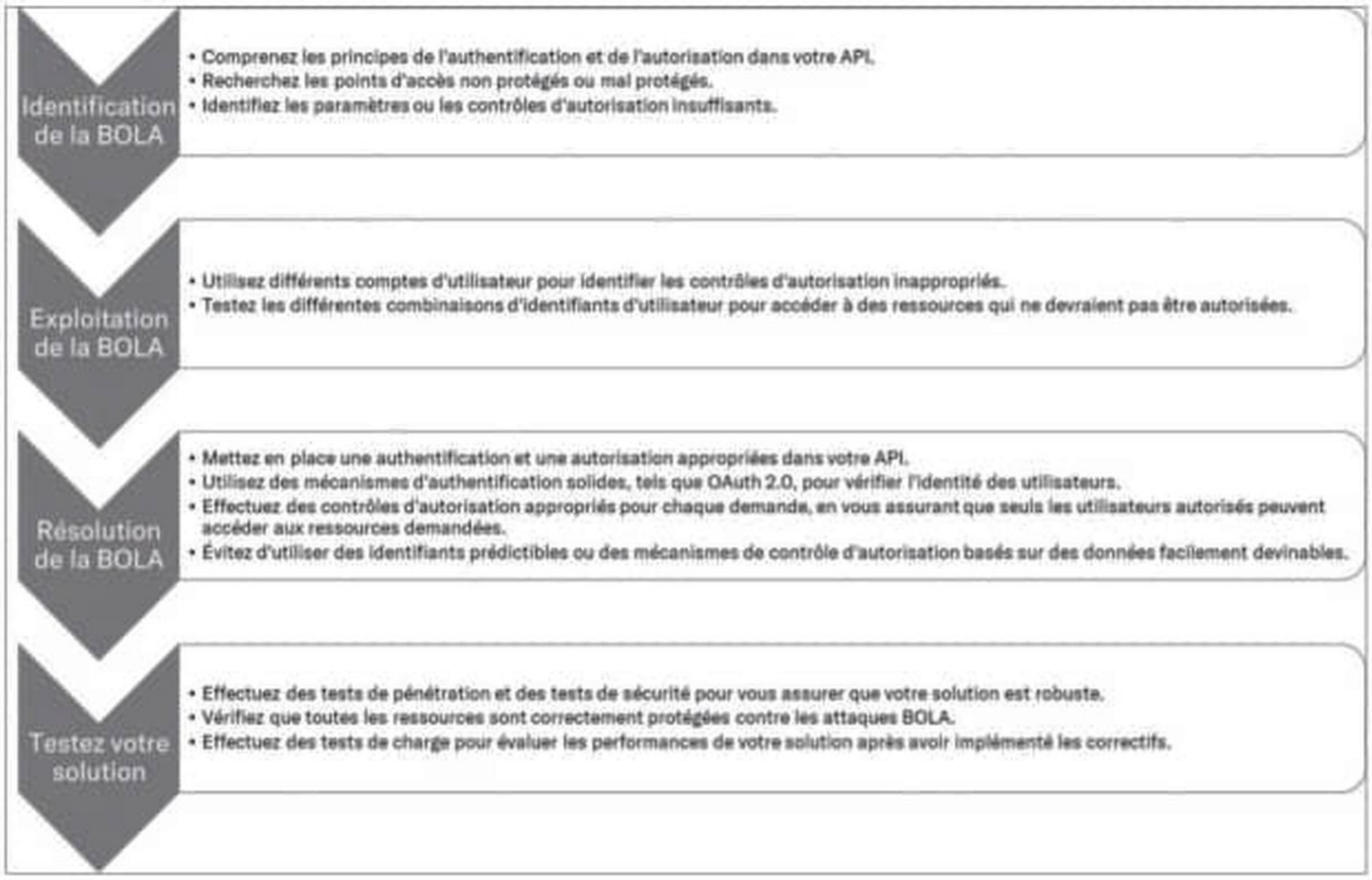


Figure 1

À la recherche d'une vulnérabilité

Bob est un hacker non éthique. En surveillant les programmes bug bounty, il découvre une annonce de la société Bizent invitant les chercheurs en sécurité à tester la dernière version de son API.

Bob décide alors de vérifier si cette version récente contient des vulnérabilités qu'il pourrait exploiter dans son propre intérêt. Il créé un compte sur la plate-forme **BizEnt**, installe l'exemple d'appli d'achat et charge des données de test pour les utilisateurs et les articles. **Figure 3**

Bob ouvre son outil de proxy d'interception et commence à analyser les requêtes et réponses que l'appli échange avec les points de terminaison de l'API de la plate-forme.

Dans une requête envoyée vers le point de terminaison de gestion utilisateur **/api/admin/shops/**, destinée à obtenir la liste des utilisateurs et leurs données personnelles, Bob trouve l'UUID de son appli d'achat : **cd94a00a-dcc3-4683-8279-efb3f61fbf6d**

Une idée lui vient alors à l'esprit : que se passerait-il s'il remplaçait cet UUID par celui d'une autre appli d'achat en ligne ? Il faut immédiatement tester cette idée !

Vulnérabilité détectée

Bob enregistre un autre compte avec la plate-forme **BizEnt** et installe une autre appli d'achat. L'UUID de cette nouvelle boutique est le suivant : **9dacc415-810b-4487-b185-94ced88d60fa**

Dans une réponse, Bob reçoit de sa deuxième appli d'achat du JSON contenant les informations de tous les utilisateurs factices. Cela signifie que le point de terminaison d'API **/api/admin/shops/ est vulnérable aux attaques BOLA. Figure 4**

Code vulnérable

Examinons le code responsable de la fonctionnalité d'autorisation gérée par le point de terminaison **/api/admin/shops/** de l'API de la plate-forme BizEnt.

```
1 @PreAuthorize("hasRole('ADMIN')")
2 @GetMapping("/{id}")
3 public List<User> retrieveAllUsers(@PathVariable(value = "id") UUID id){
4   List<User> allUsers = new ArrayList<>();
5   shopUsersRepository.findByShopId(id.toString()).forEach(allUsers::add);
6   return allUsers;
7 }
```

Ligne 3 : La méthode de contrôleur présentée renvoie une liste d'utilisateurs de la boutique ayant l'UUID spécifié.

Ligne 5 : Le contrôleur requête la base de données **shopUserRepository** pour charger les utilisateurs de la boutique ayant l'UUID spécifié et renvoie la liste ainsi chargée.

Exploitation de la vulnérabilité

La vulnérabilité détectée ouvre à Bob la possibilité de voler les bases utilisateur (contenant, entre autres, les informations personnelles d'identification des utilisateurs) de toutes les applis d'achat en ligne de la plate-forme, en utilisant leurs UUID.

Mais un instant : **comme on ne peut pas deviner les UUID, comment Bob pourrait-il accéder aux UUID des autres boutiques ?**

Il est probable que les ID des boutiques ne sont pas secrètes et qu'il est possible de les trouver quelque part ailleurs dans l'API. Bob a juste à trouver le point de terminaison nécessaire et il commence pour cela à explorer l'appli.

Presque immédiatement, il trouve que la plate-forme **BizEnt** permet aux utilisateurs ordinaires de consulter la liste des applis d'achat en ligne par catégories.

Lorsque **Bob** intercepte la réponse de la plate-forme **BizEnt** contenant le JSON renvoyant la liste des applis d'achat, il remarque qu'il contient également les UUID de ces applis, qui ne sont pas affichés dans l'interface utilisateur.

Bob peut maintenant utiliser ces UUID pour voler les bases utilisateur et leurs données d'identification, et ce pour toutes les applications de la plate-forme. Il vendra pour un bon prix ces informations sur le Darknet !

Code corrigé

Le mécanisme d'autorisation JAVA Spring Security ne fournit pas de contrôles spécifiques contre les attaques BOLA. Dans certains cas, il est possible d'utiliser un contrôle à base d'expression pour gérer l'autorisation, par exemple une annotation **@PreAuthorize** contenant une comparaison entre le nom de l'utilisateur actuellement authentifié avec un nom d'utilisateur fourni dans l'objet entrant, ou une annotation **@PostFilter** qui limite une collection d'objets aux seuls objets auxquels peut accéder l'utilisateur actuel. Mais il ne s'agit pas d'une approche solide, car elle ne couvre pas tous les cas.

La protection contre les attaques BOLA doit être réalisé au niveau du référentiel ou de la BdD, ce qui n'entre pas dans le cadre de Spring Security.

Les instructions qui recherchent les données dans la base de données doivent être conçues en gardant à l'esprit l'idée d'autorisation (qu'il s'agisse d'une méthode personnalisée supplémentaire pour une classe de référentiel, d'une procédure stockée dans la BdD, d'un schéma de BdD séparé pour chaque organisation, ou de tout autre moyen de limiter les données à extraire de la BdD). En d'autres termes, plutôt que de charger tous les objets, il ne faut lire que les objets accessibles à l'utilisateur actuel. L'implémentation exacte de la logique d'autorisation dépend complètement de la logique métier de l'application et doit être réalisée de la façon la plus efficace possible.

Figure 3

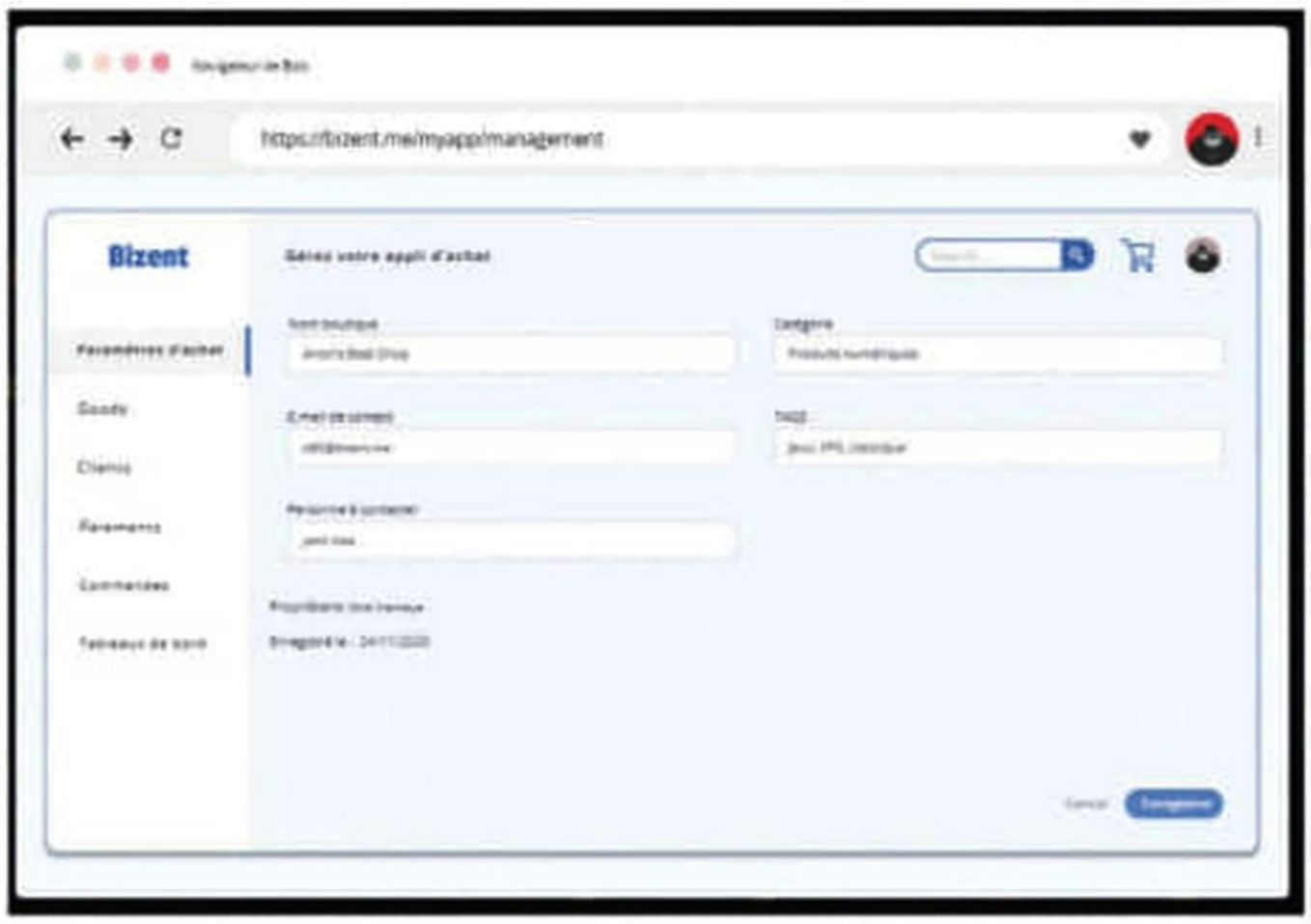


Figure 4

```
1 @PreAuthorize("hasRole('ADMIN')"
2 @GetMapping("/{id}")
3 public ResponseEntity<?> retrieveAllUsers(@PathVariable(value = "id") UUID id){
4    Authentication authentication = SecurityContextHolder.getContext().get
Authentication();
5    String currentPrincipalName = authentication.getName();
6    long currentUserId = usersRepository.findByUsername(currentPrincipalName)
.getId();
7    ArrayList<Long> shopAdmins = shopRepository.getAdminUsersByShopId(id.
toString());
8    if (shopAdmins.contains(currentUserId)){
9       List<User> allUsers = new ArrayList<>();
10      shopUsersRepository.findByShopId(id.toString()).forEach(allUsers::add);
11      return new ResponseEntity<>(allUsers, HttpStatus.OK);
12   }
13   else
14   {
15      return new ResponseEntity<String>("Unauthorized", HttpStatus.UNAUTHORIZED);
16   }
17 }
```

Ligne 8 : Dans cet exemple, on corrige la vulnérabilité au niveau du dépôt en commençant par vérifier que l'utilisateur courant est l'administrateur de la boutique ayant l'UUID fourni. Avant de charger la liste des utilisateurs, on a donc une étape de contrôle supplémentaire consistant à vérifier que l'utilisateur authentifié fait partie des administrateurs de la boutique dont l'ID est défini par l'utilisateur. On charge pour cela la liste des administrateurs depuis la BDD de la boutique et on vérifie que l'utilisateur courant figure dans cette liste.

Ligne 10 : On charge pour cela la liste des administrateurs depuis la BDD de la boutique et on vérifie que l'utilisateur courant figure dans cette liste.

Ligne 15 : Si l'utilisateur ne figure pas dans la liste, une réponse 401 Unauthorized est renvoyée.

Stratégies générales de prévention

Pour prévenir la vulnérabilité BOLA, il est possible d'utiliser les stratégies suivantes :

1. L'API doit explicitement **vérifier que l'utilisateur actuel est autorisé à accéder à l'objet désiré** (le contrôle exact dépend du contexte). En général, les objets doivent avoir un attribut utilisateur définissant qui est autorisé à y accéder, ou l'utilisateur doit avoir les rôles/autorisations nécessaires ou appartenir à un groupe habilité.

```
1 //Dans cet exemple, le code JAVA vérifie que l'objet appartient à l'utilisateur.
2 @GetMapping("/{id}")
3 public ResponseEntity<?> retrieveNoteById(@PathVariable(required = true) long id) {
4    Authentication authentication = SecurityContextHolder.getContext().get
Authentication();
5    String currentPrincipalName = authentication.getName();
6    long loggedUserId = usersRepository.findByUsername(currentPrincipalName)
.getId();
7    Note requestedNote = notesRepository.findById(id).get();
8    if(loggedUserId == requestedNote.getUser().getId()){
9       return new ResponseEntity<>(requestedNote, HttpStatus.OK);
10   }
11   return new ResponseEntity<String>("Unauthorized", HttpStatus.UNAUTHORIZED);
12 }
```

Ligne 8 : Pour commencer, l'utilisateur courant est identifié à partir du contexte de sécurité.

Ligne 9 : Un objet est extrait de la base de données.

Ligne 10 : Avant de transmettre l'objet à l'utilisateur, on vérifie que l'utilisateur actuellement identifié en est bien le propriétaire.

2. Il est possible de prévenir les vulnérabilités BOLA **au niveau de la fonction en contrôlant** que l'utilisateur actuel dispose du rôle nécessaire pour agir sur l'objet désiré. Selon le contexte, le contrôle peut être granulaire : par exemple, un utilisateur a accès à la fonction « lecture », un éditeur a accès aux fonctions « lecture » et « modification » et un administrateur a accès à toutes les fonctions.

```
1 //Dans cet exemple, le code JAVA vérifie que l'utilisateur est admin avant de
poursuivre.
2 @PreAuthorize("hasRole('ADMIN')")
3 @GetMapping()
4 public List<User> retrieveAllUsers(){
5    List<User> allUsers = new ArrayList<>();
6    usersRepository.findAll().forEach(allUsers::add);
7    return allUsers;
8 }
```

3. Pour les identifiants d'objet (comme les UUID), **l'application doit utiliser des chaînes aléatoires**. Des ID aléatoires rendent presque impossible une énumération d'ID d'objet basée sur des suppositions ou sur la force brute. Parallèlement, il ne faut pas les utiliser comme contrôle de sécurité autonome, car **ils n'atténuent pas la vulnérabilité** : ils se limitent à augmenter la complexité de l'attaque. L'utilisation d'ID aléatoires sans une gestion correcte des autorisations apporte une forme de « sécurité par l'obscurité » **qui est défectueuse par essence**. Elle peut être contournée de façon triviale si un autre point de terminaison expose les ID (intentionnellement ou non).
4. Un **schéma de BdD propre à chaque organisation** (ou une BdD séparée, hébergée séparément) peut également empêcher les utilisateurs non autorisés d'accéder à des données appartenant à une autre entreprise.

Cet exercice est extrait de la plateforme de formation Checkmarx Codebashing dédiée à la sécurité applicative et qui en propose une version interactive accessible gratuitement pour les développeurs (accès limité dans le temps) : https://codebashing.com

Azure AD et la sécurité

PARTIE 1

Azure Active Directory (Azure AD) est le service de gestion des identités et des accès basé sur le cloud de Microsoft. Il est de plus en plus utilisé par les entreprises afin de connecter les environnements Active Directory locaux à des services en ligne tels que Office 365, SharePoint, Teams, etc. Cet article présente brièvement Azure AD et explore les nouveaux enjeux qu'offre cet environnement cloud aux équipes de sécurité, sur les plans offensifs comme défensifs.

Arnaud Pilon
CSIRT Manager - Synacktiv

Theo Louis-Tisserand
Consultant Cybersécurité Pentest - Synacktiv

Aymeric Palhiere
Consultant Cybersécurité Reverse- Synacktiv

Azure AD en quelques mots

Azure AD sert de plateforme de gestion des identités pour les applications Microsoft, Azure Resources Manager et essentiellement tout ce que vous y intégrez. **Figure 1**

Malgré le nom trompeur, Azure AD n'est pas *Active Directory* dans le cloud. Cependant, un parallèle entre les deux solutions peut être établi. **Figure 2**

Il existe de nombreuses façons d'interagir avec Azure AD :

- Le portail Azure (https://portal.azure.com).
- Les modules PowerShell : *MSOnline*, *AzureAD* et *Azure CLI*.
- Azure CLI.
- API : *Exchange Provisioning Service*, *Azure AD Graph*, *Microsoft Graph*. Les deux premières sont obsolètes, car Microsoft pousse à unifier toutes les fonctionnalités de ces API au sein de Microsoft Graph.

Les rôles peuvent être différents entre les modules PowerShell ; par exemple, entre les commandes *Get-AzureADDirectory Role* et *Get-MsolRole*.

Azure AD est disponible selon plusieurs formules tarifaires :

- Gratuit.
- Premium P1 (6$ par utilisateur/mois).
- Premium P2 (9$ par utilisateur/mois).

Le plan gratuit est suffisant à des fins de test et offre de nombreuses fonctionnalités telles que la gestion des utilisateurs et des groupes, la synchronisation de l'annuaire local, l'authentification unique sur les applications Azure, etc. Cependant, certaines fonctionnalités avancées d'administration et de sécurité ne sont ouvertes qu'aux licences premium. Elles seront détaillées dans la suite de cet article.

Rôles, autorisations et terminologie

Azure AD introduit de nombreux nouveaux termes, pouvant perturber les utilisateurs réguliers d'*Active Directory*. Le niveau le plus élevé de privilèges est associé au rôle d'administrateur global, qui peut gérer tout ce qui est lié à l'abonnement Azure AD. Le terme d'administrateur de l'entreprise peut parfois être utilisé à la place d'administrateur global, mais ils se réfèrent au même rôle. Il existe également de multiples rôles d'administrateurs limités. **Figure 3**

Le rôle d'administrateur d'application est particulièrement intéressant car, dans Azure AD, tout est une application. La liste de toutes les applications peut être affichée avec la commande PowerShell : *Get-AzureADApplication*.

Par défaut, tout utilisateur Azure AD peut créer de nouvelles applications et les associer à des *Service Principals*. Un Office 365 Azure AD standard possède environ 200 *Service Principals*. Cela peut se révéler très pratique pour des attaquants cherchant à laisser une porte dérobée au sein d'un environnement compromis. En effet, il est possible d'associer un secret d'authentification à un *Service Principal* pour ensuite se connecter avec son identité. *Azure Resource Manager* (plateforme permettant notamment la gestion des machines virtuelles) s'appuyant sur Azure AD pour la gestion des identités, un utilisateur privilégié sur Azure AD peut également accéder à *Azure Resource Manager*.

Intégration avec Active Directory local

Un des aspects les plus intéressants d'Azure AD est sa capacité à se coupler à un environnement *Active Directory* d'entreprise. En effet, de nombreuses organisations utilisent les services de domaine d'*Active Directory* (AD DS) pour l'au-

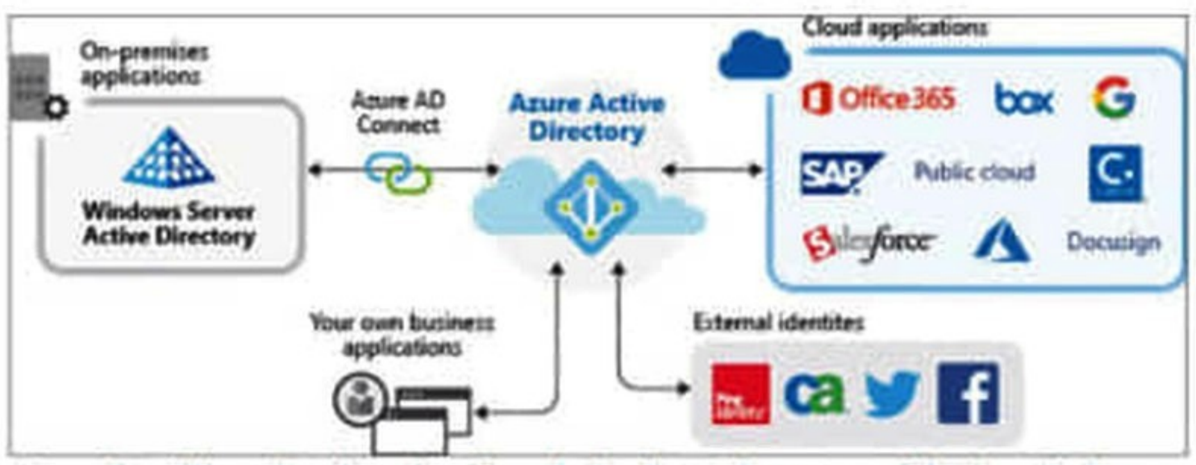


Figure 1

Source: https://docs.microsoft.com/en-gb/azure/active-directory/manage-apps/what-is-application-management

(Windows Server) Active Directory	Azure Active Directory
LDAP	REST API's
NTLM/Kerberos	OAuth/SAML/OpenID/etc
Structured directory (OU tree)	Flat structure
GPO's	No GPO's
Super fine-tuned access controls	Predefined roles
Domain/forest	Tenant
Trusts	Guests

Figure 2 *Source:* https://troopers.de/downloads/troopers19/TROOPERS19_AD_Im_in_your_cloud.pdf

Figure 3

thentification des identités associées aux utilisateurs, aux ordinateurs ou aux applications. Ces services sont fournis par des appareils sur site. Avec de plus en plus d'applications d'entreprise hébergées dans le cloud, il devient beaucoup plus efficace, en termes de latence, d'implémenter directement les services d'identité au sein d'Azure.

Afin de permettre aux utilisateurs d'Active Directory d'utiliser les mêmes informations d'identification dans l'environnement local et dans le cloud, les condensats de mots de passe des utilisateurs doivent être synchronisés. Il existe 3 méthodes de synchronisation disponibles dans Azure AD :

- Password Hash Synchronization (PHS).
- Pass-through Authentication (PTA).
- Active Directory Federation Services (ADFS).

Dans cet article, seule la synchronisation des condensats de mots de passe (PHS) sera étudiée, car c'est l'option la plus couramment utilisée par les entreprises.

Le schéma suivant, issu de la documentation Microsoft, donne un aperçu du flux de travail de PHS : **Figure 4**

Azure AD Connect est l'utilitaire responsable de la synchronisation avec le cloud. Il doit être installé sur un serveur de la forêt *Active Directory*. Les condensats de mots de passe des utilisateurs d'*Active Directory* ne transitent pas directement sur le réseau. À la place, un condensat de chaque condensat de mot de passe est envoyé.

Deux comptes sont automatiquement créés par *Azure AD Connect* :

- *MSOL_deeb213ff4bb* dans Active Directory.
- *Sync_SYNC01_deeb213ff4bb* dans Azure AD. *SYNC01* étant le nom d'hôte du serveur local où Azure AD Connect est installé, et *deeb213ff4bb* étant un identifiant différent pour chaque environnement.

