MISC
LE MAGAZINE DE LA CYBERSÉCURITÉ OFFENSIVE ET DÉFENSIVE

FÉVRIER MARS 2019

FRANCE
MÉTRO. : 12,90 €
DOM/TOM : 13,90 €
BEL/LUX/PORT.
CONT. : 13,90 €
CH : 18 CHF
CAN : 18 $CAD

HORS-SÉRIE N°19

NOUVELLE FORMULE !

SCIENCES
Détection d'intrusion grâce au Machine Learning : introduction à SecuML

COMPRENDRE LES VULNÉRABILITÉS DE L'IOT

▸AUTOMOBILE
Bus CAN en pratique : débuter dans l'analyse des communications de votre véhicule

▸FLASH
Utilisez le dump de mémoire flash pour la recherche de vulnérabilités dans les firmwares

▸TÉLÉPHONIE MOBILE
De GPRS à LTE : tour d'horizon des surfaces d'attaques contre les protocoles de téléphonie mobile

▸GPS
Du leurrage à l'analyse d'un tracker, analysez la sécurité qui entoure le GPS

L 16844 - 19 H - F: 12,90 € - RD

PROCESSEURS
Analyse de la vulnérabilité Foreshadow sur les puces Intel

THREAT INTELLIGENCE
Présentation du framework ATT&CK du MITRE pour l'analyse et la détection post-compromission

Quarkslab

SECURING EVERY BIT OF YOUR DATA

CHAQUE ENJEU DE SÉCURITÉ EST DIFFÉRENT, CHAQUE SOLUTION DOIT L'ÊTRE.

PRODUITS

IRMA Enterprise

Une plateforme flexible d'analyse de fichiers, utilisant plusieurs moteurs d'analyse pour améliorer la détection des menaces.

Epona

Une protection logicielle pour vos applications prévenant les attaquants de voler vos actifs et menacer vos utilisateurs.

SERVICES

Nos recherches de pointe en sécurité conduisent les organisations à une nouvelle posture de sécurité : obliger l'attaquant, et non le défenseur, à s'adapter.

- **Recherche de vulnérabilités : évaluer la sécurité CSPN**
  Évaluation sécuritaire des produits, attaques logicielles et matérielles.
- **Reverse engineering : comprendre la sécurité**
  Tests de protections (jeux, paiement, DRM,...), développement de patches, attaques hardware.
- **Vulnerability intelligence : trier les menaces**
  Développement d'exploits pour tester des équipements (ex.: blueborne), analyse de patches, attaques par canaux auxiliaires.
- **Sécurité logicielle : construire la sécurité.**
  Cryptographie (conception et optimisation), design sécurisé, développement de composants de sécurité.

FORMATIONS

Apprenez la sécurité avec ceux qui la pratiquent quotidiennement

- Reverse engineering comme un pro
- Applications Android du point de vue d'un reverse engineer
- iOS : sécurité des applications et de l'OS

Plus d'informations sur **www.quarkslab.com**

Quarkslab
SECURING EVERY BIT OF YOUR DATA
13 rue St.-Ambroise - 75011 Paris - FRANCE
Tel.: +33 (0)1 58 30 81 51 - Email: contact@quarkslab.com
@quarkslab - **www.quarkslab.com**

MISC Hors-Série est édité par **Les Éditions Diamond**
10, Place de la Cathédrale – 68000 Colmar – France
Tél. : 03 67 10 00 20 / **Fax :** 03 67 10 00 21
E-mail : cial@ed-diamond.com
Service commercial : abo@ed-diamond.com
Sites : https://www.miscmag.com – https://www.ed-diamond.com
Directeur de publication : Arnaud Metzler
Chef des rédactions : Denis Bodor
Rédacteur en chef : Cédric Foll
Rédacteur en chef adjoint : Émilien Gaspar (gapz)
Secrétaire de rédaction : Aline Hof
Responsable service PAO : Kathrin Scali
Responsable publicité : Tél. : 03 67 10 00 27
Service abonnement : Tél. : 03 67 10 00 20
Impression : pva, Druck und Medien-Dienstleistungen GmbH,
Distribution France : (uniquement pour les dépositaires de presse)
MLP Réassort :
Plate-forme de Saint-Barthélemy-d'Anjou. Tél. : 02 41 27 53 12
Plate-forme de Saint-Quentin-Fallavier. Tél. : 04 74 82 63 04
Service des ventes :
Abomarque : 09 53 15 21 77
IMPRIMÉ en France - PRINTED in France
Dépôt légal : A parution
N° ISSN : 1631-9036
Commission Paritaire : K 81190
Périodicité : Bimestrielle
Prix de vente : 12,90 Euros
Première parution : Février 2019

Les articles non signés contenus dans ce numéro ont été rédigés par les membres de l'équipe rédactionnelle des Éditions Diamond.

CHARTE DE MISC :

MISC est un magazine consacré à la sécurité informatique sous tous ses aspects (comme le système, le réseau ou encore la programmation) et où les perspectives techniques et scientifiques occupent une place prépondérante. Toutefois, les questions connexes (modalités juridiques, menaces informationnelles) sont également considérées, ce qui fait de MISC une revue capable d'appréhender la complexité croissante des systèmes d'information, et les problèmes de sécurité qui l'accompagnent. MISC vise un large public de personnes souhaitant élargir ses connaissances en se tenant informées des dernières techniques et des outils utilisés afin de mettre en place une défense adéquate. MISC propose des articles complets et pédagogiques afin d'anticiper au mieux les risques liés au piratage et les solutions pour y remédier, présentant pour cela des techniques offensives autant que défensives, leurs avantages et leurs limites, des facettes indissociables pour considérer tous les enjeux de la sécurité informatique.

ÉDITO

ÉTHIQUE ET SÉCURITÉ

En ce début d'année 2019, le prix d'un exploit pour un remote jailbreak d'iOS est désormais passé à la modique somme de 2 millions de dollars sur la plateforme Zerodium. Pour rappel et pour ceux qui ne suivent pas cette actualité, l'ancien prix était d'un 1.5 millions de dollars et l'on parle bien de vendre de manière exclusive un exploit 0day fonctionnel (non d'une vulnérabilité comme on peut l'entendre ici et là).

Cette récente actualité a ravivé quelques intenses discussions sur une question importante, mais souvent peu traitée par la communauté de la sécurité informatique : l'éthique. Derrière les discours moralisateurs et les réactions caricaturales existe une réalité simple : un cadre légal le plus souvent dépassé ou qui n'est pas respecté, et des usages de ces exploits sans aucun contrôle possible. Car oui, on peut déployer très discrètement ce genre d'outil, contrairement à bien d'autres utilisés lors de conflits plus conventionnels (sabotage, guerre classique). L'usage qui est d'ailleurs fait de ces exploits n'est pas connu, même si l'on peut facilement imaginer qui se les paye et les utilise : forces de l'ordre, grandes entreprises, États, etc. Et comme toujours, on peut aussi facilement s'imaginer que cela peut servir d'un côté à la lutte contre les pédo-nazi-criminels-etc, je caricature volontairement, et d'un autre côté, à surveiller et réprimer des opposants politiques. Je vous épargnerai l'analogie entre les exploits 0days et les armes, la sécurité informatique est un pan à part entière des conflits modernes dont les bords se précisent peu à peu, comme le montre la récente possibilité pour les forces armées françaises de pouvoir effectuer une riposte.

La question éthique n'est bien sûr pas nouvelle. Le piratage de l'entreprise Hacking Team, qui vendait et vend toujours des logiciels de surveillance, a permis de révéler sa liste de clients qui était composée notamment de forces de l'ordre de pays extrêmement répressifs vis-à-vis de leurs opposants politiques. On a également trouvé dans toutes ces données, des exploits provenant du marché des 0days, tout naturellement.

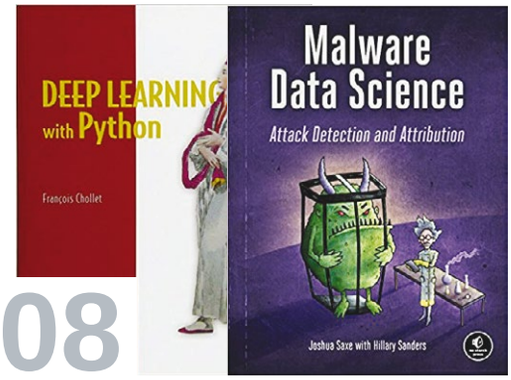
En ces temps de débats (ou pas), ne devons-nous pas nous ressaisir de la question éthique en profondeur et ne plus ignorer les conséquences directes des travaux en sécurité informatique ? De tout temps, l'excellence technique a été et est encore le marqueur principal pour la communauté, ne manque qu'une éthique à la hauteur de cette dernière. On notera par ailleurs la tribune du directeur général de l'ANSSI dans *Libération* **[0]**, qui s'oppose à l'introduction de porte dérobée pour des besoins de légaux (enquête et renseignement), surtout d'un point de vue technique (affaiblissement globale de la sécurité à long terme si l'on introduit une porte dérobée, donc une vulnérabilité supplémentaire), et laisse l'épineuse question éthique en suspens. Est-ce si fou que cela de demander à nos outils numériques du quotidien de ne pas nous surveiller ?

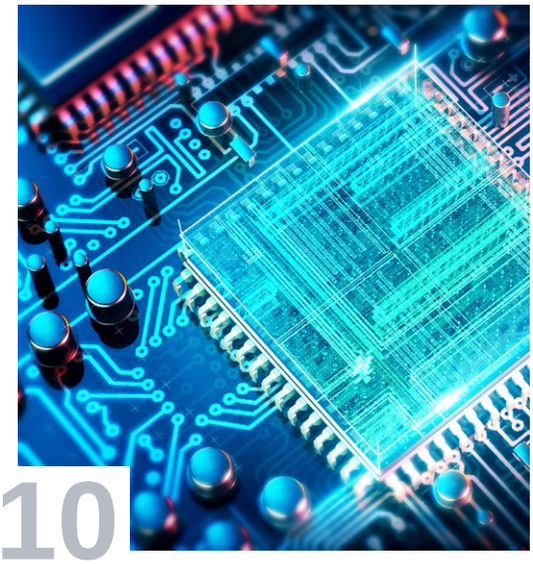
Émilien GASPAR / gapz / eg@miscmag.com

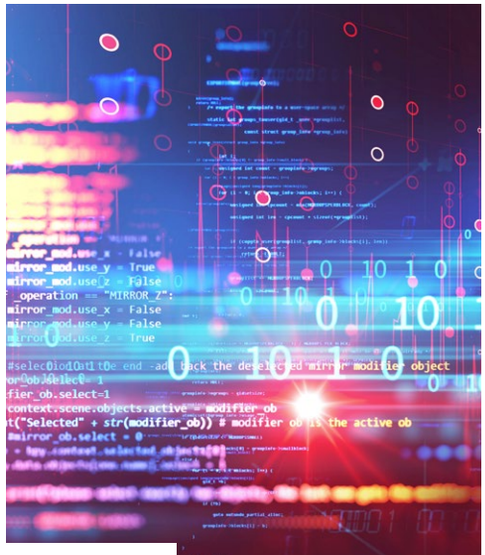
[0] https://www.liberation.fr/debats/2019/01/21/tous-connectes-tous-responsables_1704228

SOMMAIRE

MISC HORS-SÉRIE N° 19







FACILITEZ-VOUS LA VEILLE TECHNO

Sécurité informatique, Open Source, Linux, Électronique, Embarqué...

CONNECT

c'est la solution pour vous et votre équipe !

À découvrir sur :
connect.ed-diamond.com

Pour vous abonner :
www.ed-diamond.com

Accès par connexions simultanées pour :
Pro, R&D, Enseignement...

En savoir plus :
Tél. : +33 (0)3 67 10 00 28
E-mail : connect@ed-diamond.com

ET L'ACCÈS À LA DOCUMENTATION !

Un accès à
+ de 7000 articles

Disponibles pour :

MISC · GNU LINUX MAGAZINE / FRANCE · GNU LINUX PRATIQUE · HACKABLE MAGAZINE

Consultez les numéros d'hier et ceux d'aujourd'hui

Un outil de recherche

Un affichage par numéro paru

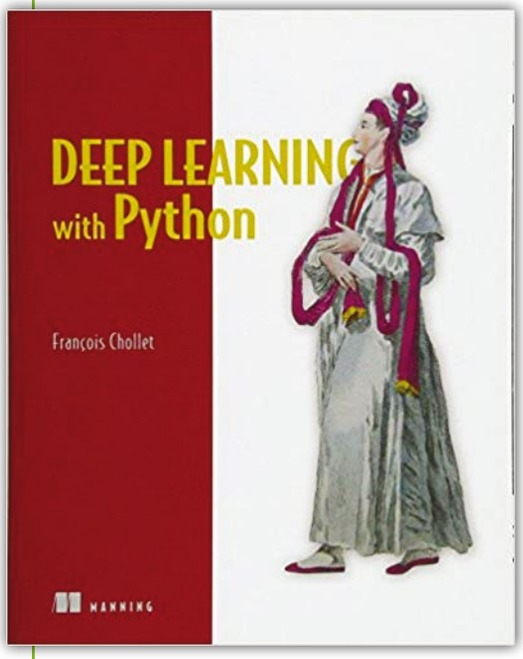
Les magazines standards et leurs hors-séries

Des articles triés par domaines

connect.ed-diamond.com

REVUE DE LIVRES...