Pour effectuer la synchronisation, ces deux comptes nécessitent des privilèges élevés sur les deux environnements. Dans la seconde partie de cet article, une méthode permettant de compromettre un domaine *Active Directory* configuré avec PHS sera présentée.

Fonctionnalités de sécurité d'Azure AD

Azure AD met automatiquement en œuvre des fonctionnalités de sécurité de base. Par exemple, il dispose d'une politique de verrouillage par défaut bloquant un compte pendant 60 secondes après 10 tentatives d'authentification échouées. Cependant, des fonctionnalités de sécurité plus avancées sont également disponibles en fonction de la licence souscrite.

Stratégies d'accès conditionnel

Les stratégies d'accès conditionnel sont des sortes d'instructions "si-alors" qui interviennent lorsque l'utilisateur tente d'accéder à une ressource. L'accès est déterminé en fonction des signaux envoyés par l'utilisateur. Le diagramme suivant issu de la documentation Microsoft devrait être plus explicite : **Figure 5**

Les stratégies d'accès conditionnel nécessitent au moins une licence Premium P1. Les critères suivants peuvent être utilisés comme signaux :

- Appartenance à un utilisateur ou à un groupe.
- Informations de localisation IP.
- Appareil utilisé.
- Application.
- Détection des risques en temps réel et calculée (partie de la fonctionnalité Azure AD Identity Protection, uniquement pour les licences P2).
- Microsoft Cloud App Security (MCAS).

À titre d'exemple, voici les stratégies couramment appliquées :

- Exiger une authentification multi-facteurs pour les utilisateurs disposant de rôles administratifs.
- Exiger une authentification multi-facteurs pour les tâches de gestion Azure.
- Bloquer les connexions des utilisateurs tentant d'utiliser des protocoles d'authentification obsolètes.
- Exiger des emplacements de confiance pour l'enregistrement de l'authentification multi-facteurs Azure.
- Bloquer ou accorder l'accès à partir d'emplacements spécifiques.
- Bloquer les comportements de connexion risqués.
- Exiger des appareils gérés par l'organisation pour des applications spécifiques.

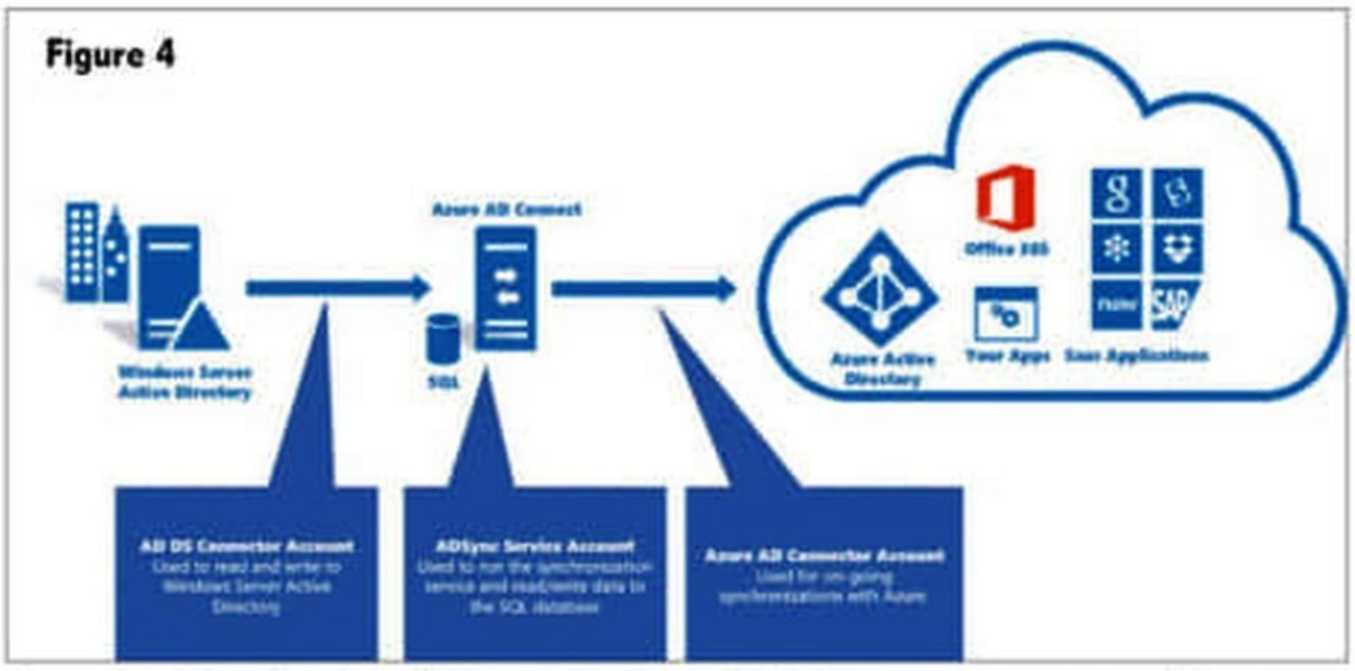
Les stratégies d'accès conditionnel sont accessibles via le portail Azure et ne sont pas visibles pour les utilisateurs non-privilégiés.

Dirk-Jan Mollema a publié un excellent outil nommé *ROADrecon*, qui parvient à les analyser en utilisant une API interne de Microsoft Graph.

Protection de l'identité

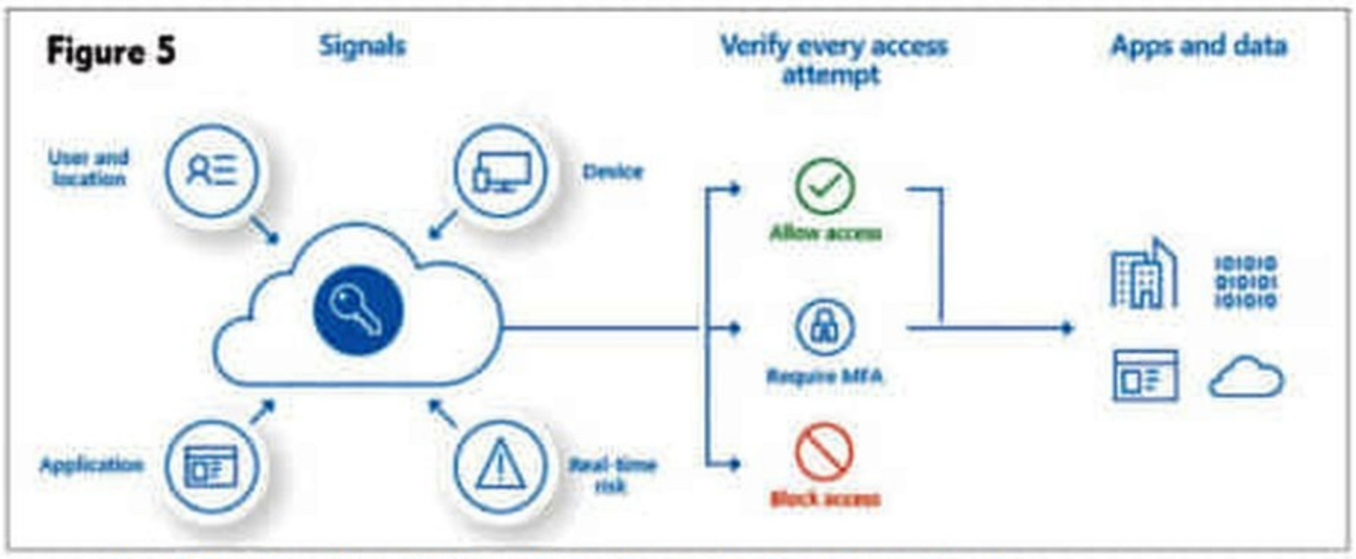
La fonctionnalité *Identity Protection* offre une couche de protection supplémentaire aux détenteurs de licences Premium P2. S'appuyant sur les données acquises grâce à leur position dans les organisations avec Azure AD, Microsoft est en mesure de détecter des comportements utilisateur à risque. Ces

Figure 4



Source : https://docs.microsoft.com/fr-fr/azure/active-directory/hybrid/reference-connect-accounts-permissions

Figure 5



Source : https://docs.microsoft.com/fr-fr/azure/active-directory/conditional-access/overview

utilisateurs peuvent alors être traités différemment par les stratégies d'accès conditionnel.

Par exemple, un utilisateur peut être identifié comme "à risque" s'il utilise un mot de passe présent dans une base de données ayant fuitée. L'ensemble des critères de détection peuvent être trouvés dans la documentation Microsoft.

Security Defaults

Les *Security Defaults* d'Azure AD sont un ensemble de configurations de sécurité visant à protéger contre des attaques courantes telles que le "password spraying", le "replay" et le phishing. En effet, avec les *Security Defaults*, tous les utilisateurs de l'abonnement Azure sont contraints de s'inscrire au service d'authentification multi-facteurs dans un délai de 14 jours. Les membres des groupes d'administration sensibles suivants doivent effectuer une authentification supplémentaire à chaque connexion :

- Administrateur global
- Administrateur SharePoint
- Administrateur Exchange
- Administrateur d'accès conditionnel
- Administrateur de sécurité
- Administrateur de support technique ou administrateur de mots de passe
- Administrateur de facturation
- Administrateur utilisateur
- Administrateur d'authentification

Tous les protocoles d'authentification obsolètes sont bloqués, ce qui signifie que les clients utilisant IMAP, SMTP ou POP3 seront rejetés. Les actions privilégiées, telles que la gestion d'*Azure Resource Manager*, nécessitent une authentification supplémentaire, même si elles sont effectuées via *Azure PowerShell* ou *Azure CLI*. Les *Security Defaults* ne sont pas activés par défaut, ce qui est un peu ironique. Cependant, Microsoft a annoncé dans un article que les abonnements créés après le 22 octobre 2019 pourraient avoir les paramètres de sécurité par défaut déjà activés. Ce n'était pas le cas de l'abonnement créé pour cet article.

Protection des mots de passe pour Active Directory

Azure AD offre la possibilité de définir des politiques de mots de passe qui peuvent être appliquées à l'Active Directory local. Cela est géré par des serveurs appelés *Azure AD Password Protection Proxy Service* et des agents déployés sur les contrôleurs de domaine. **Figure 6**

Ainsi, des listes personnalisées de mots de passe interdits peuvent être définies dans Azure AD et appliquées localement. La fonctionnalité *Identity Protection* évoquée précédemment est également livrée avec une liste prédéfinie de mots de passe interdits.

Journalisation

Le portail Azure AD dispose d'une section de surveillance où les tentatives de connexion et les modifications de configuration sont suivies. Il est possible de transférer ces journaux à *Azure Logs Analytics* pour un traitement ultérieur, sans souscrire une licence premium.

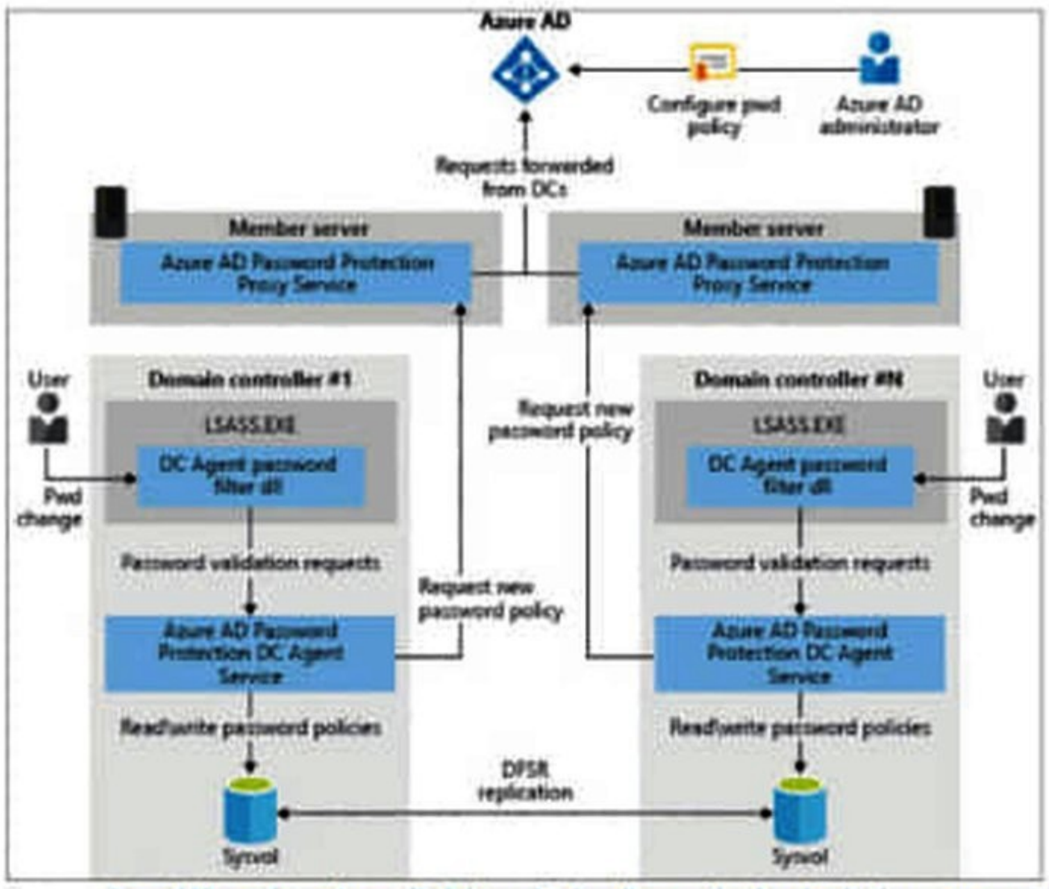


Figure 6

Source : https://docs.microsoft.com/fr-fr/azure/active-directory/authentication/concept-password-ban-bad-on-premises

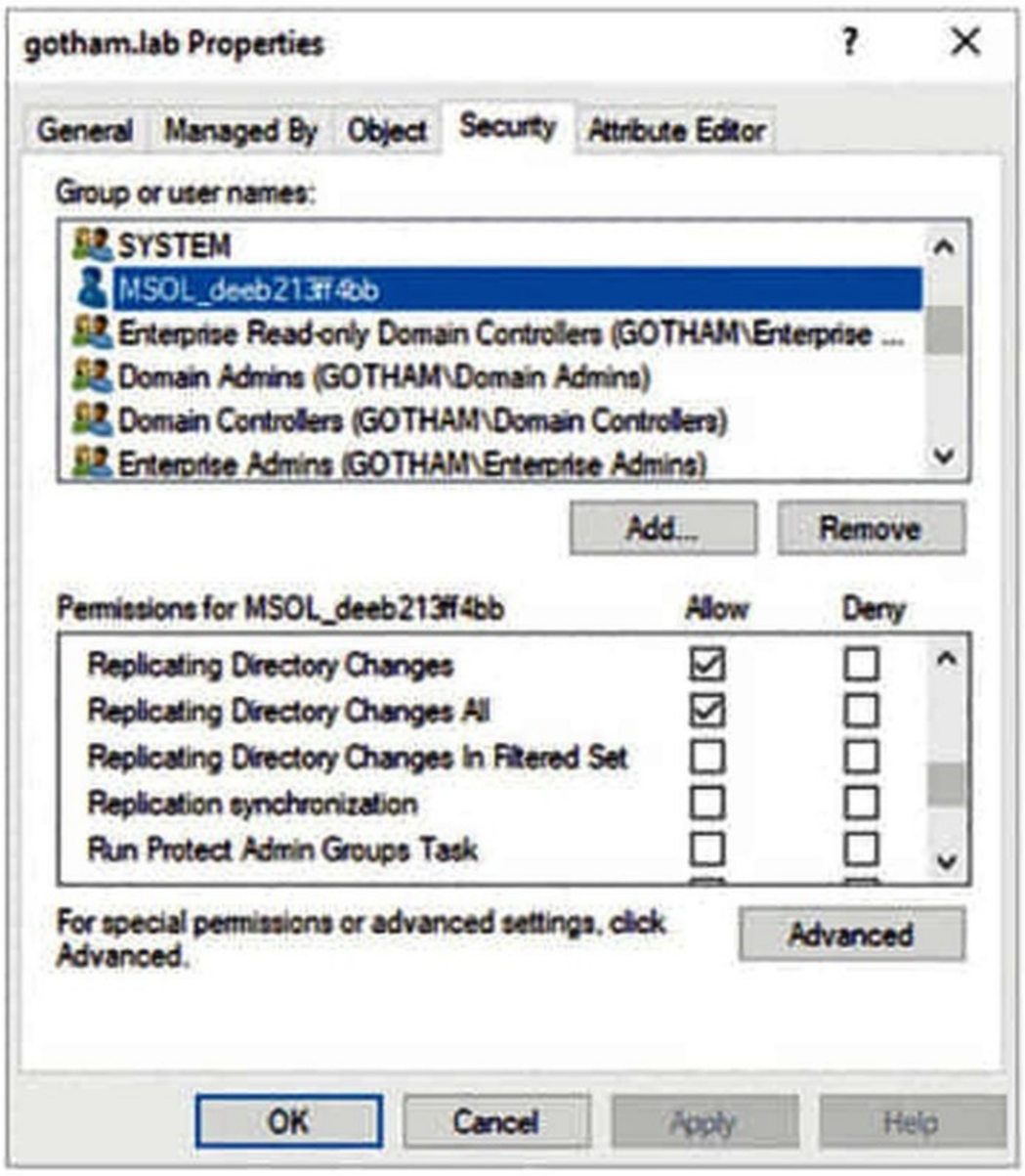


Figure 7

Attaquer Azure AD depuis Active Directory

Comme le lecteur l'aura remarqué, Azure AD est doté de nombreuses fonctionnalités de sécurité. Le reste de cet article adoptera le point de vue de l'attaquant et étudiera ce qui est encore possible et ce qui ne l'est pas lors d'un test de pénétration d'un environnement Azure AD.

Exploiter la synchronisation des condensats de mots de passe (PHS)

Cette attaque ne vise pas directement Azure AD, mais exploite l'une de ses fonctionnalités afin d'élever les privilèges de l'attaquant sur le domaine *Active Directory* local avec lequel il est synchronisé. Comme expliqué précédemment, un compte de synchronisation est créé par *Azure AD Connect* sur l'Active Directory local, lors de la configuration du PHS. Étant donné qu'il est chargé d'envoyer les condensats des mots de passe des utilisateurs vers le cloud, cet utilisateur dispose des droits de réplication sur le domaine : **Figure 7**

Ce compte est capable de répliquer toutes les empreintes de mots de passe des utilisateurs du domaine, ce qui en fait une cible très intéressante pour les attaquants. Voyons maintenant comment nous pourrions récupérer le mot de passe de ce compte. La première étape consiste à trouver le serveur sur lequel *Azure AD Connect* est installé. Microsoft nous facilite la tâche en incluant le nom du serveur de synchronisation et le tenant Azure AD correspondant dans la description LDAP de l'utilisateur MSOL. Il peut être interrogé par n'importe quel utilisateur authentifié du domaine, comme ceci :

```
$ ldapsearch -H ldap://DC01.GOTHAM.LAB:389 -D "GOTHAM\joker" -w "*****" -b "DC
=GOTHAM,DC=LAB" '(description=Azure)' description
[...]
MSOL_deeb213ff4bb, Users, gotham.lab dn:CN=MSOL_deeb213ff4bb,CN=Users,DC
=gotham,DC=lab description: Account created by Microsoft Azure Active Directory
Connect with installation identifier deeb213ff4bb47019f657e127eadecea running on
computer SYNC01 configured to synchronize to tenant gothamlab.onmicrosoft.com.
This account must have directory replication permissions in the local Active Directory
and write permission on certain attributes to enable Hybrid Deployment.
```

```
PS C:\Users\bruce.wayne\Desktop> net localgroup Administrators
Alias name     Administrators
Comment

Members

-------------------------------------------------------------------------------
Administrator
GOTHAM\bruce.wayne
GOTHAM\Domain Admins
The command completed successfully.

PS C:\Users\bruce.wayne\Desktop> .\poc2.ps1
AD Connect Sync Credential Extract v2 (@_xpn_)
        [ Updated to support new cryptokey storage method ]

[*] Querying ADSync localdb (mms_server_configuration)
[*] Querying ADSync localdb (mms_management_agent)
[*] Using xp_cmdshell to run some Powershell as the service user
[*] Credentials incoming...

Domain: GOTHAM.LAB
Username: MSOL_deeb213ff4bb
Password: MyVerySecretMSOLAccountPa$$
```

Figure 8

```
Authentication Id : 0 ; 69683 (00000000:00011033)
Session           : Service from 0
User Name         : ADSync
Domain            : NT SERVICE
Logon Server      : (null)
Logon Time        : 4/20/2020 1:36:49 PM
SID               : S-1-5-80-3245704983-3664226991-764670653-2504430226-901976451
        msv :
         [00000003] Primary
         * Username : SYNC01$
         * Domain   : GOTHAM
         * NTLM     :
         * SHA1     :
        tspkg :
        wdigest :
         * Username : SYNC01$
         * Domain   : GOTHAM
         * Password : (null)
        kerberos :
         * Username : SYNC01$
         * Domain   : gotham.lab
         * Password :

        ssp :
         [00000000]
         * Username : MSOL_deeb213ff4bb
         * Domain   : GOTHAM.LAB
         * Password : MyVerySecretMSOLAccountPa$$
```

Figure 9

```
$ python3 secretsdump.py GOTHAM/MSOL_deeb213ff4bb@DC01.gotham.lab
Impacket v0.9.22.dev1+20200416.91838.62162e0a - Copyright 2020 SecureAuth Corporation

Password:
[-] RemoteOperations failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b5140
Guest:501:aad3b435b51404eeaad3b
krbtgt:502:aad3b435b51404eeaad3
gotham.lab\bruce.wayne:1104:aad
gotham.lab\james.gordon:1105:aa
gotham.lab\alfred.pennyworth:11
gotham.lab\joker:1107:aad3b435b
gotham.lab\penguin:1108:aad3b43
MSOL_deeb213ff4bb:1111:aad3b435
DC01$:1000:aad3b435b51404eeaad3
WEB01$:1103:aad3b435b51404eeaad
SYNC01$:1110:aad3b435b51404eeaa
AZUREADSSOACC$:1112:aad3b435b51
WKS01$:1113:aad3b435b51404eeaad
```

Figure 10

Une fois le serveur identifié, nous aurons besoin d'un compte administrateur local ou d'un compte de service *ADSync* pour interagir avec la base de données *Azure AD Connect*. En effet, cette base de données stocke une version chiffrée du mot de passe du compte MSOL, qui peut être déchiffrée à l'aide de *C:\Program Files\Microsoft Azure AD Sync\Binn\mcrypt.dll* et des clés DPAPI du compte *NT SERVICE\ADSync*.

NT SERVICE\ADSync est un compte virtuel et, par conséquent, il n'est pas nécessaire de gérer de mot de passe pour celui-ci. Cependant, les comptes virtuels possèdent des clés DPAPI qui leur permettent d'utiliser le *Credential Manager*. Ce sujet étant assez complexe, il ne sera pas abordé dans cet article. Adam Chester (@xpn) a rédigé un excellent article de blog expliquant en détail toute l'attaque, ainsi qu'une preuve de concept permettant de déchiffrer ce mot de passe. Dans cet environnement LAB, le compte *GOTHAM\bruce.wayne*, qui est administrateur local de tous les serveurs, a été utilisé : **Figure 8**

La configuration par défaut d'*Azure AD Connect* utilise une base de données *SQL Server Express*, mais un serveur SQL Server entièrement déployé peut également être utilisé. Dans ce cas, la chaîne de connexion du POC doit être remplacée par la suivante :

"Server=LocalHost;Database=ADSync;Trusted_Connection=True;".

Il est important de noter que cette technique est plutôt furtive et n'est pas identifiée, au moment de la rédaction, par un *Windows Defender* à jour comme un comportement malveillant. En disposant d'un compte administrateur local sur le serveur, il serait également possible de récupérer le mot de passe MSOL dans la mémoire du processus *lsass.exe*. Cependant, cette opération est beaucoup plus suspecte et serait facilement détectée par une équipe de défense expérimentée.

À titre d'exemple, l'outil *mimikatz* a été utilisé pour extraire le mot de passe MSOL de la mémoire de lsass : **Figure 9**

En utilisant le compte MSOL nouvellement obtenu, il est maintenant possible d'effectuer une attaque *DCSync* et de répliquer tous les condensats de mots de passe des utilisateurs du domaine : **Figure 10**

Il ne faut pas oublier que ce compte est également valide et dispose de privilèges élevés dans le cloud.

Ce scénario est une voie intéressante vers la compromission du domaine *Active Directory*, car les serveurs de synchronisation Azure AD sont souvent moins protégés que les contrôleurs de domaine, ce qui facilite la tâche aux équipes d'attaque pour devenir administrateur de ces serveurs.

La suite de ce dossier dans Programmez! n°260

La sécurité des données dans MariaDB

La manipulation des données a toujours existé dans l'informatique pour faciliter le traitement des informations et des applications. Mais la sécurité se limite trop souvent à l'injection SQL. Aujourd'hui les données concernent tous les domaines (Big Data, IoT, Cloud, la voiture, etc.). Il est indispensable de repenser la sécurité des bases de données qu'elles SQL, NoSQL. MariaDB propose des mécanismes par défaut.

Christophe Villeneuve

Consultant open source pour 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), Lizard.

La configuration de la sécurité par les développeurs doit tenir compte de certains facteurs, car les menaces sont diverses dans leurs natures et leurs origines. MariaDB offre un large panel pour sécuriser les bases et les accès et faire face aux nombreuses menaces.

Du côté des actions de sécurisation à mener, les points importants concernent :

- La donnée
- du serveur
- du cluster
- de l'architecture
- du déploiement

Comme nous le voyons, la sécurité d'un SGBD concerne toutes les couches et non pas uniquement la donnée ou les accès.

Les menaces de sécurité communes

Injection SQL

L'attaque par injection SQL est le premier réflexe que nous devons avoir. Elle se produit lorsqu'une personne mal intentionnée tape (ou injecte) des commandes en langage SQL dans les entrées de l'utilisateur. À leur tour, ces commandes injectées seront exécutées, modifiant la logique des applications, modifier le comportement des requêtes, etc.

Exemple : permettre aux utilisateurs d'accéder aux données des autres utilisateurs où celles-ci seront exposées à des données auxquelles ils ne devraient pas avoir accès. Dans de nombreux cas, cela peut inclure l'ensemble de la base de données, et dans certains cas, cela peut permettre aux attaquants de prendre le contrôle du serveur.

L'une des meilleures façons de se protéger contre les attaques par injection SQL consiste à mettre en œuvre des moyens d'empêcher les utilisateurs d'injecter du code dans le contenu fourni par l'utilisateur, comme les chaînes de requête, les données des messages, les en-têtes HTTP, etc.

Les requêtes paramétrées et la validation des entrées sont les moyens de prévenir l'injection, afin de bloquer les entrées erronées plutôt que de les faire passer par la base de données.

Attaques DDoS

Le déni de service distribué (ou DDoS) est une des attaques les plus courantes et les plus agressives pour saturer une base de données. Comme son nom l'indique, le DDoS se produit lorsqu'une base de données est submergée par un afflux rapide de requêtes provenant de plusieurs comptes. Si l'attaque dure assez longtemps, cette attaque peut ralentir, voire bloquer la base de données, la rendant inutilisable jusqu'à ce que l'attaque s'estompe ou que les requêtes soient interrompues.

Comme les attaques DDoS contre les bases de données sont généralement indirectes, c'est-à-dire qu'elle est effectuée sur une application qui interroge à son tour la base de données. La solution est de traiter cela au niveau de l'application. Vous pouvez minimiser les menaces DDoS en désactivant les connexions persistantes et en fermant les connexions ouvertes qui entraînent des erreurs. De temps en temps, des vulnérabilités sont trouvées dans les logiciels. Si c'est le cas, un ensemble de vulnérabilités d'escalade de privilèges du moteur de connexion peut apparaître.

Les risques seront :

- Débordement de la mémoire tampon basé sur le tas (Heap-Based Buffer Overflow Privilege Escalation)
- Élévation des privilèges de la chaîne de format
- Élévation de privilèges par Use-After-Free
- Débordement de mémoire tampon basé sur la pile - Escalade de privilèges

Toutes ces vulnérabilités nécessitent une authentification et sont utilisées pour élever les privilèges de l'utilisateur. Elles exploitent des failles présentes dans les processus d'interrogation SQL, mais en tirant parti de différents facteurs au sein de ces processus.

Par exemple, l'escalade des privilèges par débordement de la mémoire tampon basée sur un head bases attacks se produit lorsque l'attaquant tire parti d'une mauvaise validation des données fournies par l'utilisateur avant qu'elles ne soient copiées dans une mémoire tampon de longueur fixe basée sur le head bases attacks.

Il est recommandé de surveiller les annonces de vulnérabilités logicielles. MariaDB propose une page dédiée que vous pouvez suivre https://mariadb.com/kb/en/security/

Attaques de l'homme du milieu (man of the middle)

Ce type d'attaque est appelé attaque de l'homme du milieu (ou machine du milieu ou MITM) touche le transport des données entre l'application et la base de données.

Les utilisateurs supposent, à tort, que ces transferts sont confidentiels et sécurisés entre l'expéditeur et le destinataire. Bien que ce soit généralement le cas grâce aux bonnes pra-

tiques de chiffrement et de sécurisation du tunnel de connexion, il y a toujours un hacker qui veut écouter ces connexions et voler des données, ou injecter de mauvaises données

Les attaques MITM se produisent lorsqu'un attaquant détourne, ou corrompt, la communication entre deux parties. Cela peut se produire lorsque l'attaquant est capable de lire des données en clair ou de déchiffrer des données envoyées entre des utilisateurs, un utilisateur et une base de données, ou deux applications ou plus.

La meilleure façon de se défendre contre les attaques MITM est de garder MariaDB à jour, de l'isoler du point de vue du réseau (dans un VPC ou un réseau interne), et d'exiger le chiffrement TLS. Vous pouvez aussi vérifier l'intégration des données en entrée et en sortie et mettre en place un monitoring réseau pour observer son comportement et détecter des anomalies. En effet, un MITM provoque souvent une altération brève de la communication.

Corruption des données

La corruption des données est un risque faible, mais il ne faut pas pour autant l'exclure. Cela peut se produire dans les cas suivants :

- Un utilisateur écrit, modifie ou transfère des données de manière incorrecte,
- Un processus ou une requête est interrompu, lorsqu'une ligne de code erronée est injectée
- Une panne secteur
- Un arrêt inapproprié

La vérification régulière des données dans la base de données ou d'une requête ou tables doit être liée à la politique de sauvegarde et de récupération afin de pouvoir revenir à une version antérieure de la base de données en cas de besoin.

Configuration de la sécurité

Authentification et autorisation de l'utilisateur

L'authentification (authn) et l'autorisation (authz) est la capacité d'un utilisateur à se connecter à la base avec le niveau de privilèges qui lui est donné. Les éléments de la base, tels que les objets, les tables, les lignes et les colonnes, peuvent tous être dotés de privilèges, ce qui vous permet de contrôler de manière granulaire qui a accès à quoi.

Vous pouvez définir une méthode authentification différente de SHA-1 qui est proposée par défaut, par ed25519, named pipe, PAM, Unix socket, GSSAPI

Pour cela, il faut installer le plug-in :

```
INSTALL SONAME 'auth_ed25519';
```

et vous pouvez créer un utilisateur avec la méthode ed25519 :

```
CREATE USER username@hostname IDENTIFIED VIA ed25519 USING PASSWORD('secret');
```

Vous pouvez aller plus loin à partir de la page https://mariadb.com/kb/en/authentication-plugin-ed25519/

Les privilèges peuvent être encore plus granulaires en utilisant le masquage de données et des stratégies de sécurité au niveau des lignes, en plus de vos plug-ins ou applications de sécurité et authn/authz préférés.

Ne sous-estimez pas les autorisations et les privilèges dans votre politique de sécurité.

Vous pouvez consulter la page suivante :
https://mariadb.com/fr/resources/blog/mariadb-maxscale-masking-and-firewall/

Audit

Malgré les protections que vous mettez en place, une base de données vit, c'est pourquoi il est important d'auditer régulièrement vos données et vos bases, simplement pour des raisons pratiques : enquêtes de sécurité, le respect réglementaire, versionning, etc.

MariaDB possède son propre plug-in d'audit qui peut être utilisé pour enregistrer :

- L'activité du serveur
- Les connexions des utilisateurs et des hôtes
- Les requêtes exécutées
- Le(s) table(s) ou variables qui ont été accédées ou modifiées.

Pour aller plus loin https://mariadb.com/kb/en/mariadb-audit-plugin/

Chiffrement

MariaDB peut utiliser le chiffrement des données. Plusieurs méthodes et outils sont utilisés par MariaDB pour le chiffrement :

- Chiffrement des données au repos
- Chiffrement des données en transit
- Certificats TSL/SSL
- Chaînage de blocs de chiffrement (CBC)

L'installation du chiffrement s'effectue de la manière suivante :
Activer le plugin « file key management » et la déclaration la configuration

[mariadb]

```
plugin-load-add=file_key_management
file-key-management
file_key_management_encryption_algorithm=aes_cbc
file_key_management_filename = /etc/mysql/encryption/keyfile
file_key_management_filekey = ff5b3126d19674570890bffd789e0a12
innodb-encrypt-tables
innodb-encrypt-log
```

La prise en compte du chiffrement se passe avec le terminal de MariaDB et l'exemple ci-dessous montre le chiffrement de la table results.

```
ALTER TABLE results ENCRYPTED=YES ENCRYPTION_KEY_ID=1;
```

Pour aller plus loin, vous pouvez consulter la page suivante https://mariadb.com/kb/en/file-key-management-encryption-plugin/

Meilleures pratiques de sécurité

MariaDB vous met à disposition des bonnes pratiques à prendre en compte pour sécuriser sa base de données.

S'assurer que les comptes utilisateurs ont des informations d'identification solides

Certaines versions de MariaDB sont livrées avec des utilisateurs anonymes qui par défaut n'ont pas de mot de passe. Bien que les comptes anonymes aient généralement des privilèges très faibles (ou nuls), le fait de donner à des comptes sans mot de passe l'accès à la base de données constitue un risque potentiel.

Ayez le réflexe de supprimer les comptes d'utilisateurs sans mot de passe. Attention : tous les mots de passe doivent comporter une chaine d'au moins 15 caractères alphanumériques et symboles. Idéalement, ces chaînes ne devraient pas inclure de mots ou de phrases courants trouvés dans les dictionnaires, car ceux-ci sont utilisés par les scanners de mots de passe (merci les keyloggers).

La vérification des comptes s'effectue dans la console de MariaDB :

```
MariaDB [(none)]> select user, host, password from mysql.user where user like '';
```

pour obtenir le résultat suivant : **figure 1**

Comme le montre l'exemple, l'accès anonyme est limité à partir de localhost. Toutefois, bien que les privilèges soient limités, ils peuvent toujours exécuter certaines commandes qui peuvent afficher plus d'informations sur la base pouvant aider à identifier d'autres vecteurs d'attaque. La possibilité d'effectuer ce type d'attaque peut aussi s'appliquer avec un mot de passe faible ou identifié par deny of service.

Conserver les informations d'identification en toute sécurité

Lorsque vous utilisez des identifiants, vous devez vous assurer qu'ils, en particulier les identifiants forts (administratifs ou possédant des accès à des données sensibles), sont conservés en toute sécurité.

Exemple : les identifiants de production doivent être stockés dans des coffres-forts, et les développeurs ne doivent pas stocker d'identifiants dans des endroits tels que les canaux slack publics, les communications par e-mail, les tickets de support, etc. JAMAIS METTRE EN DUR (par exemple : dans un fichier plat ou un code) DES IDENTIFIANTS, MOTS DE PASSE.

Modifier les paramètres de la base par rapport à leurs valeurs par défaut

Les paramètres par défaut de la base sont très utiles pour installer et démarrer rapidement. Cependant, ces valeurs par défaut peuvent les attaques car ces paramètres sont connus et documentés. Personnalisez rapidement les paramètres de votre base, notamment en modifiant le port par défaut, en accordant aux utilisateurs les privilèges appropriés et en supprimant les comptes anonymes.

Vérifiez qu'aucun paramètre par défaut reste dans votre base et serveur de données.

Restrictions des accès

La restriction des accès est le moyen d'accorder les privilèges

Figure 1

aux utilisateurs. Il n'y a aucune raison pour que l'assistant administratif du service marketing ait le même niveau d'accès que l'administrateur de la base de données.

Pour garantir la sécurité de votre base, accordez à chaque utilisateur juste assez de privilèges pour qu'il puisse faire son travail, ni plus, ni moins. Le principe du moindre privilège minimise les risques que des acteurs malveillants prennent le contrôle d'un compte disposant de plus d'autorisations qu'il n'en faudrait.

Il faut ajuster les privilèges des utilisateurs lorsqu'ils changent de poste ou de rôle. S'il est bon d'attribuer des privilèges à chaque nouvel utilisateur ajouté à la base de données, il est préférable de vérifier régulièrement les privilèges des utilisateurs - une tâche que de nombreuses entreprises négligent.

En outre, lorsque vous accordez des privilèges temporaires pour un projet (ce qui arrive souvent), assurez-vous que ces privilèges sont révoqués lorsqu'ils ne sont plus nécessaires.

Soyez proactif et non passif sur la politique d'accès.

Pour cela la gestion des droits :

```
CREATE ROLE role;
GRANT privileges ON db-objects TO role;
```

Et définir le choix du rôle actif sur la session :

```
SET ROLE role;
SET ROLE NONE;
```

La visualisation du rôle actif sur la session :

```
SELECT CURRENT_ROLE();
```

Vous pouvez avoir plus de détails https://mariadb.com/kb/en/roles/

Effacer les données de test

Lors de l'installation d'une base sur un nouveau serveur, le cahier de recette propose de tester avec un jeu de données de test pour vérifier que celle-ci fonctionne correctement. Ce jeu de données n'est pas utile pour votre production, mais peut parasiter votre base de données.Il est indispensable de supprimer la base de test et le compte associé.

Obscurité d'accès

Il est bien connu que MariaDB fonctionne sur le port 3306, et son super-utilisateur est appelé « ROOT ». Pour rendre les choses plus difficiles, il est assez simple de changer cela. Dans une certaine mesure, il s'agit d'un exemple de sécurité par l'obscurité, mais il peut au moins arrêter les tentatives automatisées d'accéder à l'utilisateur « root ». Pour changer de port, vous devez éditer my.cnf et définir la

variable « port » vers une autre valeur.
En ce qui concerne les utilisateurs – après l'installation de MariaDB, vous devez créer un nouveau super-utilisateur (GRANT ALL... AVEC L'OPTION GRANT) et ensuite, supprimer les comptes « ROOT ».

Procéder à des évaluations régulières de la sécurité des données

Il va sans dire que vous devez vous assurer régulièrement de la sécurité de vos données. Tout comme vous sauvegardez vos données en cas de corruption ou de panne soudaine, la réalisation d'audits réguliers vous permet d'identifier les comportements douteux ou malveillants, ainsi que les dysfonctionnements au sein de la base de données elle-même. Effectuez ces évaluations pour identifier les correctifs nécessaires à votre MariaDB.

Sécurité des réseaux

Idéalement, MariaDB ne doit pas être disponible sur le réseau et toutes les connexions seraient gérées localement, via le socket Unix. Dans certaines configurations, c'est possible - dans ce cas, vous pouvez ajouter la variable « skip-networking » dans my.cnf. Cela empêchera MariaDB d'utiliser n'importe quelle communication TCP/IP, seul Unix socket serait disponible sur Linux.
La plupart du temps, une sécurité aussi étroite n'est pas possible. Dans ce cas, vous devez trouver une autre solution. Tout d'abord, vous pouvez utiliser votre pare-feu pour n'autoriser le trafic que d'hôtes spécifiques vers le serveur MariaDB. Par exemple : les hôtes d'application, la couche proxy, et peut-être un serveur de gestion. D'autres hôtes de votre réseau n'ont probablement pas besoin d'un accès direct au serveur MariaDB. Cela limitera les possibilités d'attaque sur votre base de données, dans le cas où certains hôtes de votre réseau seraient compromis.

Si vous utilisez des proxys qui permettent une correspondance d'expression régulière pour les requêtes, vous pouvez les utiliser pour analyser le trafic SQL et bloquer les requêtes suspectes. Il est très probable que vos hôtes d'application ne devraient pas exécuter :

```
DELETE * FROM votre_table
```

sur une base régulière.
S'il est nécessaire de supprimer certaines données, elles peuvent être exécutées à la main, localement, sur l'instance Mariadb. Ainsi, vous pouvez créer de telles règles en utilisant quelque chose comme ProxySQL : bloquer, réécrire, redirection de telles requêtes. MaxScale vous donne également la possibilité de bloquer les requêtes basées sur des expressions régulières.
Au niveau du code, cela se traduit de la manière suivante :

```
CREATE USER 'alice'@'%'
REQUIRE SUBJECT '/CN=alice/O=My Dom, Inc./C=US/ST=Oregon/L=
Portland'
AND ISSUER '/C=FI/ST=Somewhere/L=City/ O=Some Company/CN=Peter
Parker/emailAddress=p.parker@marvel.com'
AND CIPHER 'SHA-DES-CBC3-EDH-RSA';
```

Pour aller plus loin https://mariadb.com/kb/en/configuring-mariadb-for-remote-client-access/

Conserver un journal d'audit accessible

Vous devez disposer d'un journal d'audit des accès aux données, vous permettant de savoir qui a accédé à quelles données et à quel moment. L'audit de la sécurité des données est également imposé par la loi : GPDR, CCPA/CPRA et HIPPA sont quelques exemples d'audits obligatoires. Les organisations qui ne respectent pas cette réglementation s'exposent à des amendes et à une perte de confiance des consommateurs si le public sait que ces données ne sont pas aussi sûres qu'elles devraient l'être. Chaque semaine, de nouvelles fuites de données sont annoncées.

Désactiver la localisation des données

Si le serveur et le client ont la possibilité d'exécuter LOAD DATA LOCAL INFILE, un client sera en mesure de charger des données à partir d'un fichier local vers un serveur MariaDB distant. Cela peut potentiellement aider à lire des fichiers auxquels le client a accès.
Exemple : sur un serveur d'application, l'un d'entre eux pourrait accéder à n'importe quel fichier auquel le serveur HTTP a accès. Pour l'éviter, vous devez ajouter la variable local-infile dans le my.cnf.
L'implémentation est simple :

```
local-infile=0
```

Plus de détails sur le lien suivant : https://mariadb.com/kb/en/load-data-infile/

Privilèges de fichiers

La configuration de l'OS doit être associée à la sécurité de la base : car celle-ci stocke les données sous forme de fichiers. Le serveur MariaDB écrit de nombreuses informations dans les journaux. Parfois, ces informations contiennent des données (parfois sensibles), un journal des requêtes lentes, un journal général ou un journal binaire... Vous devez vous assurer que ces informations sont sûres et accessibles uniquement aux utilisateurs qui doivent y accéder.
Typiquement, cela signifie que la racine et l'utilisateur ont les droits sur le MariaDB en cours d'exécution. Ils devraient avoir accès à tous les fichiers liés à MariaDB. La plupart du temps, c'est un utilisateur dédié. Vous devez vérifier les fichiers de configuration MariaDB et tous les journaux générés par MariaDB et vérifier qu'ils ne sont pas lisibles par d'autres utilisateurs.

Conclusion

La sécurité des données est une priorité absolue pour chaque DBA et développeur. Que votre motivation soit le respect des exigences réglementaires ou la protection de vos clients et de la réputation de votre entreprise, les conseils pour sécuriser vos bases MariaDB vous aideront à sécuriser davantage votre infrastructure et à vous donner la tranquillité d'esprit. Dans cet article, nous avons couvert les fondamentaux tels que le chiffrement des données, les contrôles d'accès au réseau, l'authentification et les privilèges des utilisateurs, la sécurité du système d'exploitation, et plus encore.

MegaLinter : votre meilleur allié pour du code clean et sécurisé !

Si vous lisez cet article dans Programmez!, c'est probablement que vous savez coder. Mais savez-vous bien coder ? Respectez-vous les bonnes pratiques des langages que vous utilisez ? Et laissez-vous traîner des failles de sécurité par ignorance voire inadvertance ?

Nicolas Vuillamy

Nicolas a commencé la programmation à 10 ans sur un Commodore 64, joué avec du PHP / MySql à 15 ans, arrêté prématurément ses études à 16 ans, et était développeur junior en CDI à 17 ans. Il est aujourd'hui CTO chez Cloudity, intégrateur Pure Player Salesforce. Il est aussi très actif dans l'écosystème open source, ayant créé et gérant la maintenance de différents projets connaissant un certain succès, comme npm-groovy-lint, VsCode Groovy Lint, MegaLinter, sfdx-hardis et VsCode SFDX Hardis.

https://github.com/nvuillam | https://www.linkedin.com/in/nicolas-vuillamy/ | https://nicolas.vuillamy.fr/

Vous pouvez choisir de mimer le comportement d'un grand mammifère australien, et enfouir votre tête dans le sable, car « après tout ça marche », ou encore faire confiance à de petits bijoux créés et maintenus par les plus grands esprits de la communauté des développeurs open source : les **linters** !

En les utilisant, vous :

- Perfectionnez **vos connaissances** envers vos langages de programmation préférés
- Évitez de mettre des « **code smells** » en production (du code qui fonctionne, mais est peu optimisé et évolutif… en clair : du code moche)
- Améliorez les **performances** de vos applications
- Prévenez les **risques de sécurité**

Il existe des centaines de linters, que vous pouvez par exemple trouver sur l'excellent site *analysis-tools.dev*, mais quels sont les plus adaptés à votre projet ? Et comment les mettre en place ? Une de vos options est de **les sélectionner, les installer et les configurer un par un**, mais cela peut s'avérer long et fastidieux.

La seconde option est d'utiliser des **agrégateurs de linters**. Ces outils réunissent les linters les plus populaires, automatisent leur exécution et agrègent leurs résultats. La plupart de ces outils sont payants, comme SonarQube, Codacy et CodeScene, cependant il existe des **alternatives 100% open source et gratuites** comme Super-Linter et **MegaLinter**.

Super-Linter a initialement été conçu par l'équipe DevOps de GitHub, et MegaLinter est un hard-fork, avec une réécriture complète de bash à python, apportant **un grand nombre de capacités supplémentaires**, ainsi que de bien **meilleures performances**.

MegaLinter

MegaLinter contient plus de 100 linters qui vont permettre d'analyser et formater le code de tous types de projets, quel que soient les langages et technologies utilisées dans ces derniers.

Les linters sont classés en plusieurs catégories :

- **Langages** : Bash, C, Clojure, C++, C#, Dart, Go, Groovy, Java, Javascript, Kotlin, Lua, Makefile, Perl, PHP, Powershell, Python, Raku, Ruby, Rust, Salesforce (Apex, LWC) , Scala, SQL, Swift, Typescript, VBA
- **Langages descriptifs** : CSS, .env, GraphQL, HTML, JSON, LaTex, Markdown, Protocol Buffers, RestructuredText, XML, YAML
- **Technologies, souvent liées au DevOps et à l'IaC (Infrastructure as Code)** : GitHub Actions Workflow, Ansible, Azure Resource Manager, Bicep, CloudFormation, Dockerfile, EditorConfig, Gherkin, Kubernetes, OpenAPI, Puppet, Snakemake, Tekton, Terraform
- **Sécurité** : Checkov, Devskim, GitLeaks, Grype, Kics, Secretlint, SemGrep, Trivy, Trufflehog
- **SBOM** (Software Bill Of Material) : génère des rapports de composition des applications, de leurs dépendances et licences associées
- **Documentation** : Orthographe, style de rédaction, détection de lien hypertextes morts
- Détecteurs de **copier-coller excessifs**

Voici quelques exemples de linters populaires nativement intégrés : eslint, prettier, black, pylint, flake8, shellcheck, checkstyle, PMD, golanci-lint, rubocop, stylelint, hadolint, kubescape, tflint.

L'usage le plus courant est dans le cadre de **jobs d'intégration continue**, qui vont permettre de contrôler les évolutions proposées dans chaque Pull Request, à chaque nouveau commit sur le serveur git : MegaLinter va lui-même poster un commentaire avec les résultats, grâce à son **intégration native avec GitHub, Gitlab, Azure et BitBucket. Figure 1**

Note : *Il est aussi possible de lancer MegaLinter en local grâce au package NPM mega-linter-runner, éventuellement déclenché par un hook pre-commit, mais l'architecture étant basée sur docker, je trouve personnellement que c'est un peu lourd à l'usage.*

Un projet avec MegaLinter, c'est l'assurance d'avoir une base de code propre, sécurisée, ainsi que des normes et un formatage communs à toute l'équipe :

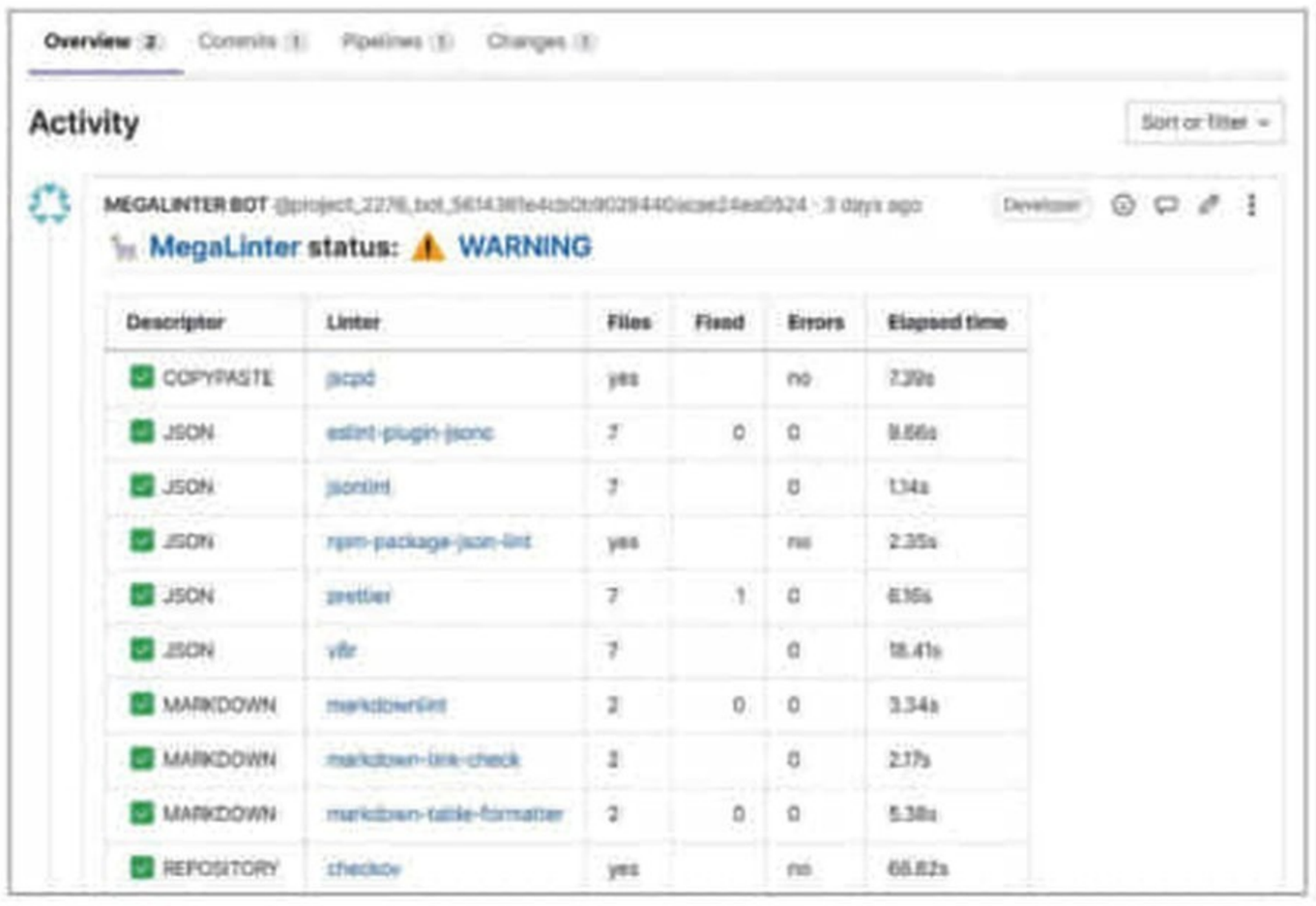


Figure 1 : *Exemple de résultats MegaLinter sur une Merge Request Gitlab*

MAINTENANCE ET COMMUNAUTÉ

Chacun des linters est soigneusement sélectionné par la communauté et validé par le **groupe d'administration** constitué de quatre « Maintainers » qui s'assurent de la **cohérence de l'architecture** et le **release management.**

Si un linter n'est plus maintenu, il sera considéré comme moins fiable, puis déprécié et enfin retiré lors de la version majeure qui suivra.