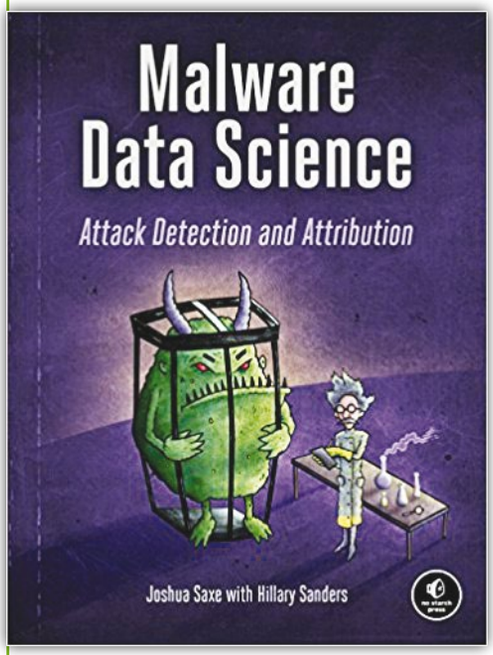
Robert ERRA



DEEP LEARNING WITH PYTHON

Pour celles et ceux qui pensent que le Deep Learning demande un doctorat en mathématiques, ce livre est fait pour vous. Attention au titre il est trompeur : c'est un livre sur Keras. Clair. Précis. Les codes (encore du Python) sont incroyablement faciles à lire et à utiliser, et pour cause, il est écrit par le créateur de Keras, un des frameworks de Deep Learning les plus remarquables. Les exemples sont remarquables et montrent qu'il est relativement facile de démarrer un projet de Machine Learning, dans ce cas précis de Deep Learning. C'est un complément idéal à l'autre ouvrage présenté ici, si vous voulez aller plus loin. On peut citer deux chapitres, le premier (chapitre 6) montre comment on peut aujourd'hui analyser des textes avec le Deep Learning. Un autre chapitre me semble tout aussi remarquable : le chapitre 9 traitant des réseaux adverses génératifs (en anglais *Generative Adversarial Networks* ou GANs) qui datent de quelques années et qui permettent de générer assez facilement des images extrêmement réalistes à partir d'un jeu de données. Les lecteurs sont vivement encouragés à tester les codes. Cet ouvrage (tout autant que l'autre) est une preuve que le Machine Learning se démocratise.

- **Auteur :** François Chollet
- **Éditeur :** Manning Publications
- **Parution :** janvier 2018
- **Nombre de pages :** 384
- **Prix :** 31,13 €



MALWARE DATA SCIENCE: ATTACK DETECTION AND ATTRIBUTION

Ce livre contient une courte introduction au *reverse engineering* et au désassemblage de code, mais ce n'est pas le thème du livre. De même, il y a un chapitre sur le Deep Learning, mais ce n'est pas non plus un livre sur le Deep Learning. En revanche, si vous aimez ce qu'on peut maintenant appeler la Malwarologie, soit la (data)science des malware, ce livre est fait pour vous ! Qu'est-ce donc que la datascience ? Les auteurs en donnent une bonne définition : « *A growing set of algorithmic tools that allow users to understand and make predictions about data using statistics, mathematics, and artful statistical data visualizations* ».

J. Saxe travaille chez Sophos, mais auparavant il a travaillé plusieurs années chez Invincea Labs (rachetée par Sophos), années pendant lesquelles il a développé de nombreux algorithmes et outils d'analyse et de comparaison de malware, et on sent dans cet ouvrage qu'il maîtrise ce sujet. H. Sanders travaille aussi chez Sophos et tous deux sont des habitués des conférences comme Black Hat.

Livre très facile à lire, il contient du code (Python évidemment) qu'on peut aisément s'approprier. Donc, si vous aimez, ou si vous avez besoin de comparer, classifier et clustériser des malware, lisez ce livre. Les exemples sont faciles à reproduire : si vous n'avez pas de malware sous la main, les jeux de données sont téléchargeables (voir les appendices) et il y a même un site qui accompagne le livre sur lequel vous trouverez une machine virtuelle prête à l'emploi.

Si vous voulez utiliser des graphes, ce qui est presque toujours une bonne idée, lisez le chapitre 4. Le chapitre 5 quant à lui vous aidera à comprendre comment trouver le code commun à des malware tandis que vous trouverez dans le chapitre 11 comment construire (rapidement) de A à Z un détecteur de malware via Keras (sujet du second livre présenté ici).

Un regret, les problèmes de passage à l'échelle, c'est-à-dire de la maîtrise de la complexité des algorithmes, sont peu traités dans le livre. Passer de quelques centaines ou quelques milliers de malware à plusieurs millions demande disons un « peu de doigté ». ▌

- **Auteurs :** Joshua Saxe, Hillary Sanders
- **Éditeur :** No Starch Press
- **Parution :** septembre 2018
- **Nombre de pages :** 272
- **Prix :** 34,69 €

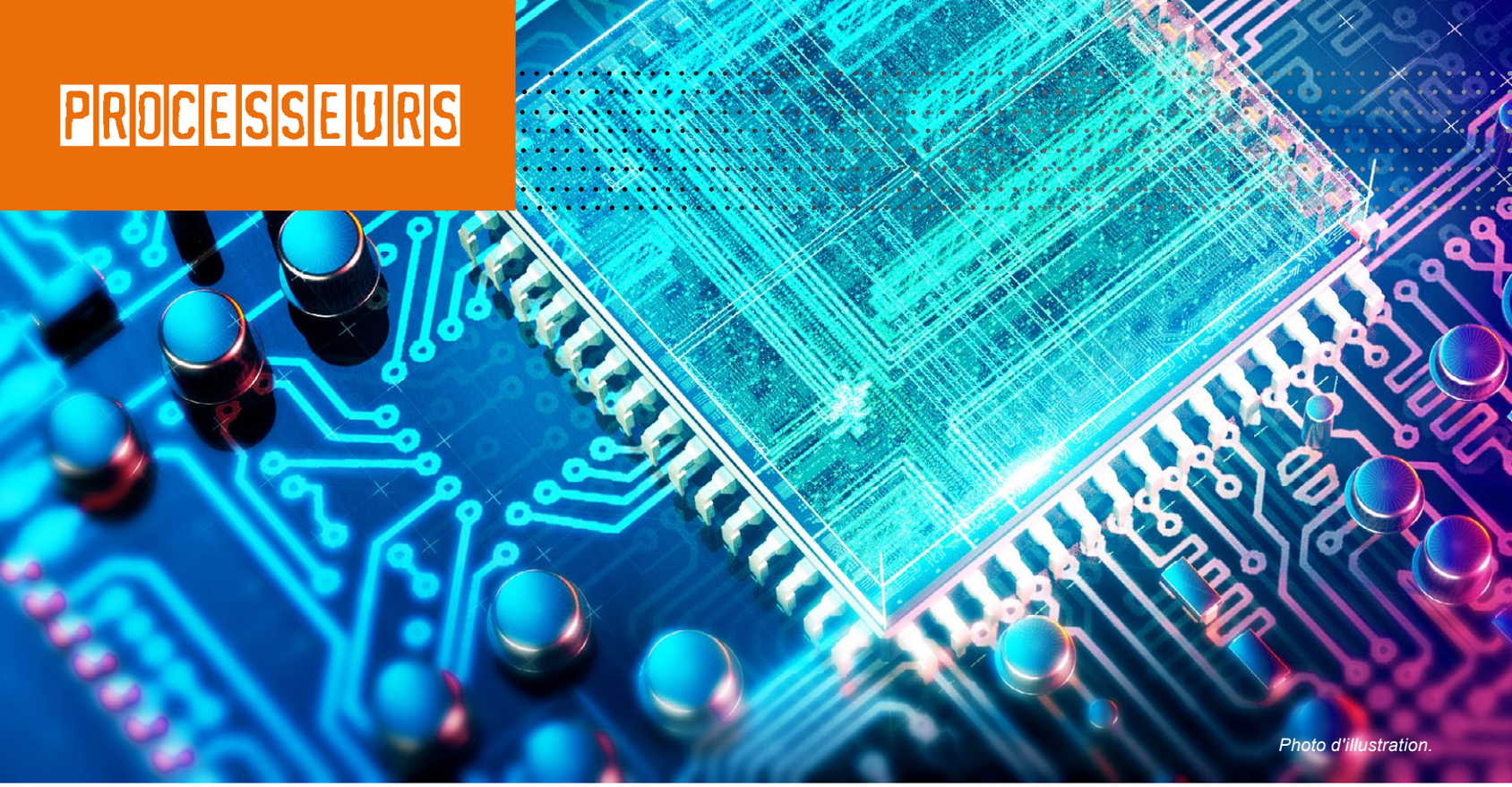


Photo d'illustration.

FORESHADOW-SGX, NOUVELLE VULNÉRABILITÉ SUR LES PROCESSEURS INTEL

Salma EL MOHIB

Experte sécurité @ digital.security – @lychnis42

En début d'année 2018, les attaques Meltdown et Spectre ont remis en question la conception de la plupart des processeurs du marché en exploitant des optimisations CPU. Cependant si certaines technologies comme les SoftGuard Extensions d'Intel ont pu leur résister, la publication de la vulnérabilité Foreshadow en août 2018, encore une fois liée aux optimisations du fabricant, laisse entrevoir de nouvelles menaces pour le modèle de confiance conféré par Intel SGX. Cet article présente l'attaque Foreshadow sur les enclaves de la technologie SGX.

Durant les 50 dernières années, l'industrie des processeurs a connu une évolution vertigineuse. La miniaturisation de la taille des transistors et la finesse de leur gravure ont permis aux fréquences auxquelles tournent les processeurs actuels de dépasser le GHz, pouvant ainsi exécuter plus d'un milliard de cycles d'instructions par seconde.

Toutefois, au vu des limites physiques imposées au rétrécisse-

ment des transistors, un mur technologique se profile. Des techniques d'optimisation comme l'exécution spéculative ont alors été introduites pour booster la puissance de calculs.

Si les fabricants se sont rués sur la performance, la découverte des vulnérabilités matérielles Spectre et Meltdown a néanmoins démontré que la sécurité a été laissée pour compte. En effet, l'exploitation de ces vulnérabilités tire parti essentiellement de ces mêmes optimisations qui ont permis à la puissance de calcul de franchir de nouvelles étapes. Par ailleurs, si les enclaves d'Intel SGX ont été épargnées jusqu'alors, la publication de la vulnérabilité matérielle Foreshadow **[1]**, en août 2018, a permis de révéler l'existence d'une nouvelle menace pour le modèle de sécurité conféré par les processeurs Intel SGX. De fait, ces processeurs ont pour but de protéger des régions sélectionnées d'un programme (code et données) en renforçant leur sécurité au niveau matériel par la mise en place d'enclaves. Elles protègent alors ces parties du programme de tout accès extérieur, même dans le cas d'une compromission de la plateforme sur laquelle elles s'exécutent.

L'exploitation de Foreshadow, présentée dans la suite de cet article, a permis néanmoins d'anéantir ce modèle de confiance en biaisant le caractère privé des enclaves et en mettant à mal leur intégrité.

PROCESSEURS

Hiérarchie du cache et TLB

Le processeur, ou unité centrale de traitement, est le composant matériel en charge d'exécuter les intructions et traiter les données d'un programme. Il inclut une unité de contrôle dont la fonction est de décoder les instructions, gérer les registres ainsi que les interruptions. Cette unité de contrôle communique avec une unité d'entrée/sortie qui s'interface avec les autres périphériques comme la mémoire. Chaque processeur intègre une unité de gestion mémoire, la MMU (*Memory Management Unit*). Les adresses virtuelles référencées par l'instruction en cours doivent être traduites en adresses physiques. Ce procédé, nommé translation d'adresse, est réalisé par la MMU et nécessite le parcours de la table des pages qui est localisée en mémoire vive RAM (*Random Access Memory*).

Si la RAM permet de stocker un nombre important de données, elle nécessite néanmoins un temps d'accès conséquent. Afin de corriger cette latence, le CPU utilise alors un tampon, nommé *Translation LookAside Buffer* (TLB) qui garde une copie de toutes les translations d'adresses récemment effectuées.

Si le processeur tente de réduire les accès en mémoire vive pour les translations d'adresses, il en va de même pour la récupération de leurs contenus. C'est là qu'intervient le concept de cache **[2]**.

Le cache est un type de mémoire rapide et de taille réduite qui permet d'accélérer l'accès aux données. Il conserve une copie du contenu de certaines adresses physiques susceptibles d'être accédées par le processeur dans un futur proche.

Les caches sont divisés en niveaux, on parle alors de hiérarchie. Plus un cache est au plus près du microprocesseur, plus son accès est rapide, son coût cher, et sa capacité de stockage réduite. Intel intègre deux niveaux de cache dans la micro-architecture **[3]** de ses processeurs. Le cache L1 est le premier niveau de cache et le plus rapide. Souvent, il est divisé en deux types selon le contenu qu'il contient. Les données d'un côté sont stockées dans le cache L1 de données tandis que les instructions sont sauvegardées dans le cache d'instructions. Vient ensuite le cache de niveau 2 (L2). Ces deux caches sont situés sur le processeur et sont par conséquent partagés par les différents threads qui s'exécutent dessus.

On peut également retrouver un troisième niveau de cache appelé LLC (Last Level Cache)

de taille plus importante, partagé par tous les cœurs de processeurs physiques d'un microprocesseur multicœur.

De manière générale, lorsque le CPU doit traiter une nouvelle adresse, il commence par vérifier si son contenu est présent dans le cache. Si ce n'est pas le cas, il y a défaut de cache qui nécessite sa récupération depuis la mémoire centrale (cf. Fig 1).

Afin d'optimiser les accès en RAM, le chargement de cache se base sur les deux principes de localité : temporelle et spatiale.