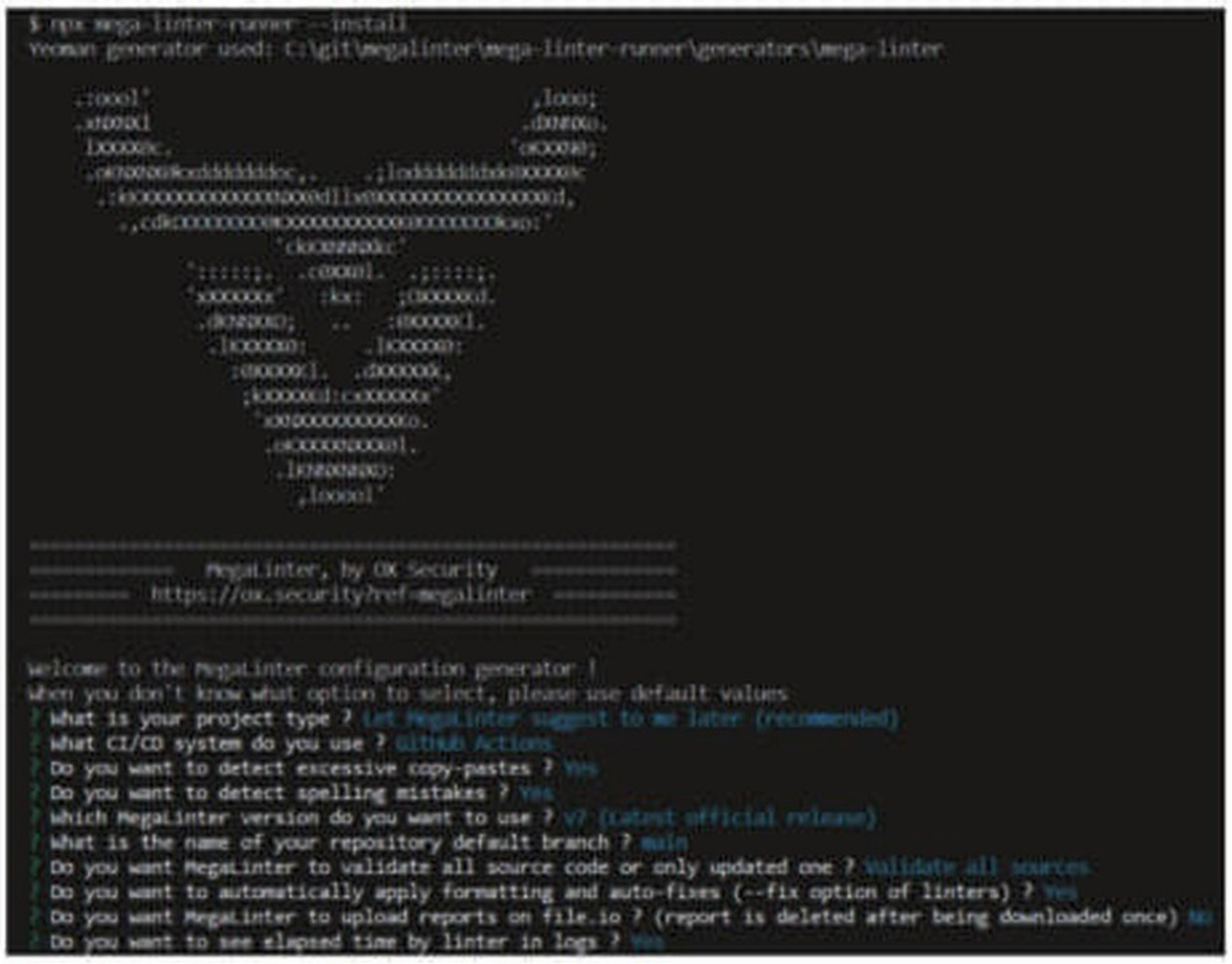
La **communauté est très active** sur le repository GitHub : une **centaine de contributeurs** ont déjà participé à des évolutions et corrections de bugs sur l'architecture et la documentation, et les questions posées restent très rarement sans réponse grâce à l'entraide entre utilisateurs, qui n'hésitent pas non plus à écrire des articles (une quinzaine à aujourd'hui) et créer des vidéos expliquant comment facilement utiliser leur application de clean code et sécurité préférée !

Sponsorisé par la start-up OX Security (spécialisée dans le Software Supply Chain Security), **MegaLinter sera toujours 100% open source et gratuit**, donc si vous décidez de l'utiliser, aucune chance que du jour au lendemain vous receviez une facture !

De **nombreuses entreprises commerciales** utilisent MegaLinter en interne, ce qui est tout à fait autorisé par la licence AGPLV3 tant qu'il n'y a pas mise en place d'un service monétisé basé sur des analyses MegaLinter.

Ces entreprises peuvent se faire assister pour l'installation et la configuration de MegaLinter par des **experts reconnus**, comme Étienne Deneuve et ses collègues de Simplifi'ED et IT Challenge, qui le rendent bien à la communauté en fournissant feedbacks et contributions, les dernières en date étant l'intégration de Kubescape en natif ainsi que l'auto-commit sur Azure Pipelines.

Figure 2



- **Clean code** obligatoire !
- **Finis les débats stériles** sur l'indentation en espaces ou en tabulations, car si vous le souhaitez MegaLinter est capable de lui-même **pousser de nouveaux commits après reformatage automatique du code** !
- **Finis les credentials en dur** dans le code qui seront ensuite oubliés et récupérés par des hackers !
- **Finis les copier-coller** de dizaines de lignes de code « parce que ça va plus vite, on le refera plus tard », car même les mauvaises langues l'admettent : le « on refera plus tard » veut très souvent dire « jamais » quand on parle de code, et après ça donne des fonctionnalités impossibles à maintenir et faire évoluer !
- **Finies les vilaines coquilles** dans la documentation !
- **Finis les liens morts** qui rendent fous nos pauvres utilisateurs !

Installation

Pour installer MegaLinter sur votre repository, c'est très simple, après vous êtes assurés que vous disposez de Node.js (https://nodejs.org/) sur votre ordinateur, allez à la racine de votre projet et tapez la ligne de commande suivante :

```
npx mega-linter-runner —install
```

Cela lancera l'installeur automatique de MegaLinter, et après avoir répondu à quelques questions vous verrez apparaître des fichiers supplémentaires, dont :

- Un **workflow GitHub Actions** (ou Gitlab, Azure ou Bitbucket selon votre git provider)
- Un fichier de configuration pré-renseigné : **.mega-linter.yml**. **Figure 2**

Committez les fichiers, poussez le commit sur le serveur, et la première analyse sera lancée !

Selon le contenu de votre projet, MegaLinter va automatiquement sélectionner et exécuter les linters appropriés avec une configuration par défaut : vous disposez donc **après seulement une commande et un commit d'une analyse complète de votre projet** par les linters les plus populaires parmi la communauté open source !

Configuration

La première exécution de MegaLinter devrait logiquement prendre quelques minutes et remonter beaucoup d'erreurs, que vous verrez apparaître en commentaire de votre Pull Request, ou alors directement dans les logs de votre job CI.

Figure 3

Concernant les performances, afin d'**optimiser le chargement** de MegaLinter, ce dernier est décliné sous de nombreuses **saveurs (Flavors)**, contenant uniquement les linters adaptés à certains types de projets (Documentation, JavaScript, Go, Python, Salesforce…), qui vous seront suggérés lors de la première exécution de MegaLinter.

Concernant les erreurs, pas de panique : beaucoup de linters sont très bavards, et vous allez pouvoir traiter les erreurs détectées de différentes manières :

- **Corriger l'erreur** dans votre code, tout simplement !
- Configurer le linter pour qu'il **ignore une ou plusieurs erreurs spécifiques**

- Configurer MegaLinter pour qu'il **ignore des fichiers ou dossiers**, pour un linter en particulier ou chacun d'entre eux
- Définir le **linter comme non bloquant** (en vue de correction ultérieure ou non, selon votre évaluation de la criticité du linter)
- Définir un **nombre maximum d'erreurs pour un linter donné** : cette technique permet de ne pas avoir à corriger immédiatement les erreurs, tout en bloquant les Pull Request ultérieures en cas de nouvelles erreurs.

Toutes les variables de configuration de MegaLinter sont décrites en détail dans la documentation en ligne commune, ainsi que sur les pages dédiées à chacun des linters. **Figure 4**

Après ces opérations, qui peuvent prendre 5 minutes si vous rendez juste non bloquants les linters en erreur, comme des heures si vous décidez d'améliorer votre code et éliminer tous les risques de sécurité identifiés, votre projet sera déjà plus propre et sécurisé !

Voici un exemple de fichier de configuration **.mega-linter.yml**, qui pour plus de confort dispose d' auto-complétions et d'auto validation dans votre IDE préféré ! *(pour les utilisateurs de Visual Studio Code, installez l'extension YAML de Red Hat)* **Figure 5**

Usage

Mais en fait, il ne manquerait pas un truc dans le paragraphe précédent ? C'est bien sympa d'afficher le nombre d'erreurs affichées par chaque linter, mais comment effectivement consulter le détail de ces erreurs ?

Une fois de plus c'est très simple, et il existe trois manières différentes :

- Lire les **logs console d'exécution** du job, classés par linter, avec des sections de log dépliables pour une meilleure navigation
- Télécharger les **artifacts** : un fichier zip contient les erreurs en sortie de chacun des linters dans des fichiers distincts
- Utiliser le mode **SARIF** pour une intégration avec les UI de GitHub, Azure, ou même GitLab si vous utilisez l'excellente intégration de MegaLinter par la team R2DevOps.io.

Figures 6, 7 et 8

Et pour chacune des erreurs, la **documentation en ligne sur megalinter.io** vous donne des raccourcis vers les pages officielles de documentation publiées par les linters eux-mêmes :

- Nom et format de **fichier de configuration**, vous permettant de facilement affiner les options d'analyse des linters
- Nom et format de **fichier d'exclusion** (ignore)
- **Liste des erreurs détectables** par le linter, avec des indications permettant de corriger ces dernières
- Instructions à utiliser pour **ignorer une erreur directement dans le code** (inline-disable)
- Repository, statistiques et popularité des linters
- License

Grâce à ces liens directs, pas besoin de chercher vous-même sur google ou ChatGPT les informations cruciales concernant vos linters préférés, MegaLinter vous mâche le boulot !

La documentation en ligne vous fournit aussi de précieuses informations permettant de personnaliser l'utilisation d'un linter dans MegaLinter :

- **Version du linter** dans MegaLinter
- **Conditions de déclenchement** du linter (extensions de

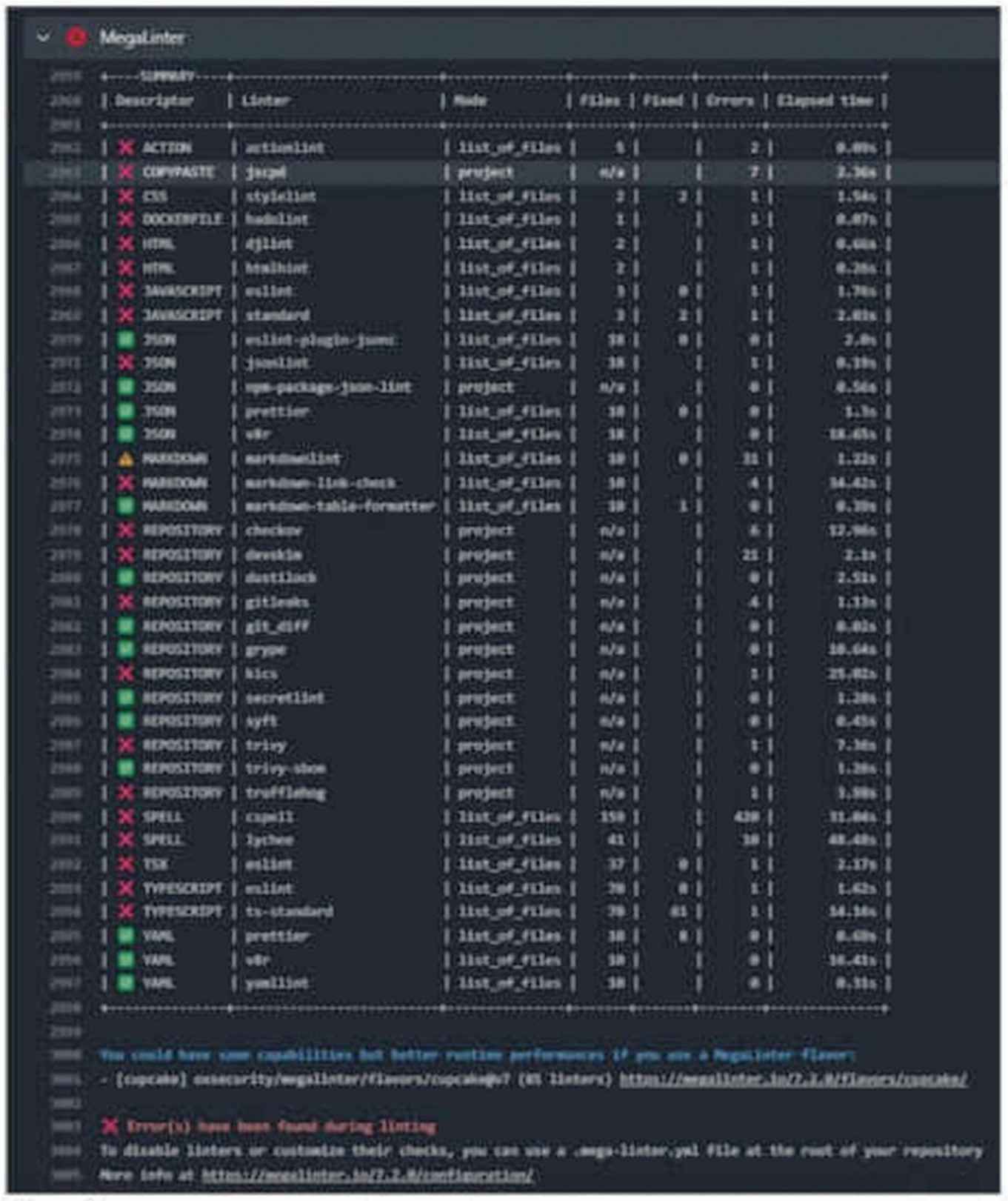


Figure 3

Figure 4

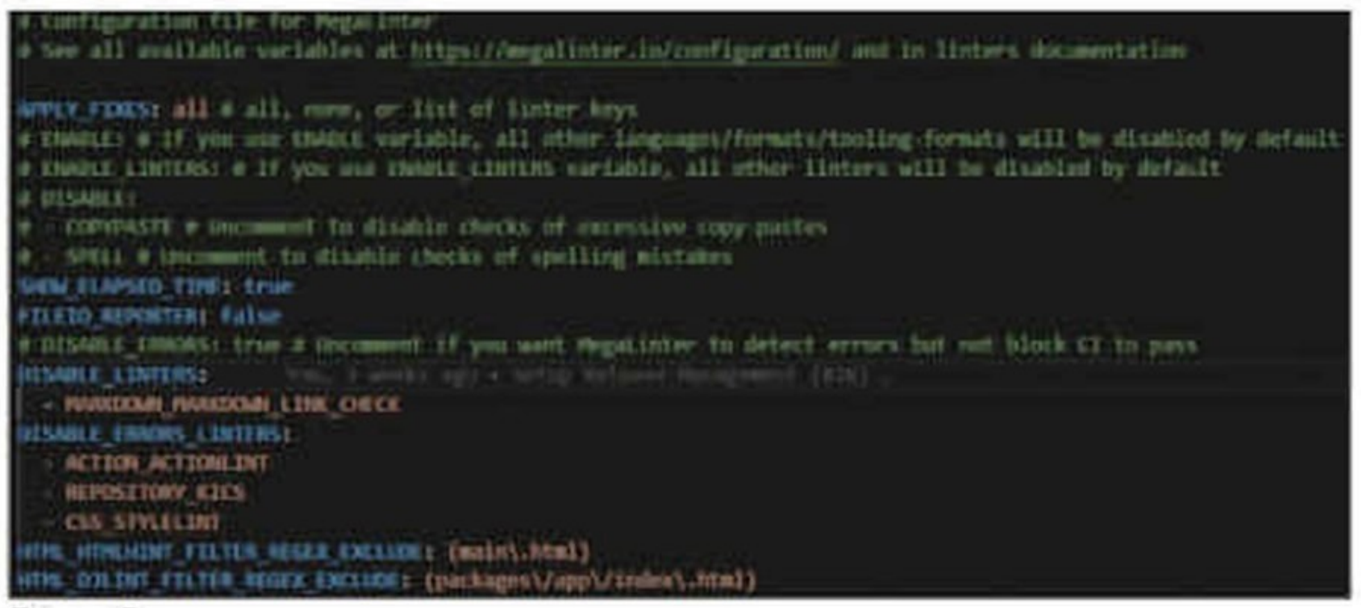


Figure 5

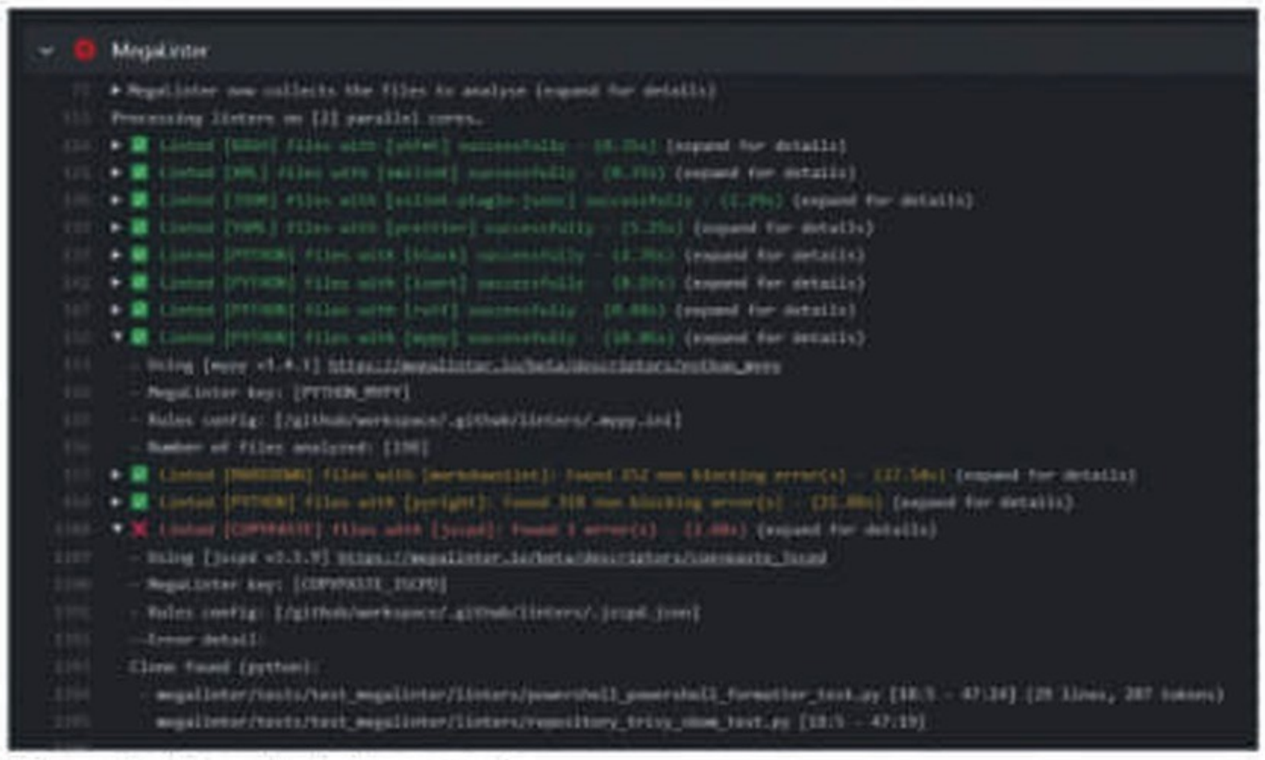


Figure 6 : *Exemple de logs console*

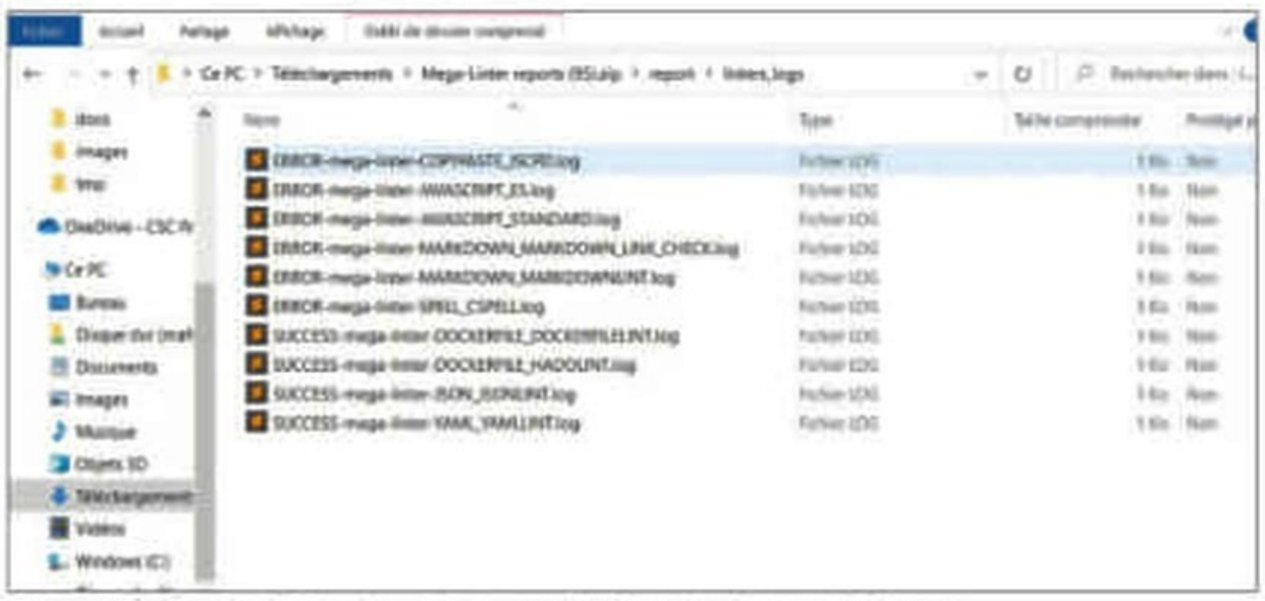


Figure 7 : *Exemple d'artefacts contenant les logs de chacun des linters*

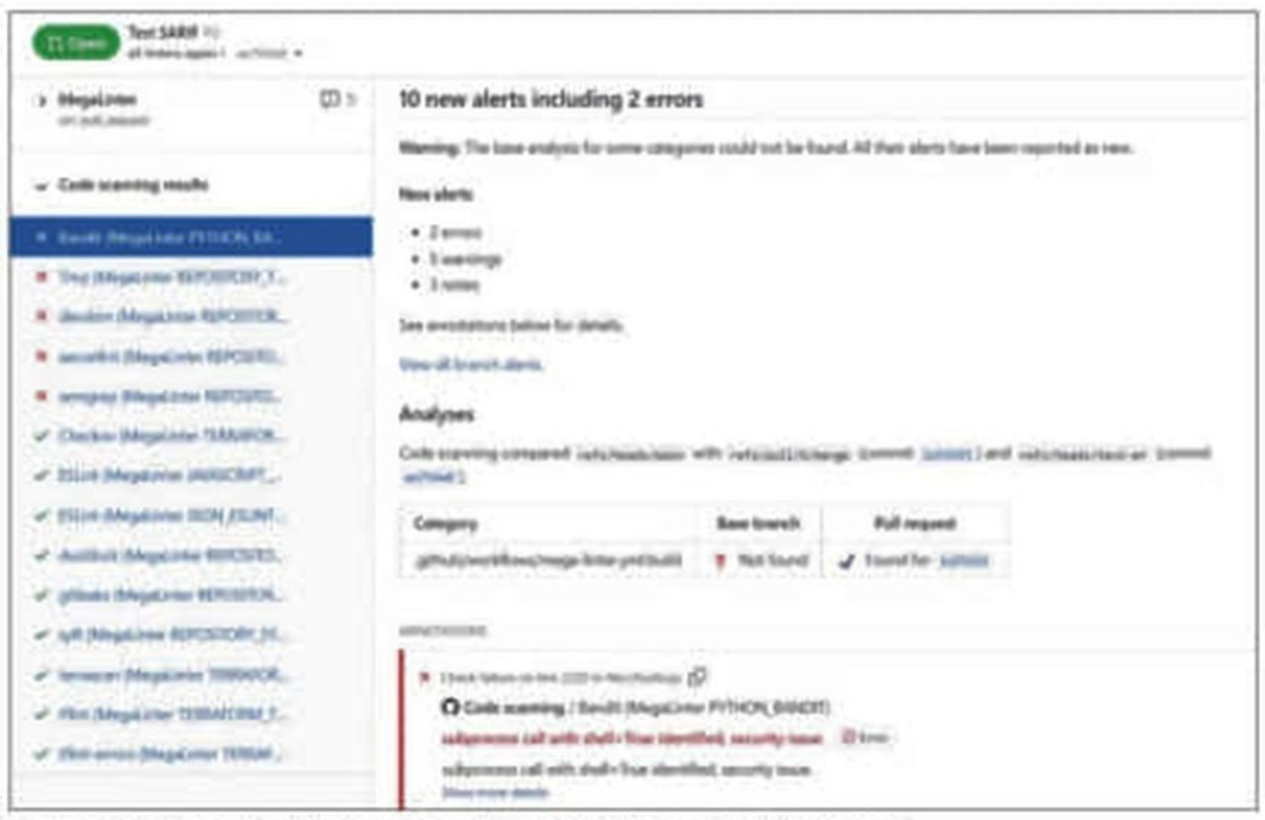


Figure 8 : *Exemple d'intégration de logs SARIF avec les UI GitHub*

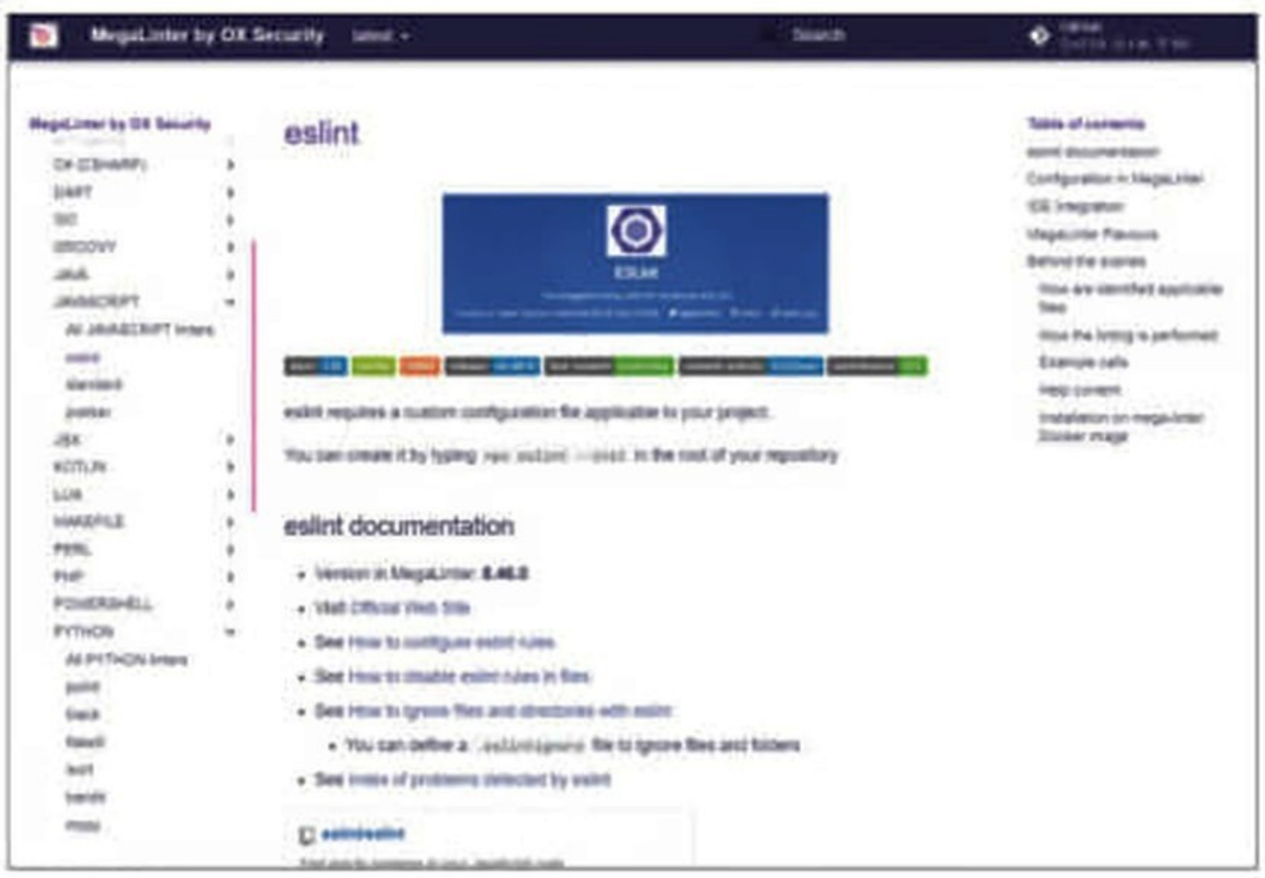


Figure 9

fichiers, présence d'un fichier de configuration, présence d'un dossier nommé spécifiquement…)

- Liste des **variables de configuration MegaLinter** relatives à ce linter
- Contenu de l'aide console (**—help**) du linter
- Liste des **extensions d'IDE** intégrant ce linter, avec lien direct d'installation
- **Indicateurs d'activité** sur le repository source du linter
- Exemples d'appels
- Instruction d'installation sur une image docker

Figure 9

Focus sur la sécurité

MegaLinter contient de nombreux linters dédiés à **contrôler la sécurité de vos sources**, cependant **comment vous assurer de la sécurité elle-même des linters** qui sont appelés avec toutes les variables d'environnement de vos jobs CI ?

Imaginez, si un seul des linters contenait un code malicieux (https://pbom.dev/techniques/?t_id=T0111), alors vos secret tokens pourraient se retrouver exposés !

Depuis sa version 7, MegaLinter appelle les linters en **masquant par défaut les variables d'environnement** les plus communément utilisées pour stocker des **secret tokens** : GITHUB_TOKEN, NPM_TOKEN, CI_JOB_TOKEN, DOCKER_TOKEN, PAT…

- Si vous avez défini d'autres variables dont le contenu est sensible, alors vous avez juste à les déclarer dans le fichier de configuration .mega-linter.yml de la manière suivante :

```
SECURED_ENV_VARIABLES: MY_SECRET_TOKEN,ANOTHER_VAR_
CONTAINING_SENSITIVE_DATA,OX_API_KEY
```

A contrario, si vous souhaitez qu'un linter ait accès à la valeur d'une variable protégée, alors vous pouvez retirer la sécurité sur cette variable uniquement pour ce linter, par exemple de la manière suivante pour le linter terraform tflint :

```
TERRAFORM_TFLINT_UNSECURED_ENV_VARIABLES:
- GITHUB_TOKEN
```

Conclusion

Vous n'avez maintenant plus d'excuse pour ne pas améliorer vos compétences et sécuriser vos projets en utilisant les meilleurs linters, formateurs et security scanners, car produire un code de qualité, robuste et sécurisé ne peut vous apporter que des bienfaits :

- Accélérateur de carrière
- Diminution des coûts de maintenance de vos applications
- Gage de qualité et de sécurité pour vos clients
- Et c'est gratuit !

La communauté MegaLinter vous attend à bras ouverts pour vous assister… et si l'envie vous prend de collaborer sur GitHub avec des développeurs brillants des quatre coins du monde, pourquoi ne pas devenir contributeur à votre tour?

Pour la suite, ça se passe par ici : https://megalinter.io/

Malwares à logique événementielle

LA MARQUE DE FABRIQUE DES CYBERATTAQUES LES PLUS SOPHISTIQUÉES, ET LE CAUCHEMAR DE LEURS VICTIMES

Natacha BAKIR

Senthorus SOC Team Lead Europe
Cyber Professional Woman Europe 2021 - Cefcys award
CEH Hall of Fame 2021 - EC-Council award
GREM certified | Kaspersky X-trained

C'est malheureusement bien connu : les logiciels malveillants (malwares) deviennent de plus en plus sophistiqués. Ainsi, bien que les entreprises disposent d'une batterie de solutions de sécurité, certains de ces malwares réussissent à contourner toutes les barrières en place. Plus inquiétant encore, lorsque les chercheurs les découvrent, ils sont parfois présents depuis des semaines, des mois, voire des années !

En 2020, les chercheurs de Sophos ont publié un article sur une attaque sophistiquée, attribuée à un nouvel attaquant: Cloud Snooper. Sa particularité ? Il est conçu pour bypasser le firewall et exfiltrer des données sensibles.

Le monde du malware est un monde qui bouge. En 2022, les systèmes de détection de Kaspersky ont découvert 400 000 fichiers malveillants par jour en moyenne, un chiffre en augmentation de 5 % par rapport à 2021.

Les groupes d'attaquants ne tarissent pas d'imagination pour faire évoluer leurs outils, créer de nouvelles techniques d'évasion, de persistance, et de self-défense.

Même si certaines de ces techniques sont bluffantes, certains malwares comme Duqu, Sofacy, ou encore Maze ransomware sortent du lot. Cloud Snooper fait partie de ceux-là. Mais comment ? Si la plupart des développeurs de malware optent pour une programmation classique, c'est-à-dire séquentielle et donc linéaire, Cloud Snooper fait partie de ces malwares qui font exception. Sa programmation de type évènementielle se définit par ses réactions aux différents événements qui peuvent se produire (callbacks).

Une programmation évènementielle ?

Les malwares ayant une logique évènementielle (event-based) ont un flux d'exécution non linéaire basé sur des fonctions de rappels (callbacks), ce qui rend leur analyse très difficile. Souvenez-vous, enfant, des jeux de labyrinthe dans les magazines, vous preniez un stylo et vous testiez les différents chemins pour rejoindre le trésor. Imaginez maintenant que ce labyrinthe soit plus compliqué que d'habitude et que certains « tips » habituels, comme l'utilisation d'une couleur, ne soient pas possibles ? Pour donner un exemple technique, si habituellement l'utilisation de "Cross-references" nous aident à y voir plus clair en mettant en lumière les relations entre les différents composants d'un système logiciel, en identifiant par exemple d'où une fonction, strings, variable, est appelée; et bien, avec la programmation événementielle, imaginez que les xrefs ne fonctionnent tout simplement pas [xrefs , la boîte de dialogue des références croisées affiche une liste de références aux différents éléments].

Et bien, nous nous trouvons dans la même situation que devant ce labyrinthe : à nos crayons pour tenter de comprendre ce que fait le malware !

Retour en arrière sur 2 malwares basés sur des logiques événementielles, ciblant Linux (ELF) et Windows (PE/EXE) pour comprendre leurs capacités.

EXEMPLE N°1 : CLOUD SNOOPER SUR LINUX

Résumé du comportement

Des paquets malveillants déguisés et qui semblent innocents entrent par le port HTTP, puis sont interceptés et réassemblés par un rootkit qui les redirige vers une porte dérobée installée sur le système.

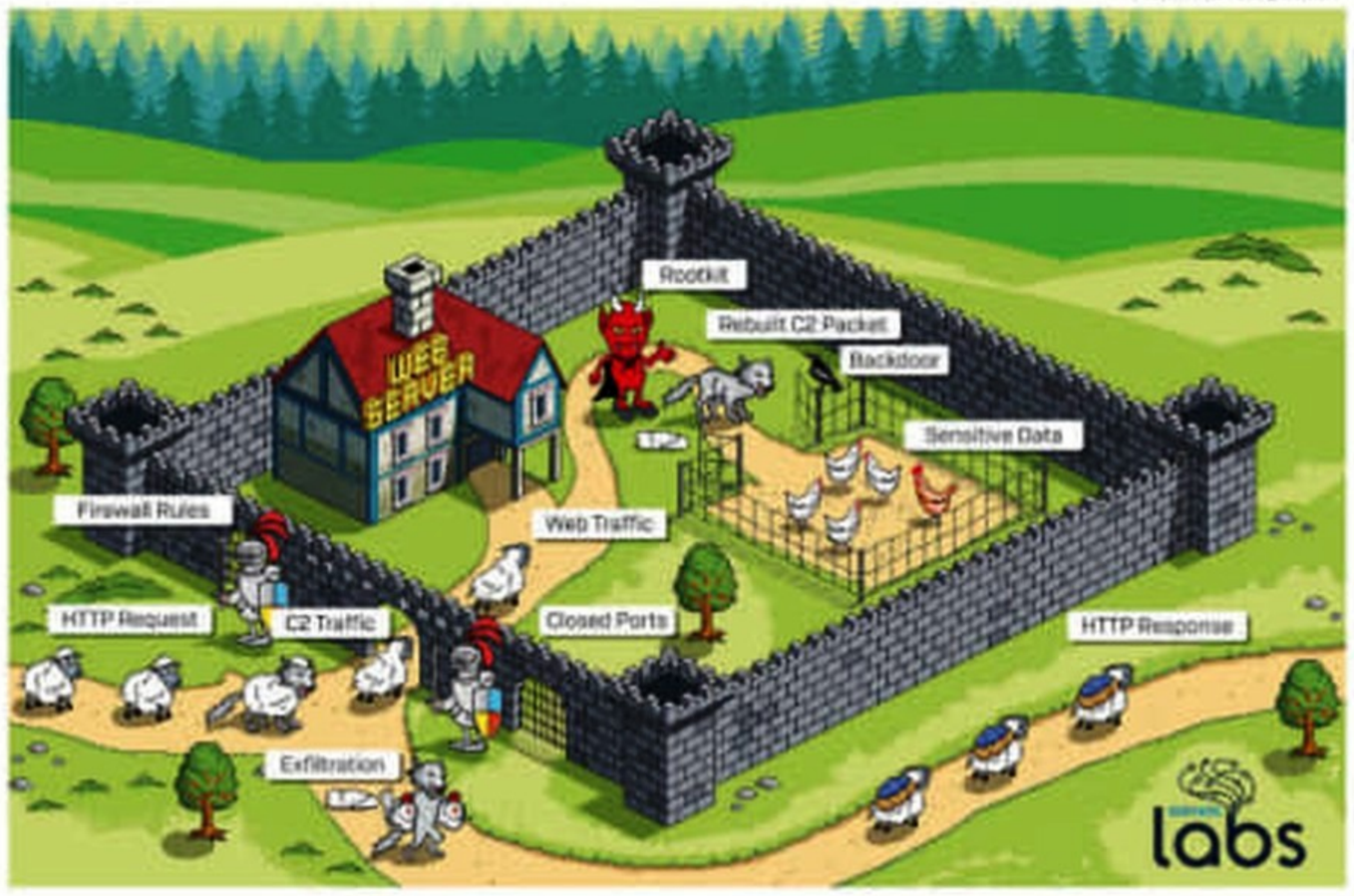
Avez-vous construit un château fort ? Dommage ! Cloud Snooper est entré par la porte ! **Figure 1**

Les chercheurs en sécurité de Sophos ont découvert qu'un système Linux compromis avec des groupes de sécurité Amazon Web Services (AWS) configurés pour n'accepter que le trafic HTTP et HTTPS entrant, écoutait toujours les connexions entrantes sur les ports 2080/TCP et 2053/TCP.

Ils ont trouvé un rootkit communiquant avec un serveur "C2", sur un trafic légitime contournant les règles du pare-feu (le serveur de commande et de contrôle C2 est un serveur distant contrôlé par l'attaquant).

Mais comment les malwares peuvent-ils utiliser des règles de trafic légitimes pour échapper précisément à ce qu'un pare-feu refuse ?

Figure 1
Source: Sophos



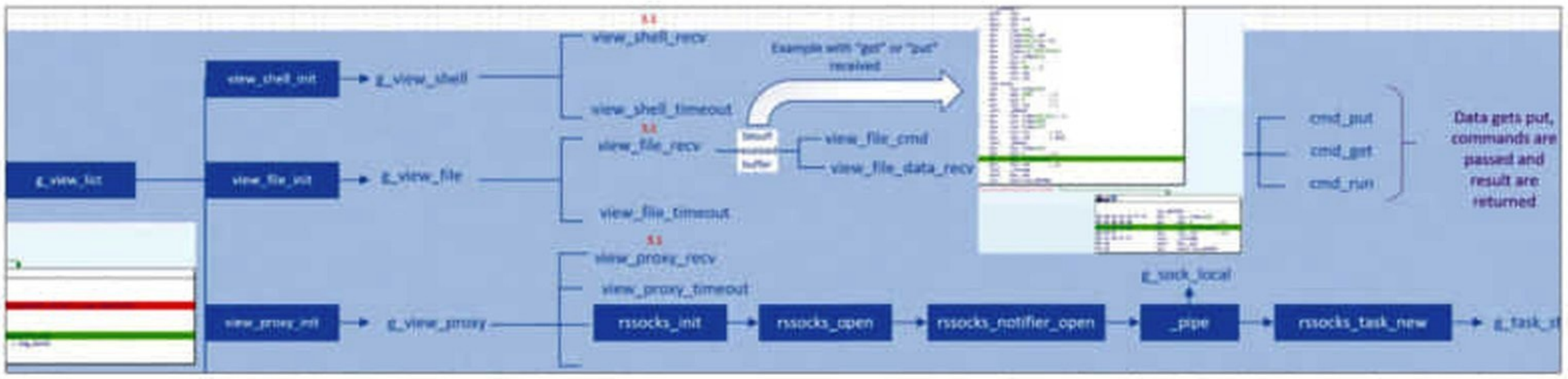


Figure 2 : *Kaspersky XTraining graph methodology*

Figure 3

IPv4 header format

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP						ECN		Total Length															
4	32	Identification																Flags			Fragment Offset											
8	64	Time To Live								Protocol								Header Checksum														
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															
(	)																																
56	448																																

Figure 4

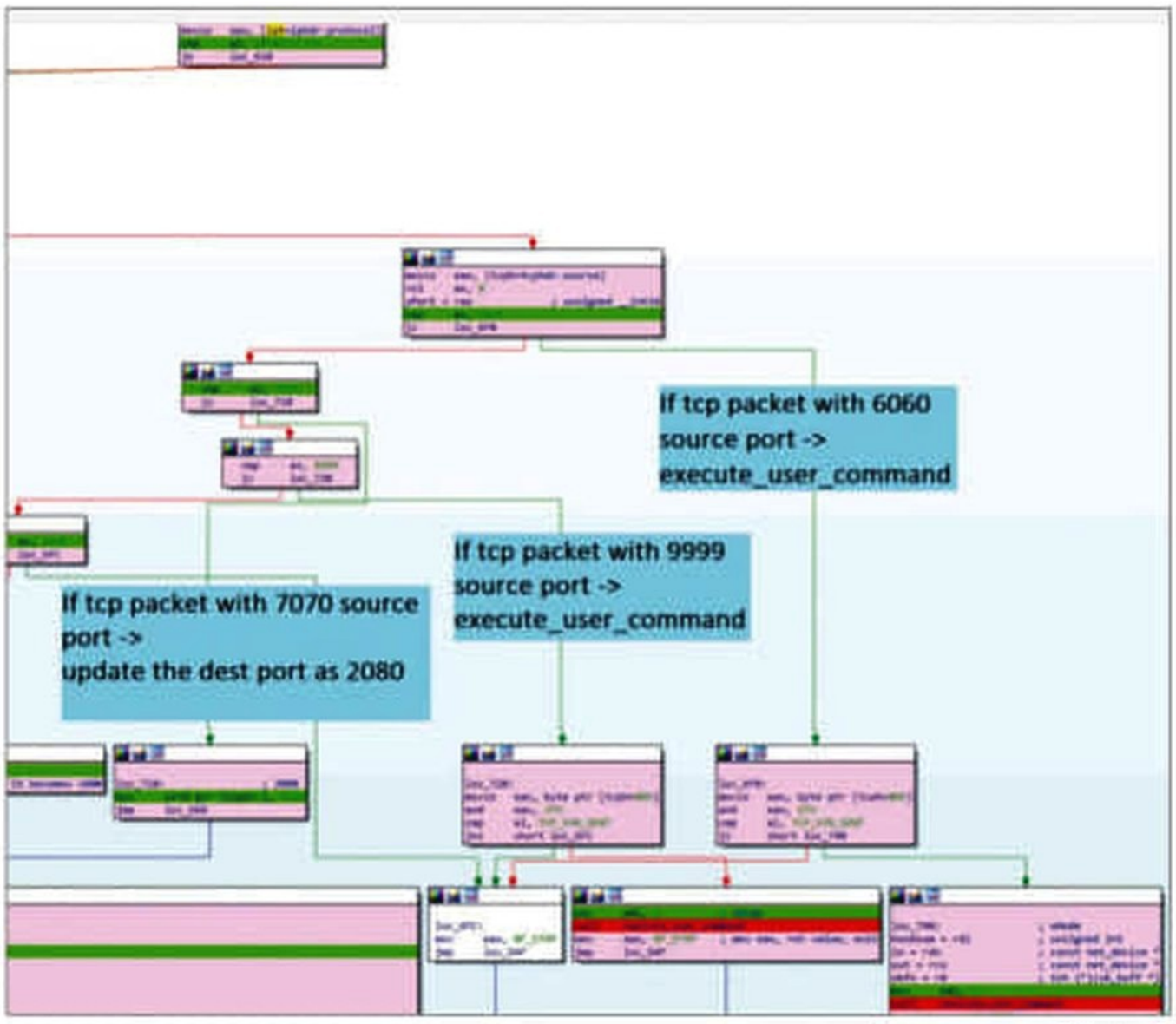


Figure 5

Architecture du malware

Cloud Snooper est un malware qui se révèle long à analyser, il faut en effet reprendre "pas à pas" son mode opératoire. Son initialisation est particulièrement longue, contenant de nombreuses fonctions de rappels (callbacks). Comme mentionné précédemment, les malwares basés sur des logiques événementielles comme Cloud Snooper ont un flux d'exécution non linéaire. Vous devez donc comprendre chacune de ces différentes fonctions avant de pouvoir comprendre ce que va faire finalement le malware, comment et quand. C'est pourquoi les malwares à logique événementielle rendent la rétro-ingénierie longue et compliquée.

Voyons un exemple ci-dessous : **Figures 2 et 3**

Ici, nous remarquons que l'action lancée sera différente selon que le malware reçoit un « put » ou un « get ».

Ainsi, la phase d'initialisation est longue et peut sembler compliquée, car en fonction de ce que le malware détecte, il déclenche une action associée.

Zoom sur la technique de contournement du pare-feu

Voyons maintenant comment l'attaquant est capable d'établir une connexion avec le client snoopy, en regardant loc_568 et loc_610. A quoi sert cette fonction ?

Nous pouvons voir qu'Ida Pro a mentionné le IHL « Internet Header Length ». Selon Wikipédia "Le champ IHL indique la longueur totale de l'en-tête IPv4, en binaire en mots de 32 bits (=4 octets). La longueur donnée correspond au nombre de lignes du schéma précédent. La valeur est comprise entre 5 et 15, car il y a 20 octets minimum et on ne peut dépasser 40 octets d'option. La longueur maximale d'un en-tête est donc de 15*4 octets soit 60 octets. » **Figure 4**

Le numéro de séquence initial est déterminé par l'expéditeur. Il crée un paquet (sk_buff) et l'envoie dans la pile de protocoles et sort sur Internet. L'en-tête TCP a une section de FLAGS à un bit. Il existe six indicateurs: URG, ACK, PSH, RST, SYN et FIN.

Fondamentalement, cette fonction analyse l'en-tête à travers différentes opérations, par exemple, pour calculer la taille de l'en-tête, on prend cette valeur IHL, on la multiplie par 4, puis on compare avec 60 qui est la valeur maximale. Et selon le type de paquet, il met à jour le port et exécute une fonction. Vous pouvez voir quelques captures d'écran du désassembleur ci-dessous. **Figure 5**

Selon le protocole détecté, UDP ou TCP, le rootkit met à jour le port de destination après que le pare-feu ait inspecté le paquet et l'ait corrigé.