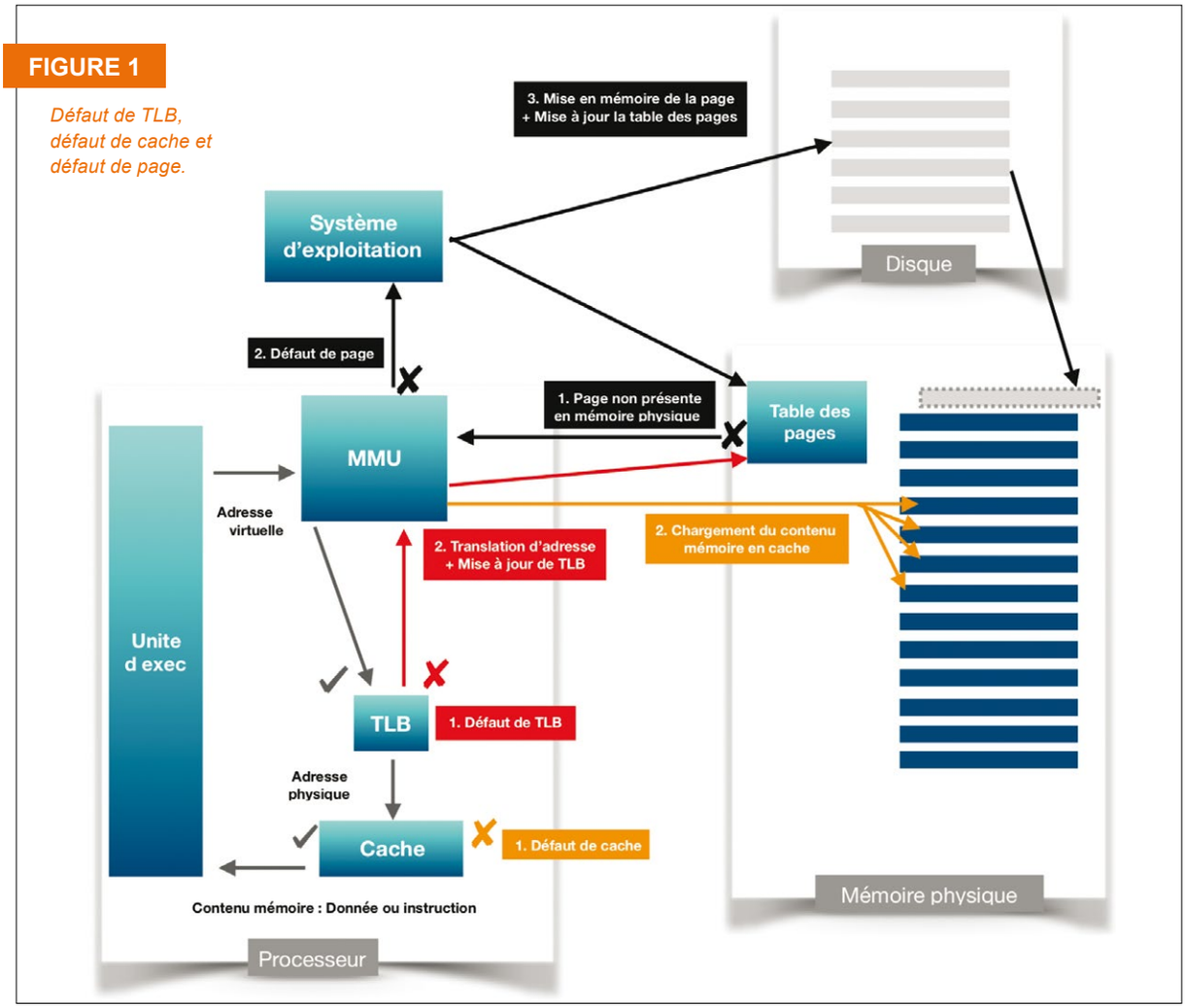
Le premier considère que si une adresse a été accédée par un programme, les chances sont accrues pour qu'elle soit de nouveau accédée dans un futur proche. Le contenu de l'adresse en question est donc conservé dans le cache. Le second principe, quant à lui, considère que des éléments proches en mémoire ont tendance à être accédés à des instants proches.

Concrètement, lorsqu'une adresse est déréférencée, son contenu ainsi que celui des adresses physiques qui lui sont contiguës en mémoire physique sont conservés dans le cache.

En outre, lorsque l'adresse physique recherchée n'est pas présente en mémoire physique, une faute de page est levée. Un défaut de page, ou

FIGURE 1

Défaut de TLB, défaut de cache et défaut de page.



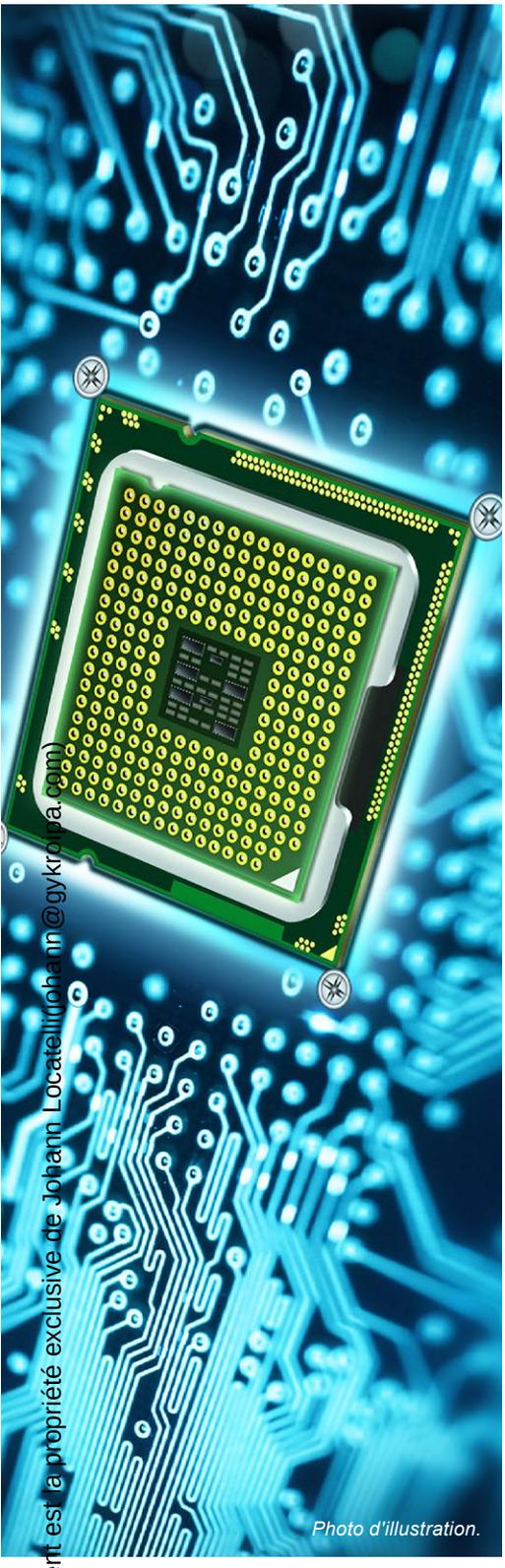


Photo d'illustration.

faute de page, est géré par le système d'exploitation qui alloue une nouvelle page mémoire et y charge le contenu de l'adresse adéquate depuis le disque. L'exécution peut alors reprendre (cf. Fig 1).

Optimisations : exécution dans le désordre et exécution spéculative

Pour profiter pleinement des ressources internes du processeur, d'importants travaux d'optimisation ont été menés. C'est dans ce sens que les exécutions spéculatives et dans le désordre (*out-of-order*) ont été intégrées. Leur implémentation permet d'exécuter de manière anticipée des instructions pour faire gagner du temps au processeur.

Dans ce modèle, les instructions sont différenciées selon deux types. D'un côté, nous retrouvons les instructions de branchement conditionnelles qui imposent au fil d'exécution, en fonction de l'évaluation d'une condition, de prendre différents branchements. D'un autre côté, nous retrouvons le reste des instructions, dites obligatoires.

Pour optimiser l'exécution du premier type d'instructions, l'exécution spéculative est utilisée. La réalisation d'une condition est alors spéculée, entraînant l'exécution en amont de la suite des instructions du branchement lui correspondant. Le résultat de cette exécution anticipée est alors stocké dans un tampon. Par la suite, lorsque le fil d'exécution réelle atteint cette condition, la condition spéculée est vérifiée. Si elle s'avère vraie, le processeur se servira directement des résultats de l'exécution spéculative, bénéficiant ainsi de l'économie de temps d'exécution. Dans le cas contraire, les résultats de l'exécution spéculative sont ignorés, et le processeur procède à l'exécution des instructions adéquates.

Pour les instructions non conditionnelles, le flot d'instructions est réarrangé en bouts de code indépendants, dont l'exécution en parallèle peut être réalisée dans le désordre.

Ce type d'exécution, dite dans le désordre, combiné avec l'exécution spéculative permet d'optimiser la disponibilité des données et d'anticiper les calculs du CPU pour lui faire économiser du temps.

INTEL SGX

Introduction

Les besoins de l'industrie informatique ont évolué avec l'externalisation des applications et leur exécution sur les serveurs de tiers, comme c'est le cas pour le Cloud Computing. La confiance habituellement conférée au système d'exploitation sur lequel tourne un programme n'est plus au goût du jour

dans ces cas de figure. C'est ainsi qu'est née l'exécution en environnement de confiance TEE (*Trusted Execution Environment*) **[4]**. Ce type d'exécution vise à protéger l'intégrité et la confidentialité d'un programme dans l'environnement dans lequel il s'exécute. Dans ce modèle, nous retrouvons les *SoftGuard Extensions* (SGX) d'Intel.

Intel SGX est un ensemble d'extensions et de composants ajoutés à la micro-architecture de certains processeurs Intel, pour permettre l'exécution de manière sécurisée d'un programme dans un environnement non fiable.

Ainsi, les extensions d'Intel SGX fournissent un ensemble d'instructions CPU permettant à une application hôte d'allouer des régions mémoires privées, appelées enclaves, où un code sensible est protégé contre tout accès depuis un code externe, peu importe son niveau de privilèges (noyau, hyperviseur, etc.).

Isolation de la mémoire

La conception d'Intel SGX considère le processeur comme étant le seul environnement où les données et le code d'une enclave peuvent résider en clair. Tout autre emplacement en dehors de la puce du processeur, est considéré non fiable et l'enclave doit y être chiffrée.

Ainsi, lorsqu'elle réside en RAM, la mémoire de l'enclave est inaccessible pour tout type d'utilisateurs. Cette protection par chiffrement permet de rendre inapplicables, entre autres, des attaques physiques par démarrage à froid (*Cold Boot Attack*) **[5]**.

Le chiffrement et le déchiffrement de la mémoire de l'enclave sont assurés au niveau matériel par un composant sur le processeur, appelé le *Memory Encryption Engine* (MEE) **[6]**.

Entrée et sortie de l'enclave

Pour effectuer une entrée/sortie à l'enclave, les instructions `EENTER` et `EEXIT` ont été ajoutées au jeu d'instructions. Elles permettent de transférer l'exécution entre un point précis de l'application hôte et un point précis de l'enclave. Toute entrée/sortie de l'enclave implique nécessairement de vider le cache de translations d'adresse TLB.

» Interruptions en mode enclave

Dans une architecture de processeur normale, lorsqu'une exception matérielle se produit, le contrôle est transféré au système d'exploitation au niveau du gestionnaire d'interruption, l'ISR (*Interrupt Service Routine*). Le contexte d'exécution de l'application, qui s'exécutait avant la levée de l'exception,

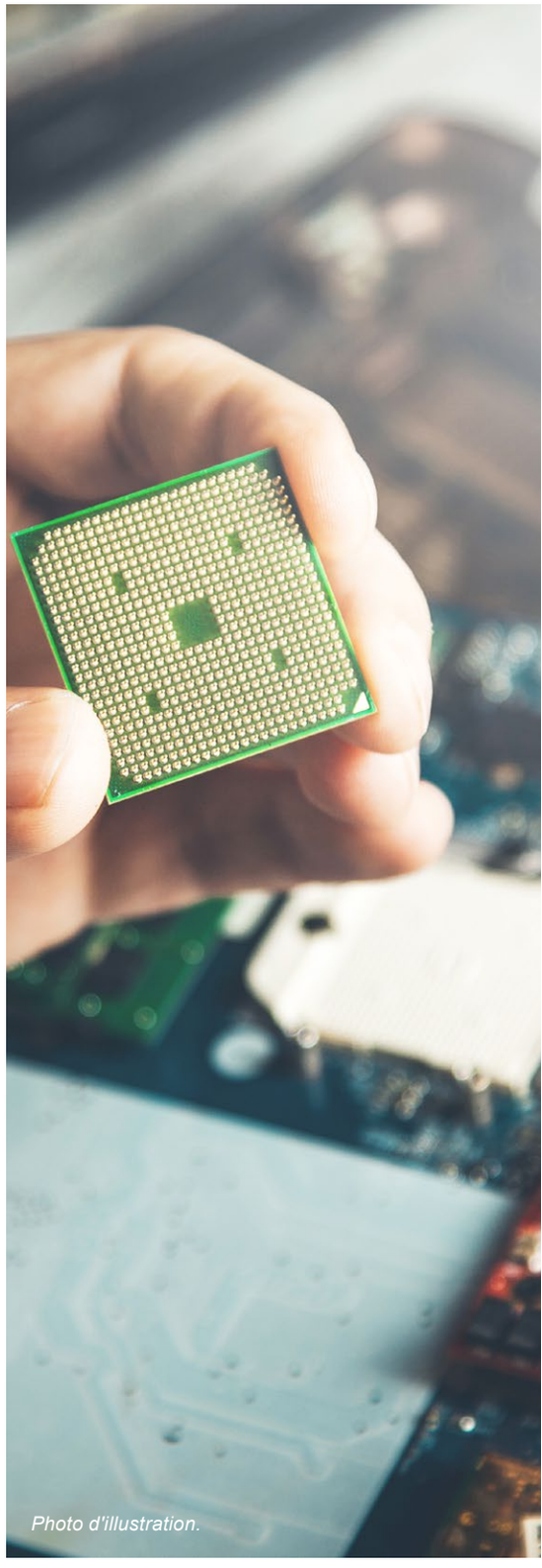


Photo d'illustration.

peut alors être lu ou modifié par le gestionnaire d'exception du système. Ce procédé est acceptable lorsque le contexte d'exécution de l'application est de confiance. En revanche, en prenant en compte le modèle d'attaque d'Intel SGX où le système d'exploitation n'est pas approuvé, lorsqu'une exception matérielle est levée le contexte d'exécution de l'enclave doit être nettoyé avant d'invoquer le gestionnaire d'exception.

La procédure matérielle de sortie asynchrone de l'enclave AEX (*Asynchronous Enclave Exit*) de SGX a pour tâche de préparer ce changement de contexte en préservant la sécurité de l'enclave. Elle commence par sauvegarder le contexte d'exécution dans une structure au sein de l'enclave, avant de nettoyer les registres et restaurer le contexte de l'application hôte.

Le gestionnaire d'exception du système prend alors la main le temps de gérer l'exception puis transfère à nouveau le contrôle au gestionnaire de sortie asynchrone AEP (*Asynchronous Exit Pointer*) de l'application hôte de l'enclave.

L'AEP décide par la suite si l'enclave reprend l'exécution ou pas. Dans le cas d'une reprise de l'exécution de l'enclave, le gestionnaire exécute l'instruction ERESUME pour remettre le processeur en mode enclave et restaurer le contexte et l'état des registres sauvegardés avant la propagation de faute. L'exécution de l'enclave peut alors reprendre.

MELTDOWN ET SGX

Meltdown

» Fonctionnement

La vulnérabilité matérielle Meltdown, rendue publique en janvier 2018, exploite une optimisation matérielle implémentée dans la plupart des processeurs Intel depuis 1995. L'optimisation en question est de type exécution dans le désordre.

Si cette optimisation augmente la puissance de calcul du CPU, elle introduit néanmoins une vulnérabilité permettant à un attaquant d'exécuter du code échappant momentanément à tout contrôle.

Les systèmes d'exploitation séparent l'espace mémoire utilisateur de l'espace mémoire noyau. Avec la virtualisation, ils confèrent à chaque processus une protection de leur espace d'adressage, empêchant tout autre processus d'y accéder et encore moins de le modifier.

Meltdown tire parti d'une situation de compétition (*race condition*) où une fenêtre de temps s'ouvre et offre à l'attaquant la possibilité d'exécuter des instructions échappant de manière transitoire à tout contrôle logiciel. Elles peuvent alors accéder pendant un laps de temps à tout l'espace d'adressage mémoire, y compris celui du noyau depuis un processus non privilégié.

Prenons l'exemple d'un code exécuté en mode utilisateur qui contient une instruction déréférençant une adresse mémoire du noyau (cf. Fig.2 (1)). L'exécution de cette instruction provoque une faute au niveau matériel, qui notifie le système d'exploitation d'une violation d'accès mémoire (cf. Fig. 2 (2)). Le système d'exploitation répondra alors par l'envoi d'un signal au processus illégitime, qui le gèrera par le biais du gestionnaire d'exception.

Néanmoins, le changement de contexte entre le processeur et le système d'exploitation crée une fenêtre de temps où il devient possible d'exécuter des instructions non maîtrisées.

En effet, cette latence dans l'intervention du noyau, combinée avec l'optimisation d'exécution dans le désordre, rend possible l'exécution d'instructions illégitimes dans le but d'accéder à du contenu mémoire protégé (cf. Fig.2 (2b)).

Une fois la faute gérée, le processeur ignore les résultats de l'exécution transitoire illégitime, et continue son exécution.

Cependant, à ce stade, au niveau matériel une brèche apparaît : le résultat de l'exécution transitoire réside désormais dans le cache. Par conséquent,

l'attaquant sait qu'il a pu exfiltrer le contenu d'une adresse protégée vers le cache, il ne lui reste plus qu'à trouver un moyen de le récupérer.

L'attaque nécessite donc d'établir un canal caché (*covert channel*) pour communiquer à l'attaquant les informations divulguées. Ce canal caché est réalisé en exploitant la micro-architecture du processeur par le biais du cache L1. En effet, les données seront récupérées au niveau du premier niveau de cache en utilisant une attaque par canal auxiliaire FLUSH+RELOAD **[7]** sur le cache L1.

Pour rappel, l'attaque FLUSH+RELOAD est une attaque par canal auxiliaire (*side channel*) qui exploite le fait que les accès en cache sont nettement plus rapides qu'en RAM. En calculant le temps d'accès à des données, un attaquant peut conclure si la donnée accédée était présente en cache et donc vient d'être traitée par le processeur ou si elle provient de la RAM.

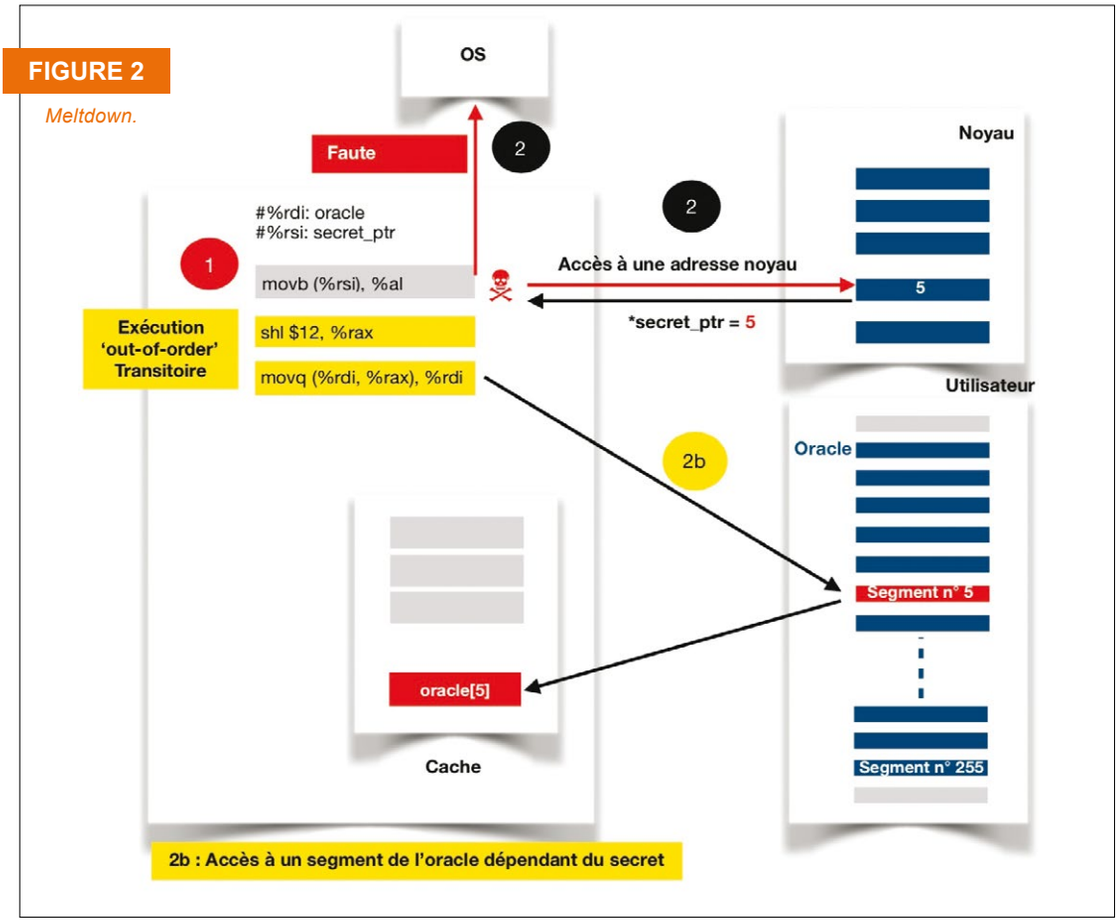
Comme cité précédemment, après le déréférencement de l'adresse protégée lors de l'exécution transitoire, l'attaquant doit pouvoir retrouver et récupérer le contenu secret dans le cache.

Lors de l'attaque, un seul octet de mémoire sera récupéré, nous parlerons par la suite alors d'un cycle. Un attaquant souhaitant exfiltrer la totalité d'une mémoire protégée va procéder en y accédant octet par octet, donc en réitérant ce cycle sur la totalité des adresses.

L'accès au contenu d'une adresse nous révèlera donc une donnée d'une taille d'un octet, donc 256 valeurs sont possibles.

FIGURE 2

Meltdown.



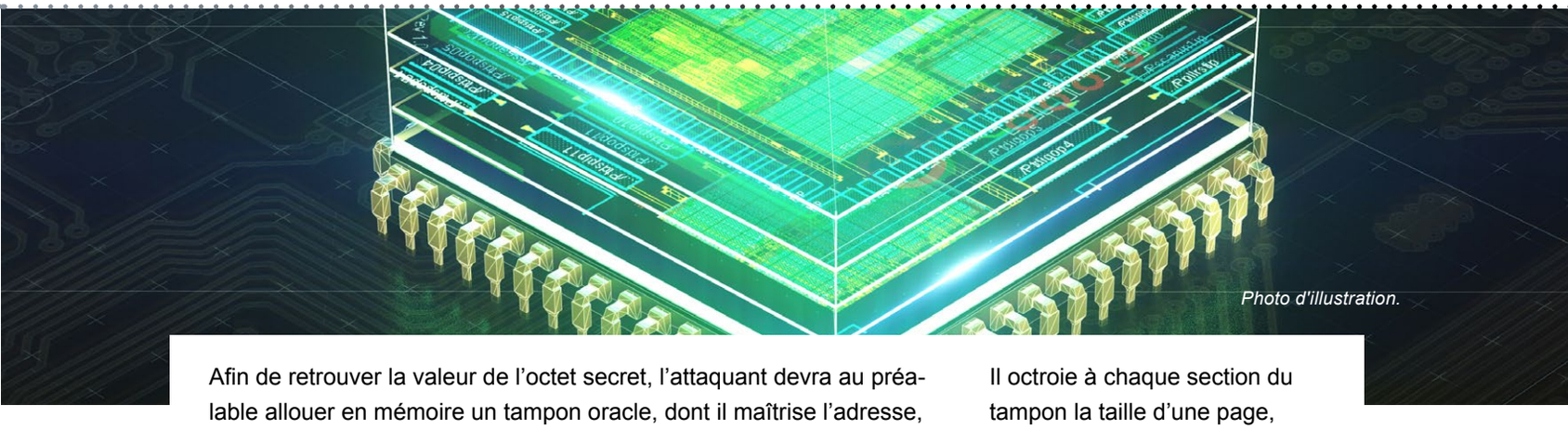


Photo d'illustration.

Afin de retrouver la valeur de l'octet secret, l'attaquant devra au préalable allouer en mémoire un tampon oracle, dont il maîtrise l'adresse, pouvant contenir les 256 valeurs possibles.

Après l'exécution de l'instruction accédant au secret lors de l'exécution transitoire, une seconde instruction sera exécutée pour accéder à une adresse qui dépend du secret, afin de la mettre dans le cache : `oracle[secret] = @oracle + secret` (cf. Fig.2 (2b)).

Par la suite, afin de récupérer le secret, l'attaquant va parcourir le tampon oracle dont il connaît l'adresse en mémoire, tout en calculant le temps d'accès à chacun de ses index. Pour un délai long, il pourra supposer que l'index était présent en mémoire physique, tandis que pour un temps d'accès rapide, il saura que l'index se trouvait en cache et donc que l'adresse accédée vient d'être traitée par le processeur et correspond à l'adresse dépendant du secret exfiltré.

```
temps_d'accès(oracle[j]) =
min(temps_d'accès(oracle[i])),
pour i de 0 à 0xff.
=> oracle[j] est dans le cache
=> (@oracle + j) vient d'être accédée, avec
@oracle connue par l'attaquant
=> secret = j
```

L'attaque Meltdown est donc réalisée en deux étapes. La première consiste, grâce à la situation de compétition, à exfiltrer la donnée secrète par un accès illégitime à une adresse protégée, tandis que la seconde sert à récupérer cette donnée en utilisant le cache L1 comme canal caché. Cette seconde étape doit néanmoins être préparée avant le démarrage de la situation de compétition. Ce pourquoi, nous retrouvons trois phases dans Meltdown.

» Phase I

Lors d'un cycle de l'attaque, l'attaquant tente d'exfiltrer une donnée d'un octet, soit 8 bits. Il alloue alors un tampon mémoire de 256 sections qui servira d'oracle, afin de couvrir les 2^8 valeurs de l'octet, dont il maîtrise l'adresse en RAM.

```
# Allocation de l'oracle
char * oracle[256] ;
```

Il octroie à chaque section du tampon la taille d'une page, mettons 4096 octets = 4ko. Cette seconde condition permet d'éviter les faux positifs. En effet, suivant le principe de localité spatiale, non seulement le contenu de l'adresse accédée dépendant du secret sera mis en cache, mais également celui des adresses qui lui sont contiguës en RAM. Or, la présence en cache de ces adresses adjacentes risque de fausser l'interprétation des temps d'accès aux index de l'oracle. Ce biais de l'oracle est dû au fait que le temps d'accès à certaines adresses du tampon sera court, même si ces adresses n'ont pas été référencées lors de l'exécution transitoire et qu'elles sont complètement décorrélées de la valeur secrète recherchée. Pour éliminer le risque de faux positifs, une taille correspondant à une page mémoire est utilisée comme distance entre chaque index du tampon.