Tant que le port d'origine est 60, quelles que soient les règles

du pare-feu, le paquet passe à travers le pare-feu, sans déclencher d'alerte. Il s'agit d'une technique d'évasion du pare-feu (piggybacking C2 traffic on legitimate traffic)
La fonction "interception du trafic sortant" manipule le trafic, ainsi suivant le protocole TCP ou UDP, il modifie le port source en 80 ou 53 pour contourner le pare-feu à l'aide d'un serveur Web ou une requête DNS pour qu'il soit ignoré.
Pour résumer les étapes techniques, la phase d'initialisation met en place tous les composants et fonctions de rappels qui seront utilisés par le malware une fois les connexions créées, et lance les fonctions "svc" correspondantes. Le malware s'est décompressé lui-même, puis appelle les fonctions rroot-kit_proto_recv, framework_recv et view.
En d'autres termes, étant donné que la plupart des pare-feu actuels sont des pare-feu dynamiques de nouvelle génération basés sur le réseau (stateful NextGen), ils inspectent les connexions réseaux et utilisent les ports de destination pour décider de bloquer ou d'autoriser le trafic entrant, et le port d'origine étant considéré comme éphémère.
Le logiciel malveillant inspecte le trafic HTTP et HTTPS autorisé par le pare-feu. Si le port source des paquets est 1010, 2020, 6060, 7070, 8080 ou 9999, le logiciel malveillant met à jour le port de destination du paquet et le corrige. Il redirige vers les ports 2080 et 2053. Dans la communication inverse, il réécrit le port source en 80 ou 443.

EXEMPLE N°2 : HERMETICWIPER

Résumé du comportement

Le fichier du logiciel malveillant est envoyé à la victime sous la forme d'un paquet compressé. Il crée le fichier du pilote EaseUS (drivers), qui énumère les lecteurs physiques. Le pilote est ensuite chargé et s'exécute en tant que service. Le pilote est utilisé par le biais de codes d'exécution [dwIoControl Code] pour écraser le master boot record (MBR) et redémarrer le système.

Architecture du malware : **Figure 6**

Les appels DeviceIoControl

Selon Hex-Rays, l'éditeur d'Ida Pro, « la boîte de dialogue xref affiche une liste de références aux différents éléments (Ida pro est un désassembleur). Chaque ligne possède les attributs suivants: Sens vers le haut ou vers le bas Significatif

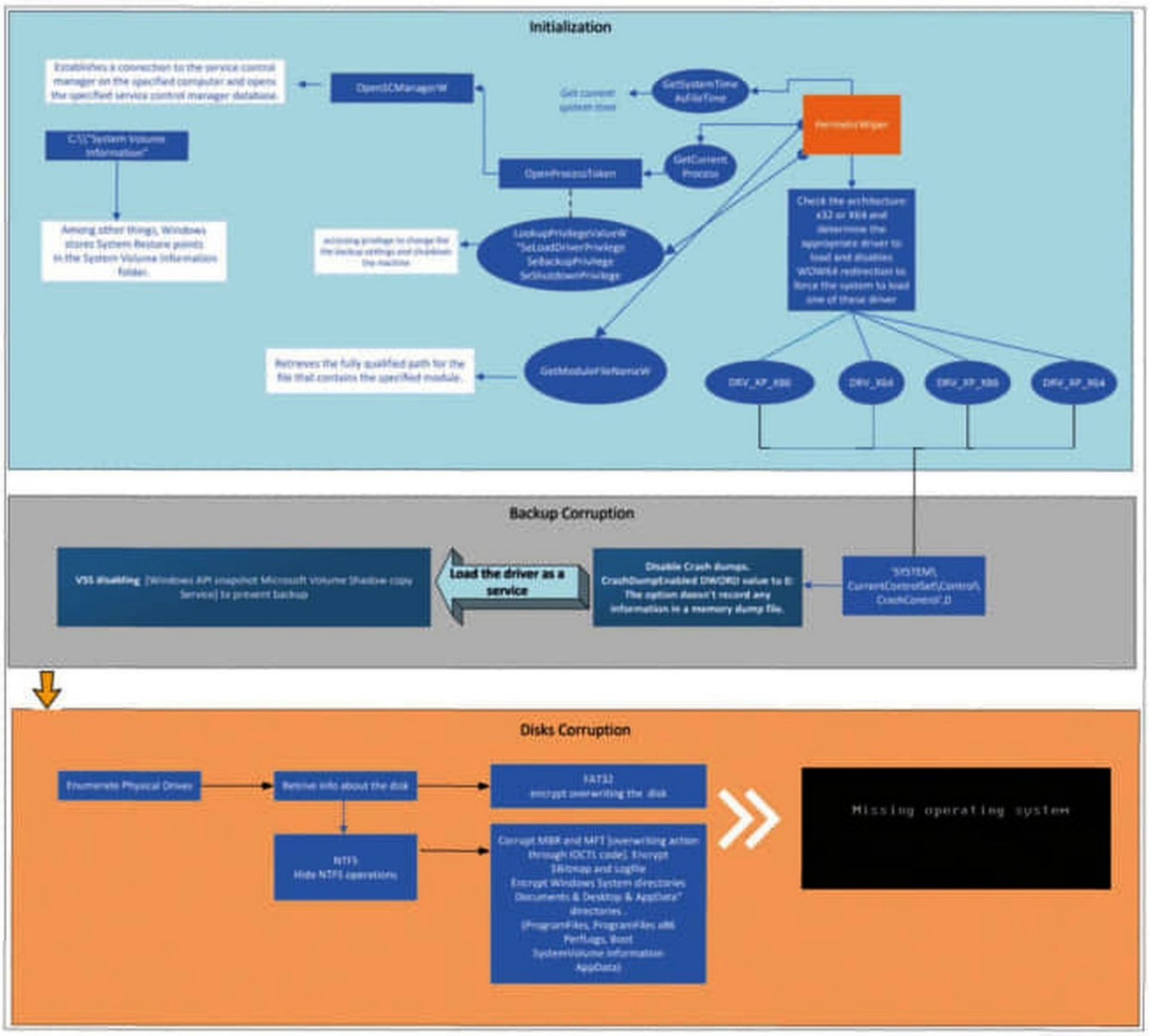


Figure 6

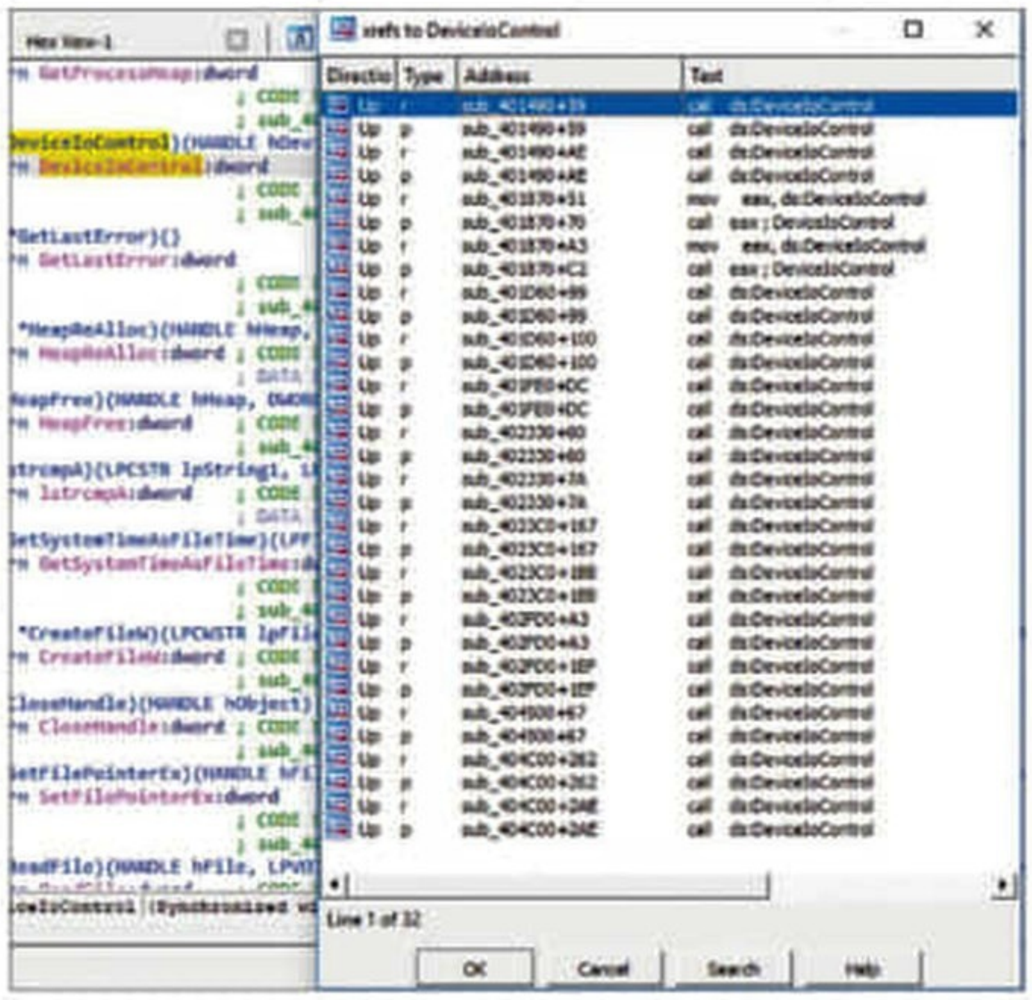


Figure 7

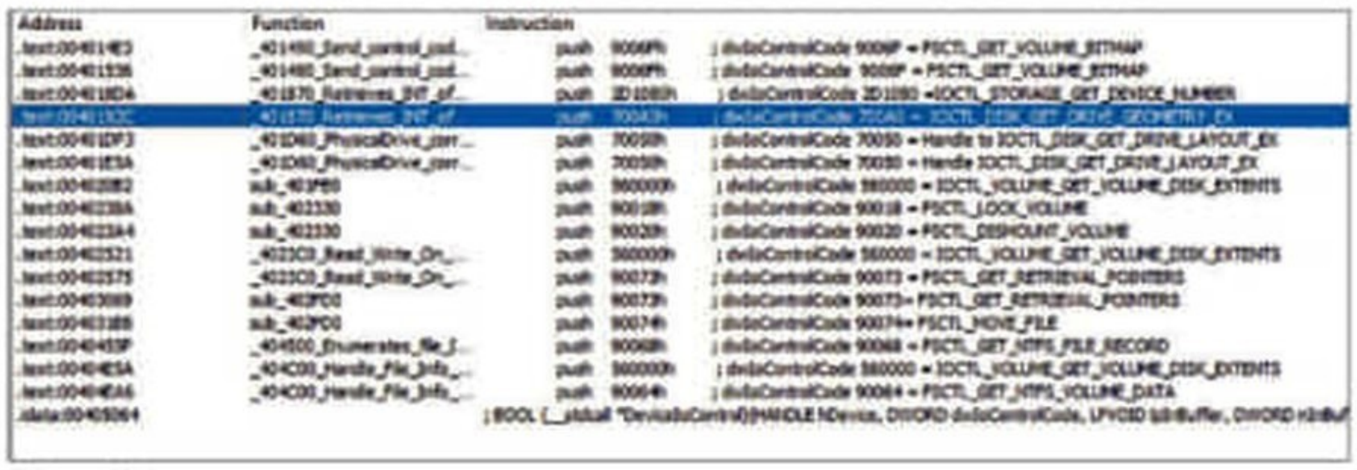


Figure 8

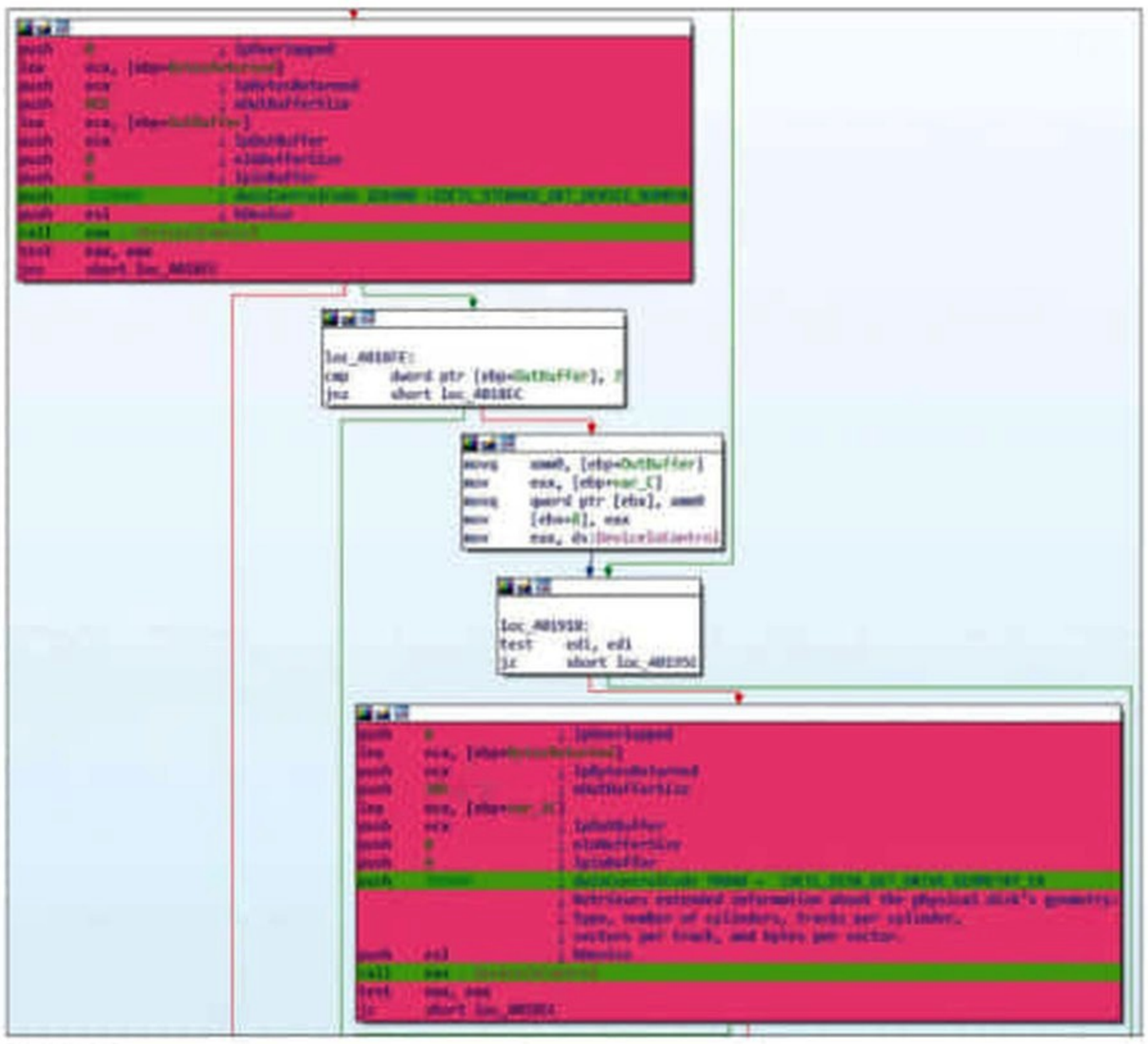


Figure 9

pour l'adresse du programme ; indique d'où vient la référence, les adresses en dessous de la cible de référence (vers le bas) ou les adresses au-dessus (vers le haut). **Figure 7**

Mais de nombreuses fonctionnalités s'appuient sur les appels DeviceIoControl avec des IOCTL spécifiques...??? Essayons de comprendre !

Selon MSDN, *"La fonction DeviceIoControl fournit une interface de contrôle d'entrée et de sortie d'appareil (IOCTL) via laquelle une application peut communiquer directement avec un pilote d'appareil. La fonction DeviceIoControl est une interface à usage général qui peut envoyer des codes de contrôle à divers appareils. Chaque code de contrôle représente une opération pour le pilote à effectuer. Par exemple, un code de contrôle peut demander à un pilote d'appareil de retourner des informations sur l'appareil correspondant ou de diriger le pilote pour effectuer une action sur l'appareil, comme la mise en forme d'un disque."*

Références Microsoft

https://learn.microsoft.com/fr-fr/windows/win32/devio/device-input-and-output-control-ioctl-

https://learn.microsoft.com/en-us/windows/win32/api/ioapiset/nf-ioapiset-deviceiocontrol

Nous pouvons remarquer que pour chaque DeviceIoControl nous avons un dwIoControlCode. La valeur associée est le code de contrôle. Les codes de contrôle d'E/S (IOCTL) sont utilisés pour la communication entre les applications en mode utilisateur et les pilotes.

Par exemple, le code de contrôle 700A0 donne le IOCTL_DISK_GET_DRIVE_GEOMETRY_EX

Selon Microsoft, le dwIoControlCode 700A0 récupère des informations détaillées sur la géométrie physique du disque : type, nombre de cylindres, pistes par cylindre, secteurs par piste et octets par secteur. Donc, pour résumer, les appels DeviceIoControl donnent des ordres définis par leurs codes de contrôle.

Liste des principaux codes "dwIoControlCode" avec leurs commandes associées répondues par l'IOCTL. **Figures 8 et 9**

Après une rétro-ingénierie un peu longue, ce malware dernière génération est désormais compréhensible. Nous avons vu ensemble, comment ces malwares intelligents peuvent challenger notre sécurité et notre esprit d'analyse. Analyser ce type de malware n'est pas seulement passionnant, c'est également très utile. En effet, au-delà de comprendre leurs techniques d'évasion, cela permet de créer des règles de détection pour stopper ces malwares, mais aussi pour tenter d'arrêter leurs futurs variants.

Les rapports d'analyse de malwares sont consultables sur https://www.senthonus.ch/threat-analysis-blog

et https://www.malwaredissection.com

Améliorer la production et la sécurité du code avec Amazon Code Whisperer

Max Thomas

Principal Partner SA chez Amazon Web Services (AWS)

Max THOMAS est Principal Partners Solutions Architect chez AWS depuis 5 ans et demi et milite pour l'emploi de la philosophie DevOps dans les entreprises. Il fait partie du groupe d'expertise sur le Serverless chez AWS et présente régulièrement des présentations au Summit AWS en France. Il aide les sociétés de services partenaires d'AWS à cranter leur expertise sur la modernisation des applications.

Dans cet article, nous découvrirons comment Amazon CodeWhisperer peut aider tous les développeurs à développer plus vite et sécuriser efficacement du code en utilisant la puissance des nouveaux modèles d'intelligence artificielle que l'on nomme la Generative Artificial Intelligence (GenAI). Pas besoin d'être un gourou d'AWS ou un spécialiste sécurité pour faire la revue de votre code ! AWS vous met à disposition un robot qui suggère le code et vous dit où améliorer la sécurité. Ouvrez les IDE, c'est parti !

Vous développez une application ? Combien de temps cela vous prend de coder de manière efficace ? Comment vérifier que vous avez mis en place les bonnes pratiques en termes de sécurité ? Si vous faites comme moi, vous allez chercher des réponses sur les forums d'entraide tels que *Stack Overflow* ou *AWS re:Post*, vous allez consulter un repository *Github* et son gist collector. Les plus avancés feront de *l'extreme programming* et des *peer reviews*, afin d'améliorer la production initiale et respecter les standards. Cela prend du temps, alors que vous êtes déjà surchargés. Et de manière assez ironique, la solution résidera parfois dans le copier/coller de 4 lignes de code après avoir cherché et testé deux heures. Et si je vous disais que vous pouviez avoir un assistant qui analyse ce que vous tapez dans l'IDE et qu'il vous fait des recommandations en temps réel afin d'économiser ce temps dont vous ne disposez pas ?

La *Generative AI*, ou encore intelligence artificielle générative, offre une technologie à la pointe permettant d'implémenter de nouveaux cas d'usage. Cela s'applique à différentes tâches comme le résumé automatique et la détection de plagiats, ou encore des agents conversationnels spécialisés. Pour les développeurs, cette technologie prend une nouvelle dimension. Par exemple, en tant que développeur je suis amené à utiliser des librairies, des APIs, des SDK que je ne connais pas parfaitement. La prise en main peut être difficile parce que je ne suis pas un expert *Kotlin* ou bien parce que la documentation est tentaculaire.

Dans la suite, j'aborde en surface les concepts fondamentaux de la *GenAI*, comment sécuriser les applications dans le Cloud AWS et comment utiliser *Amazon CodeWhisperer*.

La Generative AI, c'est quoi ?

La *GenAI* est un type de système d'intelligence artificielle qui est capable de créer du contenu texte, image ou d'autres médias en ayant appris les structures de données qui sont propres à chaque média. Il existe deux principales approches que sont les réseaux antagonistes génératifs (GAN) et les transformateurs génératifs pré-entrainés (GPT). Afin que les algorithmes GPT fonctionnent efficacement, il leur faut ingérer beaucoup de données au départ. Ils ont ainsi besoin de larges modèles de langages (LLM) pour leur permettre de générer du contenu avec une valeur ajoutée.

AWS a annoncé en Avril 2023 l'arrivée d'*Amazon Bedrock*. Ce service managé met à disposition de nombreux algorithmes à l' état de l'art, comme Amazon *Titan ou Claude 2 d'Anthropic. Ce service proposé à l'usage permet* de démocratiser l'accès à ce type de modèles.

Amazon CodeWhisperer, quant à lui, a constitué son propre modèle de langage, en ayant absorbé de larges quantités de code source déjà écrit et vérifié par des professionnels. AWS a alimenté le modèle du service avec la documentation des autres services AWS, les architectures d'exemples et le code afférent afin, entre autres, de construire une expérience optimale pour les développeurs. En bonus, ont été intégrées un ensemble de ressources qui alimentent le contexte, telles que la documentation des langages de programmation, le contenu des forums pour développeurs, et beaucoup d'autres ressources. *Amazon CodeWhisperer* atteint ainsi un niveau de suggestion pertinent pour Java, Python, JavaScript, TypeScript et supporte la génération de code pour Ruby, Go, PHP, C+, C, Shell, Scala, Rust, Kotlin et SQL.

Avant de se lancer dans le code, parlons de ce que l'on entend par la sécurité dans le cloud AWS.

La sécurisation dans le Cloud AWS

Le Cloud AWS est conçu sur une sécurité qui repose sur un modèle de responsabilité partagée. AWS s'occupe de la sécurité de l'infrastructure physique des centres de données, du

réseau, du stockage et des services managés. On appelle cela la sécurité *du* cloud. Les clients sont responsables quant à eux de sécuriser leurs applications en se basant sur leur propre outillage ou en utilisant les services d'AWS. On parle alors de la sécurité *dans* le cloud.

Les bénéfices du Cloud AWS sont bien d'accélérer ces phases d'intégration d'outillage en proposant des *services managés* qui sont utilisables sans délai, et réduisent drastiquement les efforts de maintien en condition opérationnelle. AWS propose plusieurs services qui permettent de faire remonter les sujets d'attention en termes de sécurité, en utilisant de l'intelligence artificielle. J'en détaille quelques-uns dans la suite.

Amazon Inspector permet de scanner de manière continue et automatisée les vulnérabilités de vos applications, qu'elles soient hébergées sur des VM *Elastic Cloud Compute* (EC2), des conteneurs via *Amazon Elastic Kubernetes Service* (EKS) ou *Amazon Elastic Container Service* (ECS), du code dans des fonctions *AWS Lambda*. C'est très pratique car à l'heure des architectures distribuées, il est presque impossible de tout contrôler sans une forte automatisation.

Un autre service, *Amazon CodeGuru*, effectue l'équivalent d'une revue de code systématique des commits, tels que le feraient un développeur chevronné. Le service identifie les vulnérabilités présentes dans le code, propose directement une solution de remédiation pour l'utilisateur. Par exemple, si j'ai commité un mot de passe par erreur, il génèrera immédiatement une alerte.

Amazon DevOps Guru quant à lui analyse les métriques et configuration des applications pour détecter les comportements anormaux et déceler les problèmes de performance ou de sécurité. Un bon exemple est l'utilisation particulièrement élevée d'un cluster de base de données, qui pourrait potentiellement être le signal qui marque le début d'un vol de données par un attaquant.

Amazon CodeWhisperer arrive en amont de toutes ces phases puisque ce service propose un plugin pour votre environnement de développement intégré et récupère des suggestions de code ou des résultats de scan de votre application. C'est important car au final, d'une mauvaise construction d'IaC (Infrastructure as Code) peut découler de nombreux problèmes lors de la construction des artefacts, des tests, du déploiement et du maintien en condition opérationnelle. Fondamentalement, AWS propose aux développeurs des services qui visent à réduire drastiquement les risques d'erreur humaine. La mise en place de cette approche permet d'augmenter la qualité et la robustesse du code produit. *Amazon CodeWhisperer* en est un parfait exemple. Nous avons mesuré lors d'une expérience interne, que ce service pouvait améliorer la vitesse de développement de 57. Pour utiliser le service, c'est très simple : une fois le plugin installé, vous tapez en langage naturel ce que l'application doit faire et le plugin va chercher des suggestions de code et intègre le résultat là où se trouve votre curseur.

Installation

Rentrons dans le vif du sujet, pour démarrer, il faut installer le plugin dans votre environnement de développement intégré (IDE en anglais). Parmi les IDE disponibles, on note *Visual Studio Code* (VSC), les IDEs de l'éditeur *JetBrain* (par exemple IntelliJ et PyCharm), *AWS Sage Maker Studio* avec une intégration dans les notebooks, *AWS Cloud9* qui est un service proposant une interface de développement dans le navigateur avec un stockage du code sur AWS, et *AWS Lambda*, qui dispose d'une interface de développement dans la console AWS.