Le tampon oracle contiendra alors 256 segments de taille d'une page mémoire chacun, que nous considérons être à 4096 octets.

```
# Allocation de l'oracle
char * oracle[256*4096] ;
```

» Phase II

Après l'allocation du tampon oracle, le cache est vidé. La violation d'accès est ensuite effectuée par le déréférencement d'une adresse noyau, laissant place à l'exécution transitoire qui charge en cache un index du tampon oracle dépendant de la valeur secrète récupérée :

```
secret_value = *kernel_address ;
oracle[secret_value*4096];
```

Le tampon oracle est donc accédé à l'index `secret_value*4096` et réside par conséquent en cache.

» Phase III

Pour récupérer la valeur exfiltrée, l'attaquant parcourt le tampon oracle en calculant le temps d'accès à chaque `index*4096`. Contrairement aux accès aux index de l'oracle qui se trouvent en RAM, et qui mettront plusieurs tours d'horloge, l'accès à l'adresse `oracle+secret_value*4096` sera rapide, car l'adresse sera déjà présente dans le cache.

```
#%rdi : oracle
#%rsi : secret_ptr
movb(%rsi), %al
shl $12,%rax
movq (%rdi,%rax),%rdi
retq
```

NOTE

shl $12, %rax correspond à rax * 4096, le secret est multiplié par la taille de page pour éviter les faux positifs.

Résistance de SGX face à Meltdown

Les enclaves SGX d'Intel ont pu résister face aux attaques Spectre et Meltdown en début d'année grâce à l'implémentation du mécanisme d'*abort page semantics*.

Ce mécanisme matériel empêche tout accès extérieur illégitime à la mémoire de l'enclave indépendamment du niveau de privilège (ring3/espace utilisateur, ring0/noyau, SMM, VMM, autre enclave).

Le code -1 est retourné pour les accès en lecture, tandis que les accès en écriture sont ignorés.

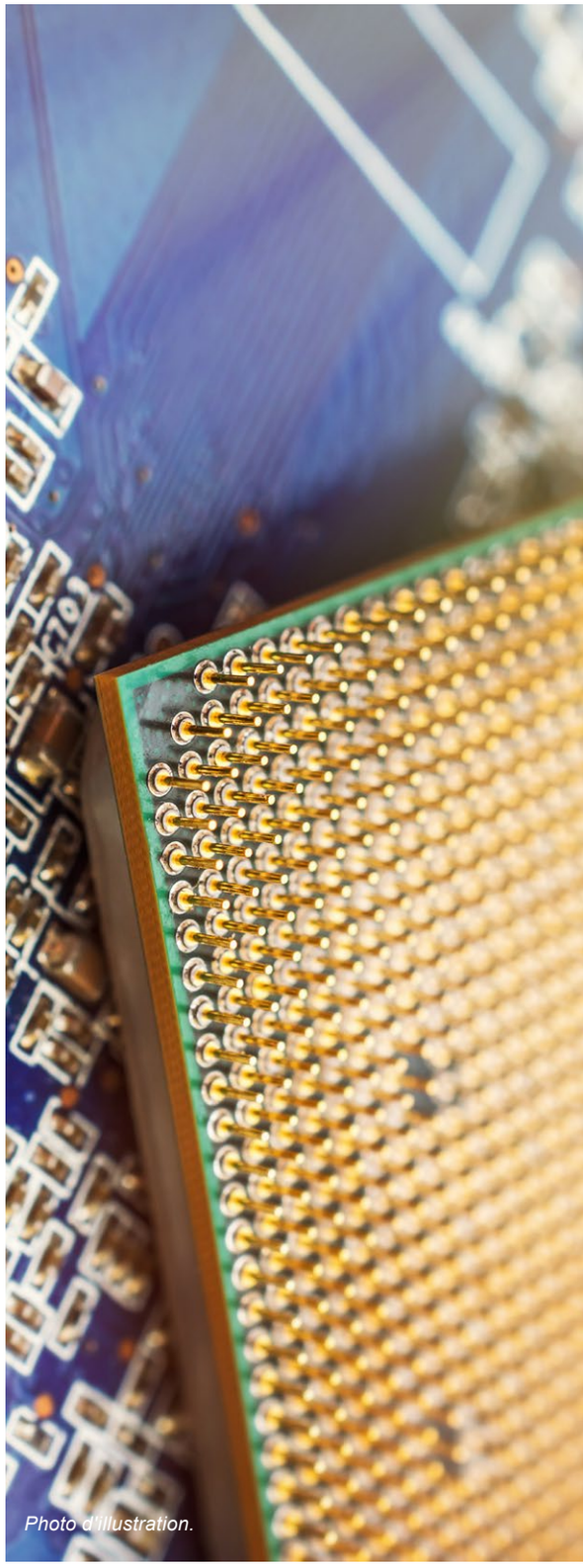


Photo d'illustration.

Dans son exploitation, Meltdown s'est heurté face à ce mécanisme d'Intel SGX. L'absence de faute de page suite à un accès à une adresse de l'enclave empêche la condition de course de se produire et rend donc toute exécution transitoire impossible. Tout attaquant tentant de déréférencer une adresse dans l'enclave n'arrive à obtenir que le code -1 envoyé par le mécanisme *abort page semantics*.

FORESHADOW

Introduction

Le 15 août 2018, Foreshadow, appelée par Intel L1TF pour L1 Terminal Fault, est rendue publique, bien que découverte et communiquée au fabricant Intel en janvier de la même année par deux groupes de chercheurs différents.

Cette nouvelle vulnérabilité impacte cette fois les CPU Intel SGX, jusqu'alors préservés contre Spectre et Meltdown. Des chercheurs ont réussi à compromettre toute la confidentialité et l'intégrité conférées aux enclaves par Intel SGX en exfiltrant des données de l'enclave, y compris les clés de chiffrement.

Bien que des recherches ont pu démontrer auparavant qu'il était possible de compromettre la sécurité de SGX, ces attaques nécessitaient un niveau de privilège élevé ou encore que le code exécuté dans l'enclave soit vulnérable.

Foreshadow est, quant à elle, une attaque menée depuis l'espace utilisateur sans exploiter de code vulnérable dans l'enclave, ni nécessiter de gadget dans des bibliothèques de fonctions.

Par opposition à Meltdown qui s'est intéressée à l'accès à l'espace d'adressage noyau depuis l'espace utilisateur, Foreshadow cible les enclaves SGX. Dans le modèle d'attaque des enclaves, l'attaquant peut même disposer de tous les droits sur un système d'exploitation. Dans ce sens, des techniques d'optimisation de Foreshadow ont été publiées avec un attaquant privilégié. Toutefois l'attaque basique présentée par la suite, est exécutée depuis l'espace utilisateur.

Même si Meltdown n'y est pas parvenu, Foreshadow, ou L1TF, démontre encore une fois, grâce à un nouveau vecteur d'attaque, qu'il est possible d'exploiter les optimisations CPU pour attaquer les enclaves SGX.

Contournement du mécanisme abort page semantics

Si les enclaves SGX vivent dans l'espace d'adressage utilisateur, l'isolation de leur mémoire physique est strictement renforcée au niveau matériel. Comme vu précédemment, le code et les données de l'enclave sont chiffrés quand elles sont en mémoire physique, et tout accès depuis l'extérieur de l'enclave est rejeté par l'implémentation du mécanisme '*abort page semantics*'.

Cependant, avant d'appliquer cette procédure, Intel SGX s'assure de la légitimité de la table des pages de l'enclave. L'implémentation de cette couche supplémentaire d'isolation matérielle est réalisée par Intel SGX, qui considère que le procédé de translation d'adresse est en dehors de sa zone de confiance. Ainsi, si lors de la vérification de la table des pages de l'enclave une anomalie est détectée, une faute de page est levée, engendrant un changement de contexte pour rendre la main au système d'exploitation. Cette nouvelle levée de faute offre une nouvelle fenêtre de temps pour l'exécution transitoire de Foreshadow, qui rejoint Meltdown dans sa méthodologie d'exploitation. Si Meltdown s'est heurté au mécanisme d'*abort page semantics*, Foreshadow empêche ce mécanisme de se produire en trouvant un moyen de lever une faute avant. Meltdown et Foreshadow utilisent, toutes les deux, l'exécution transitoire pour exfiltrer une donnée secrète, ainsi que l'attaque par canal auxiliaire

FLUSH+RELOAD pour récupérer le secret. Néanmoins, si Meltdown visait des adresses du noyau, Foreshadow requiert que le secret de l'enclave soit déjà résidant en cache L1, et donc doit d'abord exécuter l'enclave. Ainsi, la préparation de l'exécution transitoire souligne certaines différences entre les deux exploitations.

Exploitation

» Phase I : Préparation de l'exécution transitoire

Dans un cycle d'attaque, l'attaquant va tenter d'exfiltrer un octet de donnée.

- Allocation d'un tampon oracle en mémoire de 256 segments de taille 4Ko (1) (cf. fig.3).

```
# Allocation de l'oracle
char * oracle[256*4096] ;
```

- Exécution de l'enclave, afin de ramener les secrets en cache L1 (2) (cf. fig.3).
- Révocation de toutes les permissions d'accès à la page de l'enclave. Cette révocation est réalisée via l'appel système **mprotect** qui retire le bit « présent » de toutes les pages de l'enclave (3) (cf. fig.3).

```
mprotect(secret_ptr &~0xfff, 0x1000, PROT_NONE) ;
```

Cette étape est importante pour Foreshadow, car elle permettra la levée d'une faute de page pour chaque accès et ouvrira la voie pour la situation de compétition. À ce stade, le déréférencement d'une adresse de l'enclave provoquera une faute de page, pour laisser place à l'exécution transitoire au lieu d'être confronté au mécanisme d'*abord page semantics*.

- Exécution de l'instruction **clflush** sur les 256 segments de l'oracle. Néanmoins, si Foreshadow arrive à vaincre la barrière imposée par l'*abort page semantics* de SGX, toute entrée/sortie à l'enclave implique une remise à zéro de la TLB. Si cela ne posait pas de problème pour Meltdown, dans le cas de SGX l'exécution transitoire démarre avec une TLB vide. En considérant la fenêtre de temps restreinte pour l'exécution transitoire, l'accès à l'index de l'oracle correspondant au secret impliquerait une nouvelle translation d'adresse pour calculer son adresse physique.

  Ainsi, le processus chronophage de translation d'adresse pourrait causer la perte de la situation de compétition, et l'exécution transitoire n'arriverait jamais à l'instruction exfiltrant le secret.

Pour résoudre cette nouvelle problématique, Foreshadow rétablit les 256 adresses des segments de l'oracle dans la TLB, en flushant chaque segment de l'oracle avec l'instruction **clflush (4)** (cf. fig.3). Cette instruction permet également de vider le cache des segments de l'oracle évitant ainsi les faux positifs lors du **FLUSH+RELOAD**.

» Phase II : Exécution transitoire

Déréférencement de l'adresse dans l'enclave et mise en cache de : **oracle[secret*4096]** (cf. Fig.3 (5) et (6)).

```
#%rdi : oracle
#%rsi : secret_ptr
movb(%rsi), %al
shl $12,%rax
movq (%rdi,%rax),%rdi
retq
```

» Phase III : Réception du secret

Lorsque le processeur détecte la violation d'accès de l'exécution transitoire, il lève une exception matérielle. L'attaquant pourra alors exécuter le code lui permettant de réceptionner le secret depuis le cache par le biais du gestionnaire d'exception. Il mesure le temps d'accès à chaque segment de l'oracle, et déduit l'adresse relative au secret par son temps d'accès réduit.

Conséquences de Foreshadow

Le modèle de confiance d'Intel SGX établit qu'il est possible de faire exécuter un programme par un tiers en assurant au propriétaire la confidentialité et l'intégrité de son code.

Néanmoins, le composant MEE de SGX échoue face à Foreshadow, dans sa responsabilité de protection de la mémoire de l'enclave en considérant les deux premiers niveaux de cache comme zones de confiance, et donc en dehors de son périmètre de chiffrement.

L'exploitation de Foreshadow démontre qu'il est non seulement possible d'exfiltrer des données de l'enclave, mais également d'en exfiltrer la totalité des clés cryptographiques sur

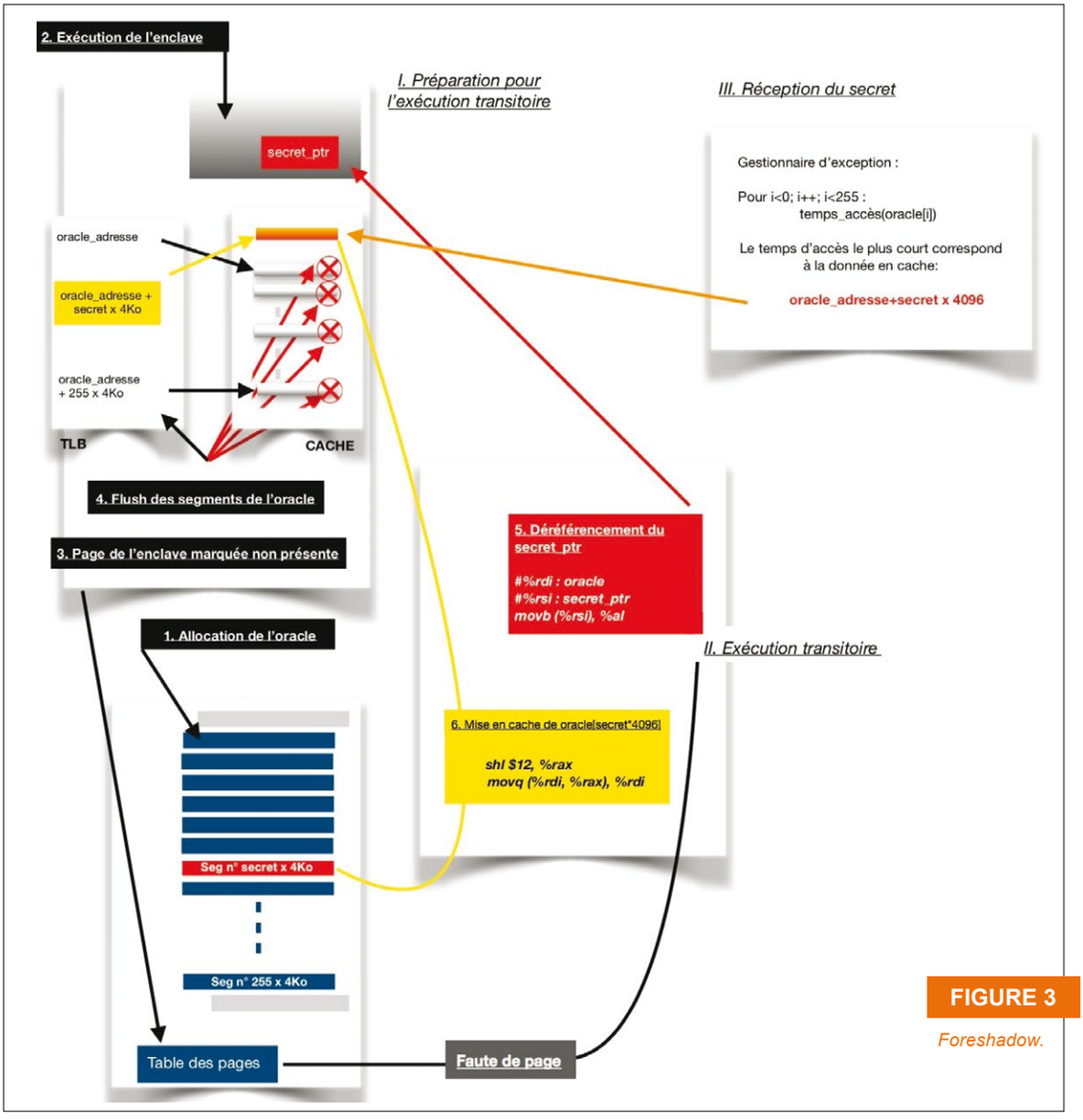


FIGURE 3

Foreshadow.

lesquelles reposent SGX. Ainsi, l'attaque **[8]** arrive à contourner le modèle d'attestation à distance **[9]** qui permet aux enclaves d'apporter la preuve de leur identité et de leur intégrité ainsi que de garantir qu'elles sont bien exécutées sur une plateforme où SGX est activé.

Par ailleurs, l'investigation d'Intel sur Foreshadow a permis de révéler que l'exploitation n'affecte pas seulement les enclaves. La nouvelle génération de Foreshadow (Foreshadow-NG) **[10]** permet d'exfiltrer toutes les données du cache L1 grâce à l'exécution transitoire dans le désordre. Ces données comprennent celles du système de gestion mémoire SMM (*System Memory Management*), des noyaux de systèmes d'exploitation ainsi que des hyperviseurs.

Globalement, toute l'isolation apportée par la virtualisation peut être détruite.

CONCLUSION

L'année 2018 aura été marquée par les vulnérabilités matérielles ciblant les processeurs. Nous assistons une nouvelle fois à un dilemme où la fonctionnalité et la performance ont primé sur la sécurité.

Si la sophistication des optimisations CPU a porté la puissance de calculs des processeurs au niveau supérieur, elle n'a pas manqué de leur introduire des vulnérabilités désormais inhérentes à leur fonctionnement.

Ces vulnérabilités propres à la conception de la micro-architecture des processeurs, ouvrent un nouveau champ de bataille pour les attaquants. Le TEE et la virtualisation ont pu voir cette année leur sécurité logicielle et matérielle compromise par des attaques par canal auxiliaire. Plus alarmant encore, les remédiations habituelles visant à corriger du code vulnérable ne sont plus applicables. Elles nécessitent désormais de repenser le fonctionnement des CPU.

Après Spectre et Meltdown, Foreshadow vient à son tour mettre à mal la sécurité des environnements d'exécution sécurisés.

Un constat qui pousse à croire que l'isolation complète d'une exécution ne peut être achevée que si elle est accompagnée par une isolation matérielle.

Tant que plusieurs threads continueront à s'exécuter sur le même cœur de processeur et à partager les mêmes ressources internes du CPU, de nouvelles attaques par canaux auxiliaires sur les caches ne cesseront de proliférer.

Afin de se prémunir totalement de ce type d'attaque, sans pour autant brider les performances, une refonte complète de la micro-architecture des processeurs doit être envisagée, chose que nous ne risquons pas de voir de si tôt. ▌

RÉFÉRENCES

[1] **https://foreshadowattack.eu/**

[2] **https://fr.wikipedia.org/wiki/Microarchitecture**

[3] **https://fr.wikipedia.org/wiki/Core_(microarchitecture)**

[4] **https://en.wikipedia.org/wiki/Trusted_execution_environment**

[5] **https://www.usenix.org/legacy/event/sec08/tech/full_papers/halderman/halderman.pdf**

[6] **https://software.intel.com/en-us/blogs/2016/02/26/memory-encryption-an-intel-sgx-underpinning-technology**

[7] **https://eprint.iacr.org/2013/448.pdf**

[8] **https://software.intel.com/en-us/sgx-sdk-dev-reference-remote-attestation**

[9] **https://foreshadowattack.eu/foreshadow.pdf**

[10] **https://foreshadowattack.eu/foreshadow-NG.pdf**

ACTUELLEMENT DISPONIBLE

MISC N°101

SÉCURITÉ DES APPLICATIONS WEB

NE LE MANQUEZ PAS

CHEZ VOTRE MARCHAND DE JOURNAUX ET SUR :

https://www.ed-diamond.com

COMPRENDE LES VULNÉRABILITÉS DE L'IOT

AU SOMMAIRE :

La sécurité des objets connectés est une vaste thématique tant ce qui est couvert par cette dénomination représente un peu près tout ce qui est désormais connecté et ne l'était pas : montre, voiture, brosse à dents, etc. Ce qui a réellement changé, c'est la possibilité de pouvoir s'interfacer facilement au travers de protocoles connus et d'équipements accessibles financièrement (merci la radio logicielle).

Pour les habitués de la sécurité, on pourrait presque avoir le sentiment qu'il n'y a rien de nouveau si ce n'est quelques protocoles de télécommunication dédiés, quelques nouvelles interfaces, qu'il ne s'agit que d'embarqué. Ce sentiment est loin d'être faux, tant les problèmes rencontrés ressemblent à ceux d'il y a 15 ans. Pour ne prendre que comme exemple le top 10 OWASP IoT 2018, les 4 premiers sont : mots de passe, service réseau non sécurisé, écosystème d'interfaces non sécurisé, absence de mécanisme de mise à jour. Bref, hardcoder le mot de passe root n'est pas une bonne idée, il faut déployer correctement https, faire de l'authentification, utiliser l'état de l'art en cryptographie, avoir un mécanisme de mise à jour sécurisé.

Mais considérer que la sécurité de l'IoT n'est que la répétition de ce qui a déjà était fait n'est pas suffisant. La très grande diversité du domaine a produit son lot de nouvelles menaces, de nouvelles vulnérabilités et tout un nouvel écosystème d'outils. Il y a même un CTF entièrement dédié à l'IoT : ph0wn.

Le hors-série numéro 15 de *MISC* s'était intéressé à la sécurité des objets connectés avec une approche généraliste en traitant des différentes pistes qui permettent d'améliorer la sécurité du domaine : cryptographie à bas coût, micro-noyau prouvé dédié à l'IoT, sécurité de protocoles comme LoRaWAN, etc.

Le numéro que vous tenez entre vos mains, ou lisez sur votre écran, traite des vecteurs d'attaque des objets connectés. Bien des cas traités ici ne sont pas spécifiques à l'IoT, mais ont une importance capitale pour ces derniers. Vous retrouverez ainsi des articles sur les bus CAN, le dump de firmware, les attaques sur les protocoles de la téléphonie mobile et, enfin, terminerez sur une analyse de trackers GPS et une présentation des techniques de leurrage GPS en utilisant la radio logicielle.

Bonne lecture !

Émilien GASPAR / gapz / eg@miscmag.com

BUS CAN : SE LANCER DANS L'ANALYSE DES COMMUNICATIONS DE VOTRE VÉHICULE

Sébastien MAINAND

Cet article a pour but de s'intéresser aux communications internes d'un véhicule. Il sera présenté comment réaliser sa propre interface CAN/USB basée sur un Arduino, puis sera décrite la manière d'analyser et d'injecter des paquets à l'aide d'outils d'analyse réseau classiques.

Photo d'illustration.

L'amélioration de la sécurité de nos véhicules, en plus d'être un sujet d'actualité, est un domaine qui nous concerne tous. Nous sommes quotidiennement en contact avec ces « objets », dans lesquels la part d'informatique ne cesse d'augmenter. En effet, l'ajout de nombreuses options de confort, d'aide à la conduite ou encore de protection des passagers s'accompagne d'autant de calculateurs, exécutant du code ayant un réel impact sur le fonctionnement du véhicule. L'étude de leur sécurité informatique par le plus grand nombre en est donc d'autant plus importante. Elle pourra concerner autant l'analyse de leur surface d'attaque que la robustesse de leurs calculateurs, à travers le réseau interne du véhicule.

BIEN COMMENCER

Choisir son sujet d'étude

Si vous avez à disposition un véhicule, il faudra valider qu'un bus CAN est bien utilisé pour ses communications internes. Cela devrait être le cas pour des véhicules européens relativement récents, âgés d'une dizaine d'années, mais une rapide recherche sur Internet pour s'en assurer est toujours la bienvenue. Vous pourrez en profiter pour localiser le port OBD-II de votre véhicule, présent dans l'habitacle. Il devrait vous donner un accès direct aux trois signaux du bus CAN, à savoir CANH, CANL et GND. Les broches correspondantes sont identifiées sur la figure 1.

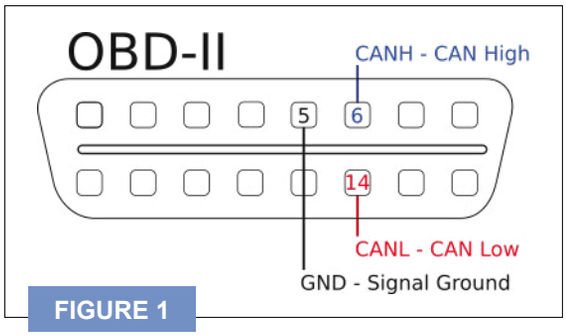


FIGURE 1

Schéma d'une prise OBD-II et identification des broches permettant l'accès au bus CAN.

Notez qu'une passerelle peut être présente entre la prise OBD-II et le bus interne sur les modèles les plus récents. Dans ce cas, l'accès au réseau sera restreint, et seules les trames de diagnostic seront autorisées. Pour obtenir un accès complet au bus CAN, il sera alors nécessaire de se brancher derrière la passerelle, impliquant cependant des modifications bien plus intrusives sur le véhicule.

NOTE

Si le sujet est complètement nouveau pour vous, je vous renvoie vers l'excellent article du *MISC* n°97 [MISC97] pour une introduction sur le sujet.

Si l'accès à un véhicule n'est pas envisageable, il existe dans le commerce des cartes permettant d'émuler un calculateur, ou ECU pour « Electronic Control Unit ». Pour un coût d'environ 250 €, le simulateur « ECUSim 2000 OBD Simulator » de **www.scantool.net** permet d'interagir avec trois calculateurs, émulés derrière un port OBD-II. Il permet de choisir entre deux fréquences de bus (250 ou 500 kbit/s), d'alterner entre les deux formats d'identifiants CAN (11bits en mode normal ou 29 bits en mode étendu) et de simuler la présence de codes de diagnostic. Il est à noter qu'aucun trafic « interne » ne sera présent en vous y connectant, ressemblant ainsi à la situation décrite précédemment, où le bus CAN est protégé par une passerelle.

Dans ce cas, il sera nécessaire de générer vous-même des échanges en envoyant à l'ECU (virtuel ou réel) des requêtes de diagnostic. Pour la suite de cet article, il sera considéré que nous nous trouvons dans cette situation.

S'équiper d'une sonde de diagnostic et de câbles OBD-II

Une fois le sujet d'étude trouvé, il est intéressant d'acquérir une sonde de diagnostic OBD-II dans le commerce. En plus de pouvoir vous en servir pour diagnostiquer les pannes de votre voiture (lorsque le voyant moteur ou MIL, pour « Malfunction Indicator Lamp » s'allume sur le tableau de bord), il sera un équipement de référence permettant de générer du trafic CAN légitime. J'ai pour ma part testé la sonde « OBDLink SX USB », intéressante pour son prix d'environ 30 € et pour la licence logicielle d'OBDWiz qui lui est associée. Tout autre équipement pourra faire l'affaire, et si aucune licence logicielle n'y est adjointe, le logiciel ScanTool sous GNU/Linux pourra être utilisé.

Vous pouvez d'ores et déjà la tester sur votre véhicule ou en utilisant un émulateur. Les fonctions de base vous permettront de relever des PID (*Parameters ID*) sur le véhicule, de récupérer et d'acquitter des messages d'alerte, comme illustrés sur la figure 2.