Dans la suite je détaille l'installation et la configuration dans VSC sous Windows, leader des IDEs en termes d'usage. Dans ce cadre-là, vous n'avez pas besoin d'un compte AWS pour bénéficier des recommandations de CodeWhisperer, il suffit d'un AWS Builder ID.

Commençons par le site du service https://aws.amazon.com/codewhisperer/.

1. Cliquez sur le bouton *Use CodeWhisperer for free* qui vous amènera à page *Get Started*.
2. Cliquez sur le lien vers la marketplace de votre IDE ou installez le plugin *AWS Toolkit* dans l'IDE. **Figure 1**
3. Une fois installé, cliquez sur l'icône du plugin et sélectionnez dans la liste des *Developer Tools CodeWhisperer*.
4. Cliquez sur *Start* et sélectionnez la première option *Use a personal email to sign up and sign in with AWS Builder ID*. **Figure 2**
5. Copier le code. **Figure 3**
6. Si vous n'avez pas de *Builder ID*, inscrivez-vous, cela prend quelques instants.
7. Dans le navigateur, collez le code et cliquez sur *Next*. **Figure 4**
8. Autorisez le plugin *AWS Toolkit for VSCode* à accéder et envoyer vos données. **Figure 5**
9. Une fois la procédure validée, le service de suggestions automatiques est démarré.

Pour la suite, j'ai creusé deux sujets à fort impact pour les développeurs : la production de logiciel et la sécurité.

Accélérons la production de code

Dans ce premier exemple, nous allons monter un projet d'IaC avec *AWS Cloud Development Kit* (CDK) qui est le framework logiciel permettant de générer du code *AWS CloudFormation* à l'aide de langages de programmations orientés objets modernes. En ce qui me concerne, je suis plus à l'aise avec Typescript mais le service supporte aussi JS, Python, Java, C# et Go.

```
//Installation de Typescript et AWS CDK
$ npm install -g typescript aws-cdk

//Création du répertoire
$ mkdir code-whisperer && cd code-whisperer

//Initialisation d'une app CDK en TypeScript
$ cdk init app —language=typescript
```

CDK génère une structure complète de projet et nous travaillons dans le répertoire *lib*, éditant dans VSC le fichier *code-whisperer-stack.ts*. Dans la logique de CDK, il convient de déclarer toutes les ressources qui composent notre infrastructure, pour réutiliser réutilises ces mêmes ressources entre piles applicatives, ou encore renforcer les contrôles avant génération du template *AWS CloudFormation*, bénéfice

d'un langage fortement typé comme Java ou TS.
Et maintenant, il suffit de commencer à taper pour que la magie opère. *Amazon CodeWhisperer* propose du code complémentaire et pour l'appliquer, il suffit de taper sur la touche *TAB*. Essayons :

```
new lambda
```

Le code proposé est :

```
new lambda.Function(this, 'CodeWhispererFunction', {
```

Figure 7

Acceptons cette proposition avec *TAB* et allons à la fin de la ligne, appuyons sur *Entrée* puis *Tab* de nouveau. Le code proposé est :

```
runtime: lambda.Runtime.NODEJS_14_X,
```

Ca fonctionne. VSC râle car il ne résout par la dépendance amenée par le code et la mention de lambda. Remontez en haut dans les imports et tapez *import*. Le code proposé est :

```
import * as lambda from "aws-cdk-lib/aws-lambda"
```

Revenons dans les paramètres de notre Lambda à la fin de la ligne *runtime*. Tapez entrée puis tab plusieurs fois.

```
code: lambda.Code.fromAsset('src'),
handler: 'code-whisperer.handler',
```

On rajoute le nom de la fonction :

```
functionName: "code-whisperer.handler"
```

On a donc créé une fonction lambda en quelques secondes, tapé en soi 6 charactères et quelques *Entrée/TAB* qui ont généré 6 lignes de code et 270 caractères. Pour vérifier que le code TypeScript fonctionne, dans le terminal de VSC :

```
//Création de l'asset de la fonction lambda
$ mkdir src && echo "" > src/index.js

//Syntétisation du template CloudFormation
$ cdk synth
```

Figure 8

CDK affiche le template généré par le code de la pile applicative, soit 321 lignes pour 5906 caractères. Ces 6 lignes constituent le meilleur investissement de ma journée.

Sécurisons notre code

Pour ce deuxième exemple, je prends un contexte différent. Mon application en Python tourne déjà sur AWS et j'utilise la librairie *AWS SDK pour Python* dénommée *boto3* pour interagir avec la plateforme AWS.
La démarche est cette fois différente, le plugin *AWS Toolkit for VSCode* va envoyer une demande de scan à *Amazon CodeWhisperer* qui utilise le contenu du fichier en cours d'édition dans l'IDE. La fonctionnalité de scan est intégrée à *Amazon CodeGuru* qui inclut le moteur d'analyse et permet de remonter les problèmes identifiés dans le panneau *Problems* de VSC.

1. On crée cette fois un nouveau projet dans VSC et on crée un fichier *main.py* avec un code assez simple :

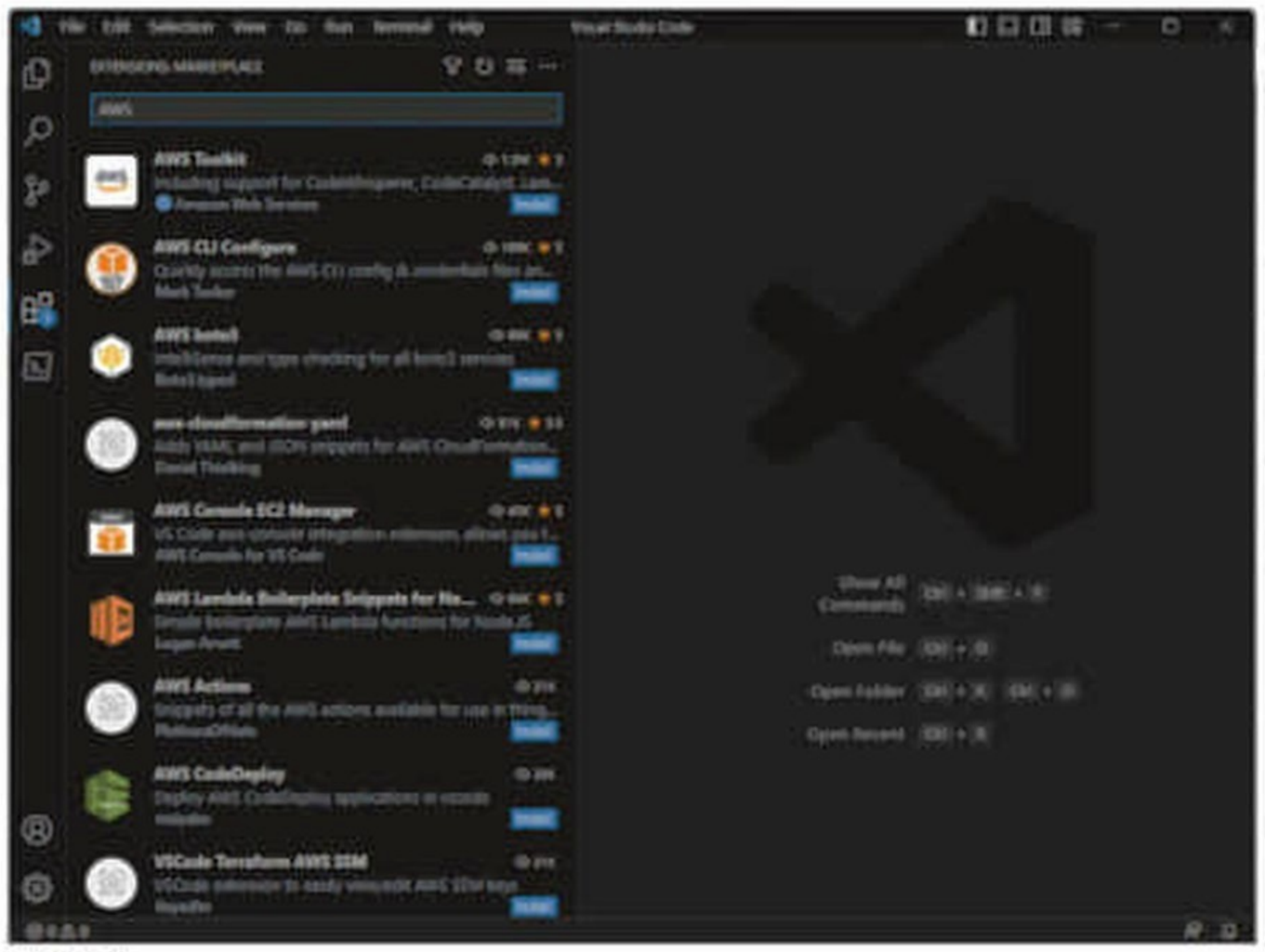


Figure 1

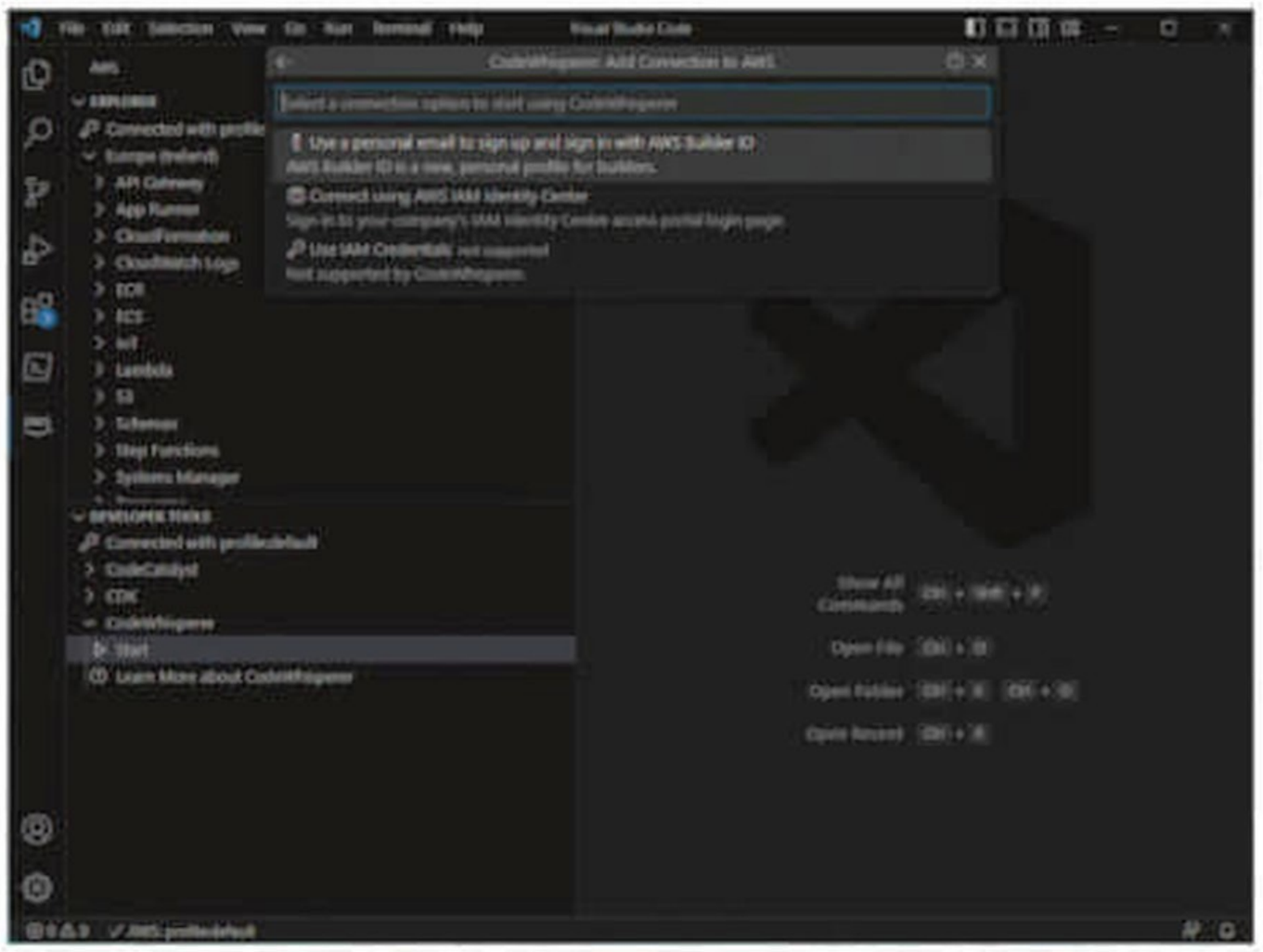


Figure 2

Figure 3

Figure 4

Figure 5

Figure 6

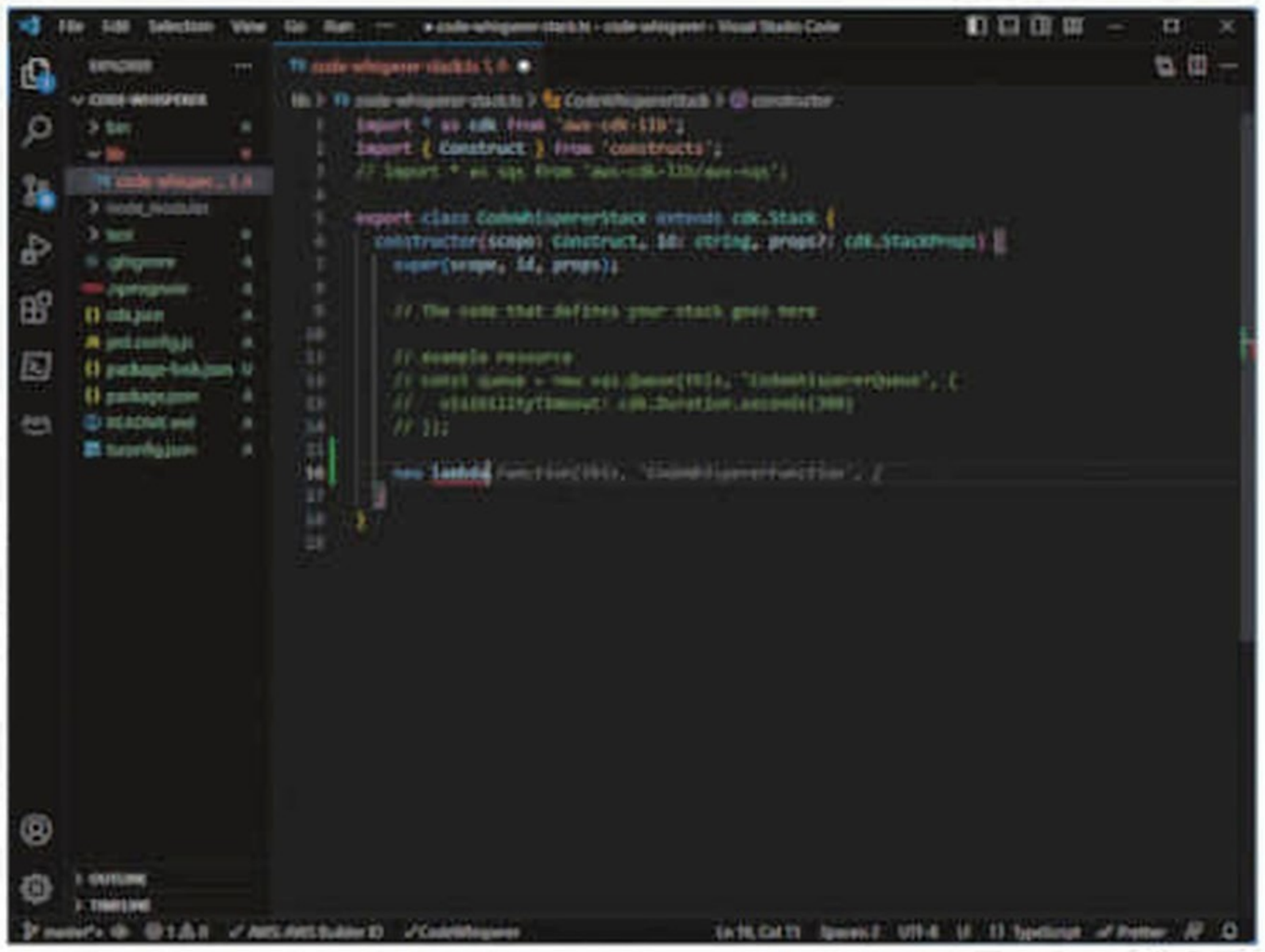


Figure 7

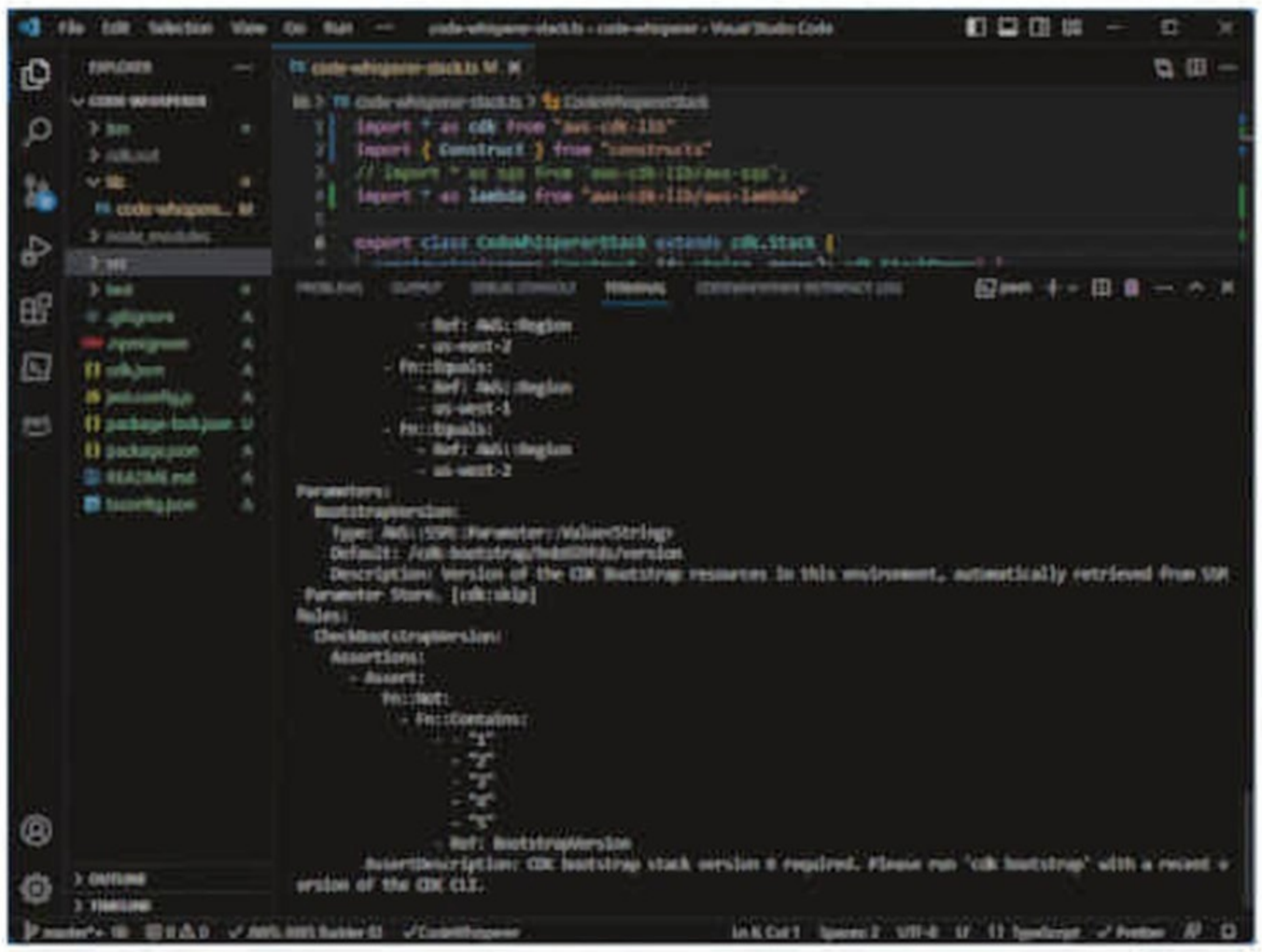


Figure 8

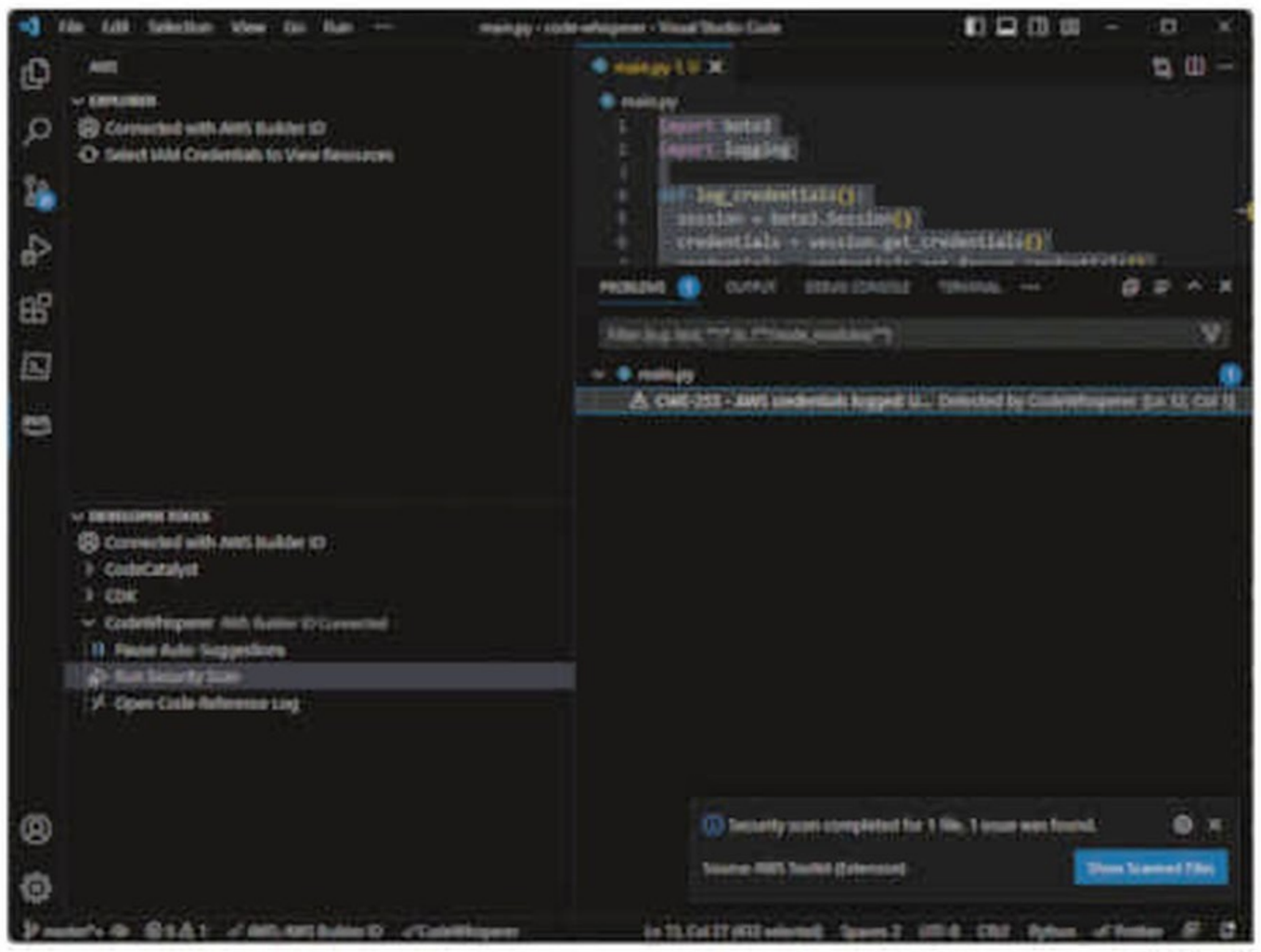


Figure 9

```
import boto3
import logging

def log_credentials():
 session = boto3.Session()
 credentials = session.get_credentials()
 credentials = credentials.get_frozen_credentials()
 access_key = credentials.access_key
 secret_key = credentials.secret_key
 session = boto3.Session(aws_access_key_id=access_key,
                         aws_secret_access_key=secret_key)
 logging.info(access_key)
 logging.info(secret_key)
```

2 Cliquez sur le logo AWS pour accéder à l'interface du plugin, sélectionnez *Developper Tools* > *CodeWhisperer* et *Run Security Scan*

3 Le scan se lance et peut prendre jusqu'à 60 secondes le temps de procéder à l'analyse du code et de le confronter à la base de vulnérabilité d'*AWS CodeGuru*. **Figure 9**

4 Bingo ! *Amazon CodeWhisperer* remonte que ce n'est pas une bonne pratique de logger ses identifiants de connexion. Dans *Problems*, la *CWE-225* apparait et si vous cliquez dessus, l'IDE vous amène sur la ligne qui pose problème.

En quelques instants, nous avons reçu une revue de code sans quitter l'interface de développement, sans consulter les best practices d'AWS sur l'intégration de boto3, et sans se connecter à la console AWS. Et la recommandation vient avec des propositions de solutions qui permettent au développeur de changer son code au moment où il l'écrit.