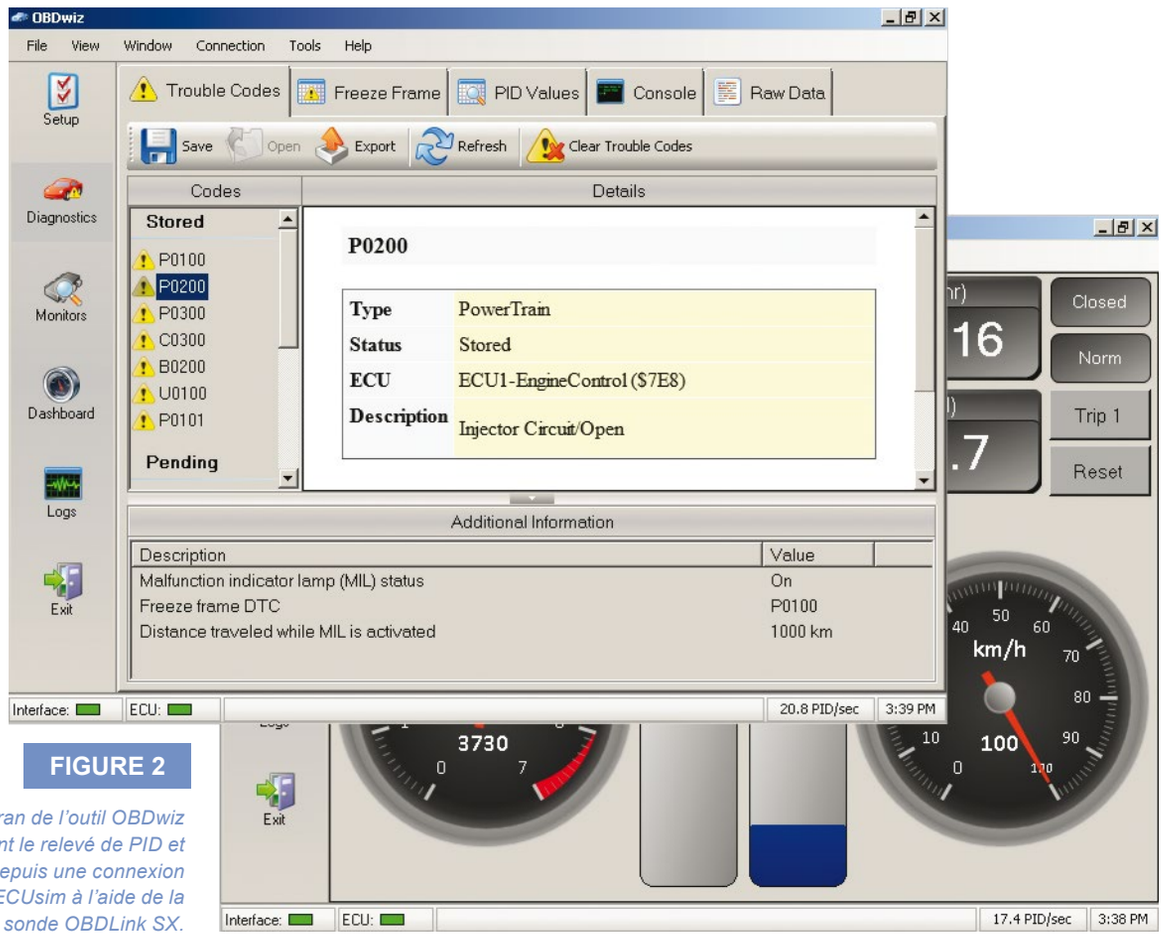


FIGURE 2

Captures d'écran de l'outil OBDwiz illustrant le relevé de PID et d'alarmes depuis une connexion à une carte ECUsim à l'aide de la sonde OBDLink SX.


Photo d'illustration

Vous pouvez dans la foulée commander un ensemble de câbles. Il est intéressant d'acquérir tout d'abord une rallonge OBD-II (prise mâle vers femelle) : elle pourra être utilisée pour se connecter avec plus d'aisance dans un véhicule avec votre sonde de diagnostic. Dans la suite de l'article, elle sera également découpée afin de s'interconnecter simplement avec votre sonde de test. Enfin, il est utile d'acquérir un adaptateur OBD-II vers DU-9 afin d'obtenir une connectique standard entre votre sonde de test et un port de diagnostic. Attention cependant à la convention de câblage côté DU-9, car il en existe une dite UK et une dite US. Pour information, c'est cette dernière qui est utilisée dans le matériel choisi pour la suite. Ces connectiques peuvent s'obtenir pour une dizaine d'euros chacune.

ASSEMBLER SON INTERFACE CAN

Partie matérielle

Maintenant que vous disposez d'un bus opérationnel, animé a minima par des échanges de diagnostic, il ne vous reste plus qu'à réaliser votre sonde de travail CAN, permettant d'écouter et d'injecter librement du trafic. Elle sera assemblée à base de composants électroniques de faible coût, de logiciels open source et permettra de travailler confortablement depuis un ordinateur sous GNU/Linux avec les outils Wireshark et Scapy.

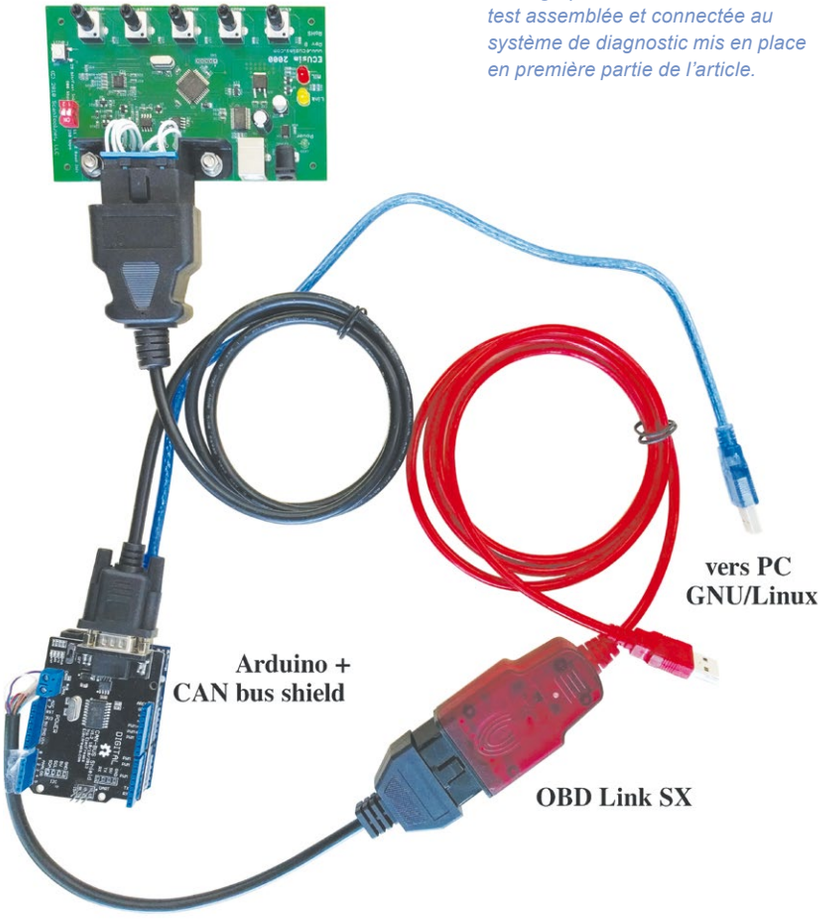
Les différents composants nécessaires sont les suivants :

- un ordinateur sous GNU/Linux (Ubuntu 18.04 LTS 64 bits) ;
- un Arduino Uno ;
- un Arduino « CAN bus shield » ;
- le câble USB de l'Arduino, qui servira d'alimentation et de lien de communication avec l'ordinateur ;
- du matériel d'électronique standard, tel qu'un poste à souder (afin d'assembler éventuellement les broches de votre « CAN bus shield »), et de quoi couper, dénuder et identifier (multimètre) les fils d'un câble OBD-II.

Il existe différents vendeurs proposant chacun leur version de « CAN bus shield ». Les caractéristiques à valider en sont la fréquence de l'oscillateur (16 MHz), la présence de deux ports CAN, dont un en DU-9, et de la convention de câblage de ce dernier. La carte utilisée pour les tests est celle produite par ElecFreak, et coûte une vingtaine d'euros.

Il ne vous reste plus alors qu'à assembler le tout. Une image valant mieux qu'un long discours, le résultat final est visible sur la figure 3, page suivante.

FIGURE 3



Photographie de la sonde de test assemblée et connectée au système de diagnostic mis en place en première partie de l'article.

L'Arduino est raccordé à un ordinateur par son port USB et protégé par son « CAN bus shield ». Ce dernier est branché via son port DU-9 à la prise OBD-II du système, et via son port CAN « deux fils » à la sonde de diagnostic, à l'aide de la rallonge OBD-II préalablement découpée.

Pour ce qui est de l'identification des fils du câble OBD-II une fois découpé, il n'existe malheureusement pas de raccourci. Les couleurs variant d'un modèle à l'autre, il vous faudra les tester un par un avec un multimètre en mode « testeur de continuité » pour retrouver les deux correspondants aux broches 6 et 14 de la prise.

L'avantage de cette solution, par rapport à un équipement du commerce, est de pouvoir d'une part maîtriser l'intégralité de la chaîne, et d'autre part de pouvoir mettre à contribution un Arduino délaissé depuis des années dans le fond d'un tiroir !

Les deux interfaces CAN du « CAN bus shield » étant reliées, votre sonde de test est par défaut conductrice électriquement. Le plus simple pour valider votre câblage est de renouveler l'expérience avec votre sonde de diagnostic. Veillez bien entendu à ne pas alimenter (connecter le câble USB) de l'Arduino pour réaliser ce test.

Partie logicielle

Dans notre configuration, l'Arduino, à travers sa liaison série USB, va servir de passerelle entre le bus CAN et un socket de type `PF_CAN` sur l'ordinateur. Pour lui faire remplir cette tâche, il suffira de mettre un œuvre une bibliothèque prévue à cet usage.

ATTENTION !

Un véhicule est un système complexe, non conçu pour résister à des attaques informatiques, et dont le dysfonctionnement peut avoir de graves conséquences sur son système et son environnement. Si vous expérimentez directement sur l'un d'eux, toutes les précautions envisageables, même celles pouvant paraître excessives, sont les bienvenues. Toute interaction est laissée à votre discrétion et engage votre responsabilité.

Le chargement du programme de l'Arduino sera fait avec le cadriciel PlateformIO **[PIO]**. Les lignes de commandes pour la mise en place du projet sont présentées ci-dessous et ont été testées sur une distribution Ubuntu 18.04 LTS 64 bits.

```
# Récupération des dépendances
sudo apt install python2.7 python-virtualenv
mkdir tmp && cd tmp
git clone https://github.com/latonita/arduino-canbus-monitor
cd ..

# Installation de PlatformIO dans un virtualenv Python
mkdir venv
python2.7 -m virtualenv venv/pio
source venv/pio/bin/activate
(pio) pip install platformio

# Création du projet Arduino
(pio) pio init
(pio) cp -r tmp/arduino-canbus-monitor/arduino-canbus-monitor/ lib/
(pio) mv lib/arduino-canbus-monitor/arduino-canbus-monitor.ino src/main.cpp
```

PlateformIO a besoin d'informations sur la carte à programmer, à préciser dans le fichier `platformio.ini` :

```
(pio) cat platformio.ini
[env:megaatmega2560]
platform = atmelavr
board = uno
framework = arduino
monitor_speed = 115200
```

Enfin, quelques paramètres sont à éditer et valider dans le fichier `src/main.cpp` : la fréquence du lien série vers l'Arduino, celle du bus CAN et celle de l'oscillateur du « CAN bus shield » utilisé.

```
# src/main.cpp
Serial.begin(LW232_DEFAULT_BAUD_RATE); // default COM baud rate is 115200.

//[...]

// EDIT HERE YOUR CAN BUS FREQUENCY
Can232::init(CAN_250KBPS, MCP_16MHz);
```

Pour déterminer la fréquence du bus, plusieurs solutions s'offrent à vous :

- elle est déjà connue, car précisée dans une documentation ou sélectionnée dans le cas de l'ECUSim ;
- votre matériel de diagnostic permet de la déterminer automatiquement ;
- en essayant les fréquences les plus courantes (cf. commentaires du fichier `src/main.cpp`).

Vient ensuite la phase de chargement du programme à proprement parler. Veillez bien à vous déconnecter du bus CAN, puis connectez l'Arduino à votre ordinateur avec le câble USB. Il devrait être disponible derrière le périphérique `/dev/ttyACM0` (identifiable avec la commande `dmesg | grep tty`). Les lignes de commandes suivantes peuvent être utilisées pour compiler et charger le programme. Une connexion Internet est requise pour l'exécution de la première commande, car PlatformIO récupère automatiquement les dépendances nécessaires à l'opération.

```
(pio) pio run
(pio) pio run -t upload
```

NOTE

Il sera nécessaire d'installer la règle udev `99-platformio-udev.rules` pour que le chargement de programme fonctionne. Le lien vers la commande adéquate vous sera indiqué lors de la première exécution de la commande. Une fois réalisé, il sera ensuite nécessaire de débrancher/rebrancher l'Arduino puis de réitérer l'opération.

Après toutes ces étapes, il est enfin temps de se lancer en se connectant au bus CAN. On constatera que le système fonctionne bien au clignotement de la LED TX de l'Arduino, indiquant qu'un message du bus CAN a été transféré à l'ordinateur. Il est également possible d'observer directement les messages séries envoyés par l'Arduino, via la commande suivante :

```
(pio) pio device monitor
```

Reste maintenant à mettre en place votre socket réseau. Les commandes permettant d'installer les dépendances nécessaires pour sa création, ainsi que les outils qui seront utilisés par la suite, sont les suivantes :

Photo d'illustration.

```
(pio) deactivate
sudo apt install can-utils
sudo apt install wireshark python3
git clone https://github.com/secdev/scapy
```

« Can-utils » est la bibliothèque installant le projet SocketCAN, fournissant le support pour les sockets `PF_CAN` ainsi qu'un ensemble d'outils en ligne de commandes permettant d'interagir avec lui. Le code source est consultable directement sur la page **[CAN-UTILS]**. La version upstream de Scapy, via python 3, sera utilisée exclusivement afin d'obtenir les dernières évolutions concernant le support du protocole CAN, dont l'ajout est relativement récent. Pour éviter d'avoir à lancer Wireshark avec les droits administrateur, il est intéressant d'ajouter son utilisateur courant dans le groupe « wireshark » via la commande `sudo usermod -aG wireshark ${USER}` (effectif après avoir rouvert votre session).