Intégration native dans AWS Lambda

Amazon CodeWhisperer propose également d'utiliser l'intégration standard prévue dans AWS Lambda. Pour cela, il suffit de :

1 Se connecter à la console, choisir la région US East 1 (North Virginia) et aller dans la fonction AWS Lambda concernée.

2 Dans l'éditeur, identifier l'icône avec une roue dentée et cliquer dessus pour accéder au panneau des préférences.

3 Sélectionnez l'entrée de menu dénommé *Key bindings*

4 Vérifier que le key binding par défaut est bien activé : *Alt* + C pour Windows ou *Option* + C pour Mac. **Figure 10**

Coût

Et voici la bonne nouvelle, c'est gratuit pour les développeurs individuels qui utilisent le service depuis leur éditeur avec un AWS Builder ID. Par défaut la version gratuite est limitée à 50 scans par mois, de quoi faire deux scans par jour ouvré si vous développez en mono repo. Et si on intègre Amazon CodeWhisperer dans un pipeline CI/CD, en appelant après les tests l'api pour un scan, on peut créer un simple système de DevSecOps. Pour ceux qui utilisent un compte AWS, *Amazon CodeWhisperer* propose une Licence Professionnelle à 19 USD par utilisateur par mois pour un usage intensif en entreprise, à travers des *AWS Organizations*, et qui lève la limite à 500 scans par mois/utilisateurs.

Développement Durable

Un autre aspect du service qui est différenciant est la reconnaissance des erreurs ayant un impact sur le développement durable ou encore des biais associés dans le code. Concrètement, le service d'*Amazon CodeWhisperer* va utilise des technologies d'intelligence artificielle et de machine learning pour détecter et réduire les recommandations qui ne seraient pas pertinentes voire qui constitueraient des anti-patterns au regard d'un impact long terme sur le développement durable.

AWS propose à travers son *Well Architected Framework* un ensemble de livres blancs dont celui sur la *Sustainability*. Le développement durable dans le cloud se base sur deux aspects que sont l'élimination des ressources inutiles, d'une part, et le fait de tirer le maximum partie des ressources en place, d'autre part. Les pistes d'améliorations sont multiples. Par exemple, Amazon CodeWhisperer peut analyser les infrastructures décrites dans CDK et proposer par défaut des *snippets* optimisés pour la gestion d'instances EC2 avec les familles d'instances recommandées et les architectures optimales car il a accès au contexte de l'application qui est décrite.

De manière un peu plus avancée, Amazon CodeWhisperer peut par exemple proposer des services managés pour alléger l'effort de maintenance d'une application, avec l'implémentation correspondante. Dans mon code CDK, je suis en train d'implémenter un service basé sur EC2 donc l'objectif est d'orchestrer et d'exécuter de nombreuses tâches programmées. Basé sur mon code, la recommandation sera d'utiliser un service managé, *AWS Batch*, avec un déclenchement asynchrone, ce qui permet d'économiser une machine qui « attendrait » sans rien faire et consommerait de l'énergie sans avoir rien à produire.

Ce service analyse aussi le contexte applicatif s'il est partagé dans des fonctions et permet de proposer des fonctions natives au langage de programmation plutôt que des implémentations effectuées par le développeur. Disons que je réalise un tri sur les clés d'un tableau associatif en PHP, créant de nouvelles variables et copiant la donnée, Amazon CodeWhisperer recommandera l'emploi d'une fonction native équivalente, *ksort*, directement incluse dans le runtime PHP et qui consommera moins de cycles CPU.

Gestion des licences Open Source

Amazon CodeWhisperer intègre une fonctionnalité qui permet d'identifier ce qui ressemble à du code open source.

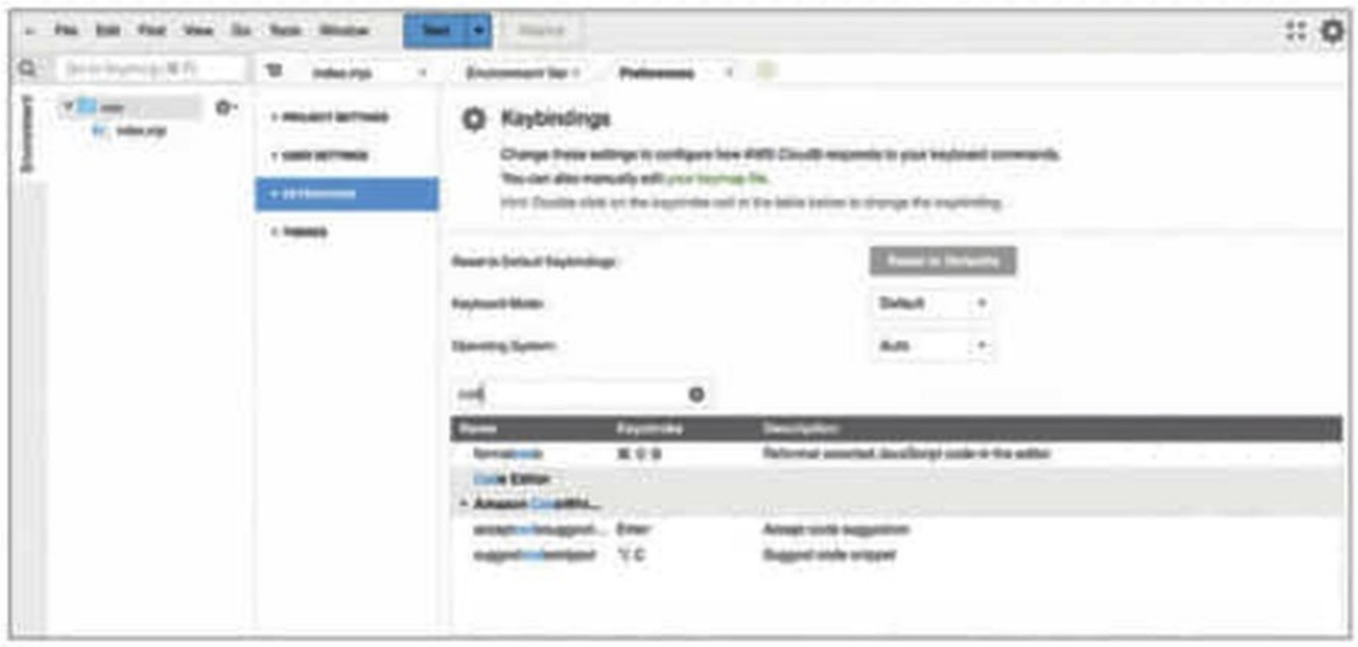


Figure 10

Cela permet typiquement de gérer les problématiques liées à la propriété intellectuelle du code ou encore d'intégrer les licences Open Source correspondantes.

Un bon exemple est celui d'un développeur qui publie un projet open source incorporant des dépendances à des projets tiers. Les dépendances sont identifiées et historisées dans le plugin, ce qui permet de vérifier s'il y a un changement de licence et qu'en faisant une mise à jour d'une dépendance, on se retrouve à utiliser du code propriétaire, soumis à un modèle commercial, par exemple. Amazon CodeWhisperer va comparer votre code pour identifier si le code contient tout *ou partie* de code d'un autre logiciel soumis à licence et propose au développeur d'intégrer la licence à son *README.md*.

Et pour aller plus loin ?

Sous l'angle sécurité, on pourrait intégrer *Amazon CodeGuru Reviewer* dans une chaine de CI/CD et analyser de manière incrémentale le code pour empêcher du code problématique ou qui crée malencontreusement une vulnérabilité. Un moyen simple et automatisé d'avoir du *DevSecOps*.

Sous l'angle de la production logicielle, cela vous libère une bonne partie de votre temps pour faire d'autres choses et notamment innover. Comme on ne sait pas ce que l'on ne sait pas et que l'on a dans les mains un outil qui nous dit quoi faire et où chercher, c'est génial, ça permet de se régaler techniquement avec tous les services de la boite à lego d'AWS et d'aller très vite en construisant des nouvelles choses. Le modèle va s'enrichir rapidement vu sa popularité et cela permettra d'avoir des recommandations encore plus fines et pertinentes dans le futur. A suivre.

François Tonic
Rédacteur en chef de Programmez !

JobTeaser : la sécurité et le contrôle sur les données de bout en bout

JobTeaser est une plateforme de recrutement et d'orientation des jeunes talents implantée dans 25 pays. L'éditeur emploie plus de 400 personnes. L'idée de départ du service est que la transition entre le monde étudiant et le monde professionnel s'opère souvent difficilement. JobTeaser propose un service complet de recherche de stages et d'emplois pour les étudiants et les jeunes diplômés. Plus de 750 écoles et universités l'utilisent, totalement plus de 5 millions d'étudiants et de jeunes diplômés. Dans ce contexte, la sécurité est critique et particulièrement sur les données personnelles des étudiants. Benjamin Aubertel, CTO et Axel Clément, CISO, reviennent pour Programmez! sur cette quête du secure by design.

Benjamin Aubertel
CTO chez JobTeaser

Axel Clément
CISO chez JobTeaser

JobTeaser adresse trois types d'utilisateurs : les étudiants, les écoles et les entreprises. « *Un des challenges est la sécurité des données. Ces données incluent des données personnelles de l'étudiant. Nous les chiffrons par défaut. La CNIL est très sensible à cette problématique. Récemment, le data privacy framework est sorti au niveau européen, que nous devons intégrer. Autre sujet sensible : le contrôle sur les données. La sécurité est dans notre ADN et elle évolue constamment.* » explique Benjamin Aubertel.

Une infrastructure cloud basée sur AWS

Côté front, JobTeaser s'appuie sur des apps en React 18.x. L'UI est une SPA. Les équipes utilisent gRPC et Ruby. « *Ruby est un langage très utilisé dans notre secteur. Nous regardons aussi le langage Go* » insiste Benjamin Aubertel. Sur la partie SPA, une migration sur NextJS est en cours. Côté base de données, la plateforme utilise PostgreSQL. Cette base est reconnue pour sa robustesse, sa capacité de gérer des grands volumes et sa montée en charge.

Le tout est déployé dans une centaine d'instances EC2. Pour l'architecture, la plateforme est en microservice et 100 % Kubernetes. Mais le choix a été fait de maîtriser totalement les clusters, Kubernetes est directement déployé sur les instances

Sur la partie sécurité, les équipes se basent sur plusieurs approches et couches. « *C'est important que le cloud ait des designs de sécurités par défaut comme le chiffrement de bout en bout. Nous utilisons aussi Terraform pour provisionner l'infrastructure. Cela nous permet de faire du secure by design, d'avoir un code standard.* » poursuit Axel Clément.

À cela se rajoute l'utilisation de référentiels reconnus tels que l'OWASP pour surveiller les tendances des failles pour pouvoir modifier les codes si nécessaire.

Des clés de chiffrement sous le contrôle de JobTeaser

Pour rassurer ses clients, JobTeaser a mis en place une architecture lui donnant le contrôle sur les clés de chiffrement. La plateforme utilise un produit de gestion des clés fourni par Thalès et utilise la fonctionnalité External Key Store (XKS) fournie par AWS. XKS permet à JobTeaser d'externaliser des clés de chiffrement, là où le service Thales est utilisé pour la gestion des clés.

« *Cette segmentation nous permet de séparer les fonctionnalités et les droits. Nous utilisons un VPC isolé pour la connexion avec le tiers de confiance et la partie HSM est du côté de Thalès. Par contre, nous restons responsables des clés. Cela signifie que ni AWS ni Thalès ne peuvent les utiliser.* » explique Axel Clément.

« *Le nouveau type de pattern qu'on utilise s'appelle Host You Own Key et répond à notre besoin de rassurer nos clients sur le fait que nous, JobTeaser, avons un contrôle suffisant sur la protection des données personnelles que nous hébergeons. Avoir le contrôle sur le processus de chiffrement de nos données permet cela.* » Ajoute Benjamin Aubertel.

Quid des performances de la plateforme avec ces différentes couches de sécurité ? « *Le recours à XKS n'impacte pas les performances de la plateforme. C'est transparent pour l'utilisateur, mais aussi pour le développeur. Car il continue à voir la partie KMS (NDLR: le service de gestion de clés d'AWS) pas le tiers de confiance. Ces problématiques sont du côté des équipes infrastructure et sécurité.* » résume Axel Clément.

Le volume de données chiffrées est d'environ 1,5 – 2 To.

Une des principales difficultés est que chaque établissement scolaire utilise son propre outil ou protocole de connexion et d'authentification. Pour faire simple : la connexion à une école lui est propre. Il faut donc implémenter chaque école dans le système. « *Si le SSO n'est pas utilisé (par l'école), il faut proposer une authentification ou intégrer le système de l'école. Cependant, nous utilisons les standards et réglementations en vigueur pour ne pas réinventer le choix* » poursuit Axel Clément.

Sur la partie API, JobTeaser se prépare à un déploiement un API Gateway pour mieux filtrer et contrôler les connexions entrantes et sortantes. Pour prévenir une attaque massive de type déni de services, un fournisseur externe est utilisé. « *C'est un choix d'architecture* » précise Benjamin Aubertel.

Pour éprouver la plateforme, du pentesting est réalisé. « *Un pentest classique est fait tous les ans. Un second pentest orienté fuite de données est réalisé selon des scénarios définis. Une équipe de hackers éthiques le réalise.* » conclut Benjamin Aubertel.

MyTF1 : supporter 3 millions de connexions simultanées

François Tonic
Rédacteur en chef de Programmez!

MyTF1 est la plateforme de streaming (direct, replay) du groupe TF1. La coupe du monde 2018 a mis à l'épreuve le service avec des indisponibilités chroniques. En cause : trop de connexions simultanées saturant l'infrastructure. À partir de là, un vaste chantier est mené.

Comme le résume Thierry Bonhomme, CTO de eTF1, le chantier ouvert il y a 5 ans doit permettre de gérer et supporter les pics de fréquentations, de supporter l'ensemble des terminaux accédant à MyTF1. L'objectif est « simple » : disposer d'une plateforme de streaming à fort volume.

La plateforme comprend le front, le back et l'infrastructure. Côté infrastructure, le cloud a été adapté apportant une plus grande souplesse. Cette 1ere étape réalisée, l'architecture back a été refaite. La couche front a peu bougé, car l'impact est limité. Une stratégie sécurité globale a été entamée dans une seconde phase.

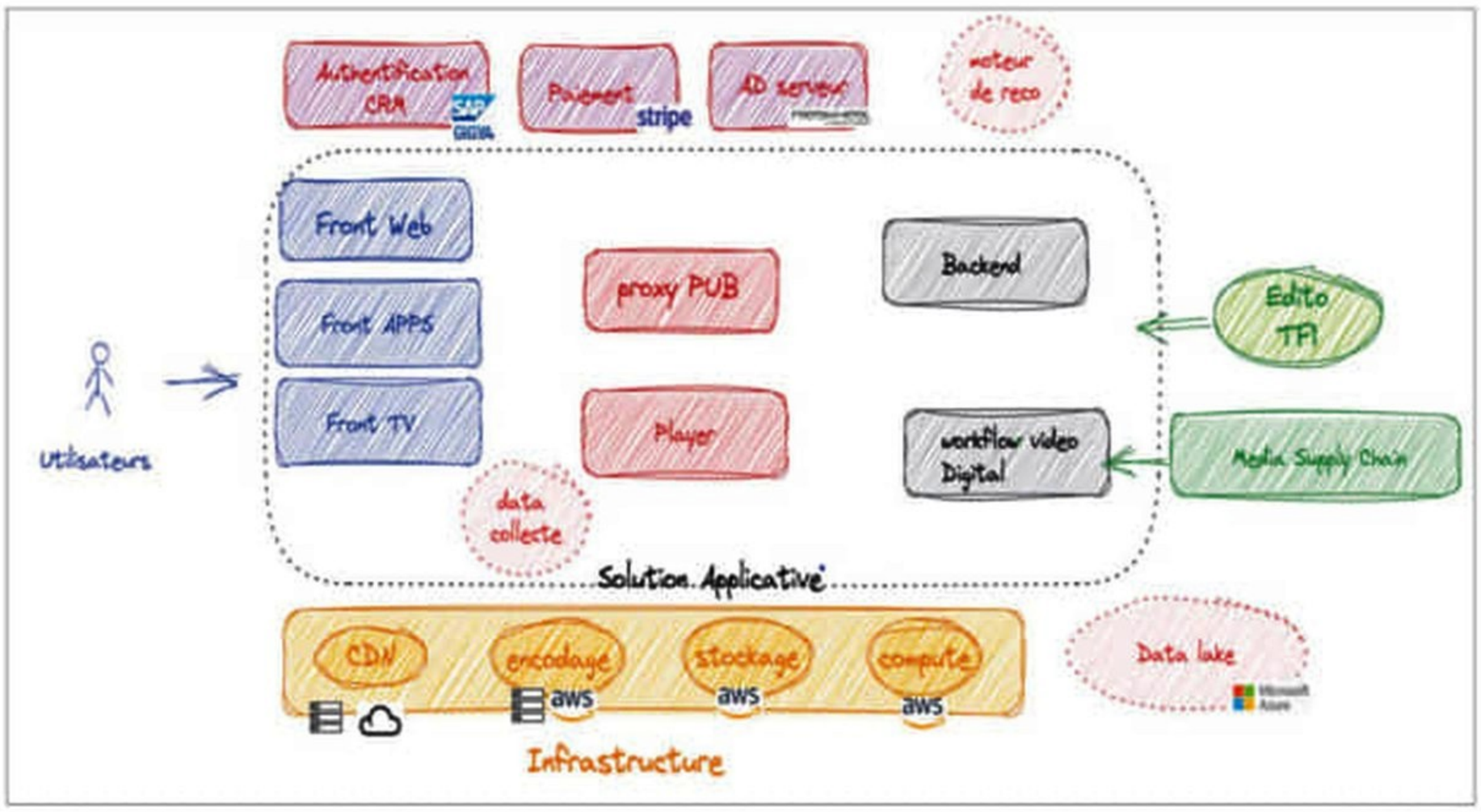
« Nos pics de charges représentent jusqu'à 3,5 millions de personnes en même temps durant plus de 5 minutes. Pour supporter une telle volumétrie, nous avons opté pour le langage Go, sur le côté back. Kafka complète le back. La plateforme supporte le direct, le streaming et le replay » précise Thierry Bonhomme. L'autre enjeu fut le lancement du service payant MyTF1 Max. Cela rajoute une nouvelle couche fonctionnelle : gérer les abonnements, qui a accès à quoi, etc.

L'infrastructure AWS a été choisie pour déployer les couches fonctionnelles et applicatives. Nous y trouvons notamment Amazon EKS (partie Kubernetes), AWS CloudFront. Le back a été écrit en Go. *« Go fut un choix purement technique et structurant. Il est assez facile à coder et consomme relativement peu de ressources. Il y a aussi un côté humain pour attirer de nouveaux profils dans nos équipes et Go est une technologie récente, mais mature »* poursuit Thierry Bonhomme. Et Rust ? Quelques PoC ont été réalisés, mais le déploiement de Rust nécessiterait une refonde du back et pour le moment Go répond parfaitement aux besoins.

Sur la partie API, elles sont programmées en GraphQL et un système API Gateway est déployé pour filtrer et sécuriser les requêtes entrantes et sortantes.

Secure by design : une sécurité par le risque

Depuis 2 ans, la sécurité est au cœur de la transformation de la plateforme. Elle est initiée par la gestion du risque. Une échelle des risques est définie de 5 à 1. Cette approche permet de prioritiser les risques par rapport à la criticité. *« En plus de cette approche, un hacker éprouve la plateforme une fois par an »*, explique Thierry Bonhomme. Ce pentest permet de voir les failles potentielles ou les points faibles à améliorer.



Aujourd'hui, la sécurité est gérée par un chef de projet dédié, avec son équipe.
Les codes sont aussi testés et stressés pour voir si un login n'est pas oublié en dur dans le code. L'autre pan de la sécurité concerne la CNIL et le RGPD. « *Cette partie est structurante dans le sens où nous avons une plateforme gratuite avec un service payant. Il faut gérer les données personnelles et les protéger.* » explique Thierry Bonhomme.

AWS propose une sécurité par défaut liée aux services cloud ce qui a aidé les équipes. Cependant, la partie run doit toujours être surveillée. « *Il faut vérifier, observer. Par exemple, détecter une IP anormale. Il faut analyser les tableaux de bord. Nous avons mis en place des outils pour ces aspects. Notre approche cloud permet aussi de mieux gérer une attaque massive de type déni de service (DoS & DDOS).* » poursuit Thierry Bonhomme. Sur la partie métrique et tableau de bord, le groupe TF1 s'appuie sur Datadog et Grafana. Cela permet de tracer les requêtes et comprendre ce qu'il se passe en cas de problème et de faire le lien entre un élément infra et un élément back.

La sécurité passe aussi par la disponibilité de la plateforme et sa capacité à monter en charge, et à descendre, selon les pics. Le choix de Kubernetes s'est naturellement imposé, avec le service EKS. « *Nous avons fait un PoC et celui-ci est fonctionné. Cette approche permet de monter en charge assez rapidement. Nous ne pouvons pas nous rater sur la disponibilité. Nous préparons en amont l'infrastructure pour prévoir un pic de charge sur un événement. L'autoscaling sera progressivement déployée pour automatiser le provisionnement de l'infrastructure* » explique Thierry Bonhomme.

Basiquement, il y a 3 types d'attaquants, d'attaquants :

- Le hacker qui veut trouver une vidéo non visible sur la plateforme pour voir à l'avance un épisode ou le résultat d'une émission telle que The Voice
- Les fausses plateformes de streaming qui poussent des flux de MyTF1 / MyTF1 Max. Pour protéger les flux et les vidéos, la plateforme utilise des DRM et des clés de chiffrement liés aux flux et aux utilisateurs.
- Le hacker qui veut saturer la plateforme avec des attaques de type DoS en passant sous le radar de surveillance, en sniffant des IP. Le hacking du site France24 il y a une dizaine d'années reste « la » référence.

Si les aspects sécurités sont bien intégrés par les équipes infrastructure, comme le dit Thierry Bonhomme, il faut remonter jusqu'aux codes, donc aux développeurs. Il faut sensibiliser, les former. Par exemple : trouver des secrets qui ne devraient pas être dans le code, vérifier les développeurs ont les bons droits (ni plus, ni moins) pour accéder aux différentes couches. Cette approche globale permet de mettre en place du DevOps, en attendant de passer au DevSecOps.

En plus de cela, le groupe TF1 utilise aussi Terraform. Cela permet de structurer les provisionnements des ressources de l'infrastructure. Objectif : industrialiser et automatiser. Actuellement, seule l'équipe infra y a accès.

PROGRAMMEZ!

Directives de compilation

Programmez! hors-série n°12
AUTOMNE 2023

Directeur de la publication & rédacteur en chef
François Tonic
ftonic@programmez.com

Contacter la rédaction
redaction@programmez.com

Les contributeurs techniques
Stéphane Dubreuil
Sabrine Guihéneuf
Andrew Vennekotter
Mikaël Barbera
Bertrand Méens
Sylvain Cailliau
Romy Alula
Guillaume Mathieu
Christophe Nasarre
Riadh Mnasri
Grégory Damagala
Adrien Pessu
Franck Ebel
Equipe Checkmarx
Arnaud Pilon,
Théo Louis-Tisserand
Aymeric Palhiere
Christophe Villeneuve
Nicolas Vuillamy
Natacha Bakir
Max Thomas

Maquette
Pierre Sandré

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

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

Impression
SIB Imprimerie, France

Dépôt légal
A parution

Commission paritaire
1225K78366

ISSN
2279-5001

Abonnement
Abonnement (tarifs France) : 49 € pour 1 an,
79 € pour 2 ans. Etudiants : 39 €. Europe et Suisse : 55,82 € - Algérie, Maroc, Tunisie : 59,89 € - Canada : 68,36 € - Tom : 83,65 € - Dom : 66,82 €.
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 €.

Nefer-IT
57 rue de Gisors, 95300 Pontoise France
redaction@programmez.com
Tél. : 09 86 73 61 08

Toute reproduction intégrale ou partielle est interdite sans accord des auteurs et du directeur de la publication. © Nefer-IT / Programmez!, septembre 2023.

Abonnez-vous à Programmez! Abonnez-vous à Programmez! Abonne

PROGRAMMEZ!
Le magazine des développeurs

nos classiques

1 an → 10 numéros
(6 numéros + 4 hors séries) — 55€ *(1)

2 ans → 20 numéros
(12 numéros + 8 hors séries) — 90€ *(1)

Etudiant
1 an → 10 numéros
(6 numéros + 4 hors séries) — 45€ *

Option : accès aux archives
Pour abonnement 1 an — 25€
Pour abonnement 2 ans — 30€

* Tarifs France métropolitaine

(1) Au lieu de 69,90 € ou 139,80 € selon l'abonnement, par rapport au prix facial.

abonnement numérique

PDF ..... 45€
1 an

Souscription directement sur
www.programmez.com

Toutes nos offres sur 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 : .....45 €
Photocopie de la carte d'étudiant à joindre

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 spécial 12

Nouvelles formations à l'IA pour les devs

avec ENI Service

Chatbot
Créer un agent conversationnel avancé avec ChatGPT

ChatGPT API
Ajouter des fonctionnalités d'IA générative dans les applications web et mobiles

ChatGPT
Augmenter la productivité, la qualité et la sécurité des développements informatiques

ChatGPT, CodeGPT et GitHub Copilot
Devenez un développeur augmenté par l'IA

ENI Service - Centre de Formation Informatique
Formations professionnalisantes en présentiel ou à distance
formation@eni.fr - 02 40 92 45 50

www.eni-service.fr