Les messages séries envoyés par l'Arduino devront être convertis au format `PF_CAN` pour être disponibles depuis Wireshark ou Scapy. Cela sera réalisé par le daemon `slcand`, en exécutant par exemple le script suivant :

```
sudo modprobe can vcan slcan
sudo slcand -o -s5 -t hw -S 115200 /dev/ttyACM0 &
sudo ip link set up slcan0
```

Il sera nécessaire d'adapter la ligne de commandes de `slcand` aux propriétés de votre bus CAN et de votre configuration. L'option `-S` devra correspondre à la fréquence du bus série de communication avec l'Arduino, et le paramètre `-s5` à la fréquence du bus CAN (ici 250 kbit/s). On notera les valeurs `-s4` et `-s6` pour les fréquences les plus courantes, à savoir 125 kbit/s et 500 kbit/s respectivement. La page **[SLCAND]** permet d'obtenir tous les détails de configuration.

Veillez bien à ce qu'une seule instance de `slcand` ne soit lancée à la fois. Une commande analogue à `sudo pkill -f slcand` pourra être utilisée le cas échéant.

À la suite de ce script, si tout a bien fonctionné jusque-là, une nouvelle interface nommée `slcan0` est apparue, et est utilisable depuis Wireshark et Scapy comme une interface réseau classique ! Tout est maintenant prêt pour commencer à interagir avec votre bus CAN.

QUELQUES EXEMPLES D'UTILISATION

Écoute réseau

Un premier exercice trivial est de capturer et d'analyser une commande de diagnostic OBD-II, récupérant la valeur instantanée du PID « vitesse du véhicule ». Pour ce faire, il suffit de mettre Wireshark en écoute sur l'interface `slcan0` apparue précédemment, et de lancer, via votre logiciel de diagnostic (pour rappel OBDwiz ou ScanTool), la récupération périodique de la vitesse instantanée du véhicule. Le résultat de la capture est visible sur les figures 4 et 5 (pages suivantes) respectivement pour la requête et la réponse.

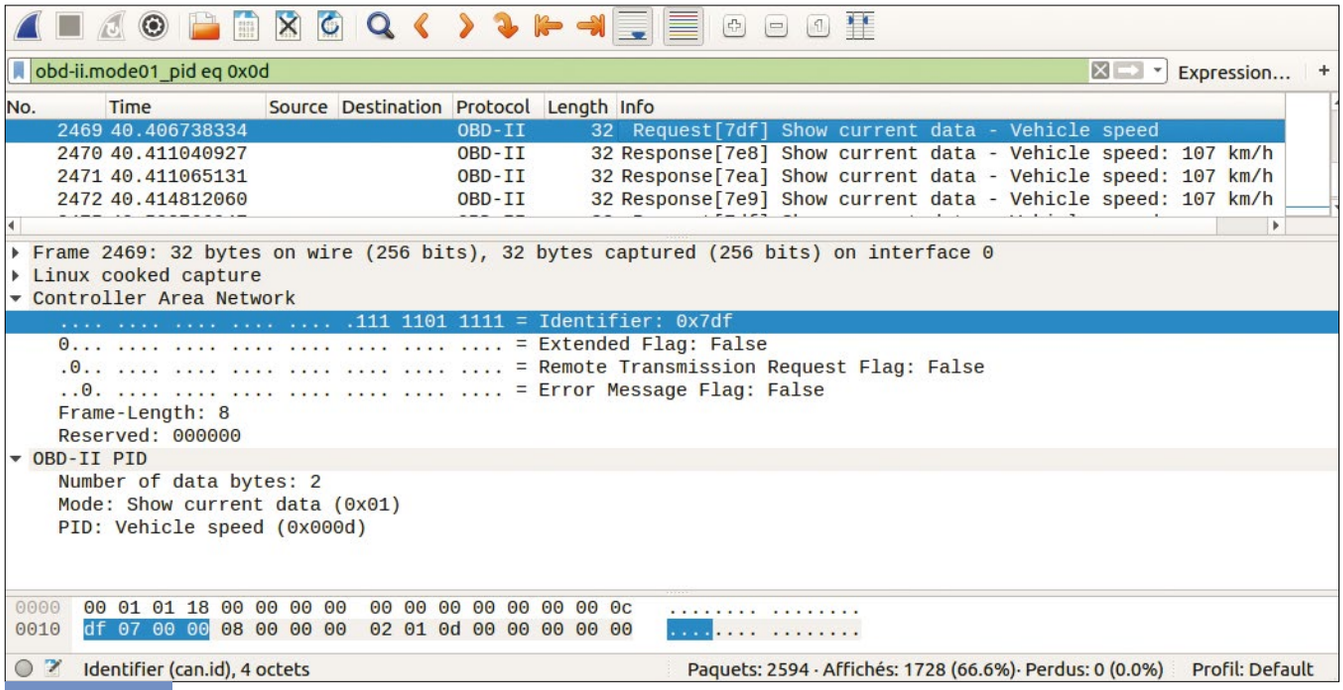


FIGURE 4 *Visualisation avec Wireshark d'un paquet demandant la valeur de vitesse instantanée du véhicule.*

Comme défini dans la norme OBD-II, la sonde de diagnostic utilise un identifiant prédéfini qui est `0x7df`. Le protocole CAN ne définit que le niveau de liaison de données (niveau 2 OSI), et laisse huit octets maximum pour le transfert de données. Dans notre cas, le protocole OBD-II est utilisé pour les échanges, et il faudra activer le dissecteur correspondant en faisant un clic droit sur la ligne ***Data*** et en sélectionnant ***Decode as > CAN next level dissector - OBD-II***.

INSTALLER SCAPY

Le support du protocole CAN étant récent dans Scapy, il est préférable d'utiliser la version upstream de l'outil. Pour ce faire et sans pour autant à avoir à l'installer sur votre système, vous pouvez charger la bibliothèque directement récupérée sur GitHub [SCAPY] et l'utiliser dans vos scripts en ajoutant en en-tête :

```
import sys
sys.path.insert(0, 'scapy/')
```

De plus, veillez bien à toujours utiliser la version python 3 en appelant vos scripts avec la commande :

```
python3 <mon_script.py>
```

Il existe en effet une distinction dans l'implémentation du support CAN en python 2, qui nécessite l'installation d'une dépendance additionnelle.

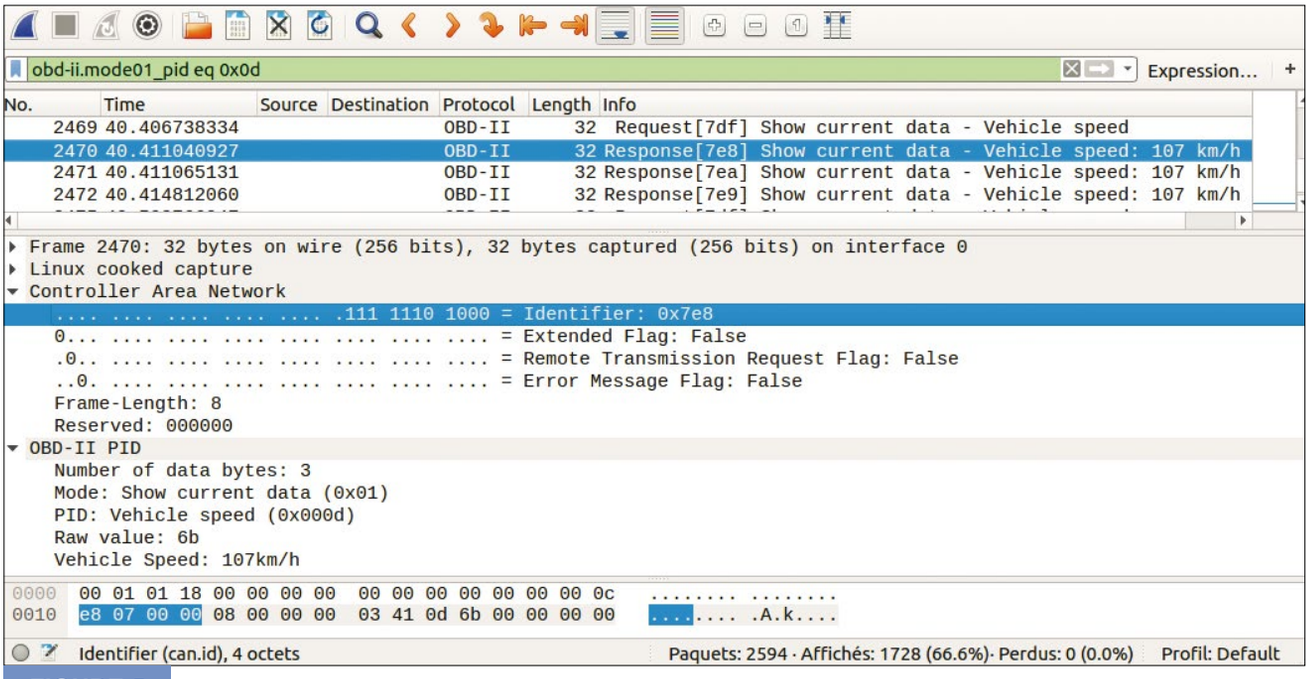


FIGURE 5 *Visualisation avec Wireshark d'un paquet retournant la valeur de vitesse instantanée du véhicule.*

La taille des données étant assez réduite, et si le dissecteur ne fonctionne pas avec votre version de Wireshark, il est également possible de filtrer les paquets manuellement en se basant sur la valeur de certains octets. Ainsi, un filtre `can[10] eq 0x0d` pourra être utilisé pour obtenir un résultat similaire au filtre utilisé dans les figures 4 et 5, à savoir `obd-ii.mode01_pid eq 0x0d`.

La même chose peut bien entendu être réalisée directement avec Scapy.

Le script suivant peut être utilisé afin d'enregistrer dans un fichier `can.pcap` les messages capturés pendant une durée de 5 secondes :

```
# [...]
# sniff_can_scapy.py

from scapy.all import *
from scapy.contrib.cansocket_native import *

interface = 'slcan0'
sniff_time = 5  # en secondes
out_pcap_filename = 'can.pcap'

socket = CANSocket(iface=interface)
print('Début de l'enregistrement du trafic CAN pendant {}
s.'.format(sniff_time))
packets = socket.sniff(timeout=sniff_time)

wrpcap(out_pcap_filename, packets)
print('Le trafic CAN a été enregistré dans le fichier {}'.format(out_
pcap_filename))
```

Vous pourrez ensuite l'ouvrir directement dans Wireshark, et constater que la couche « Linux Cooked capture » que l'on avait précédemment n'est plus présente. Cela n'a pas d'incidence sur les expérimentations.

Injection de paquets

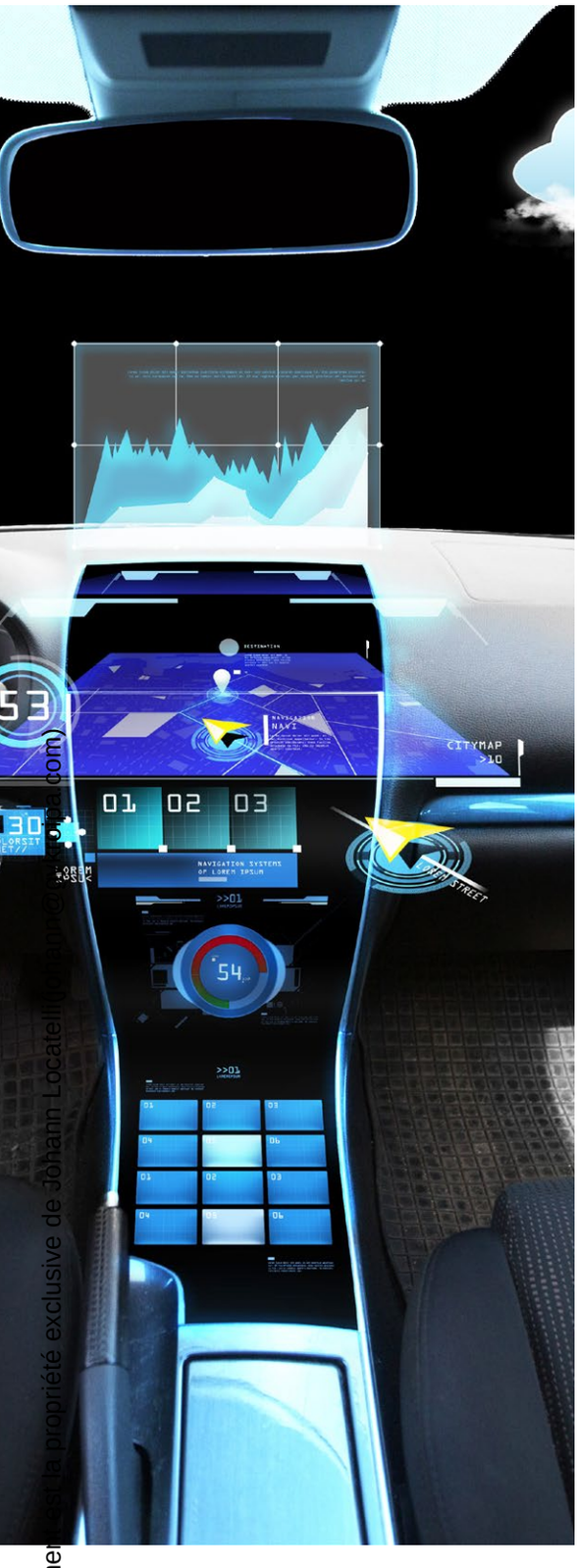
De la même manière que l'on a pu écouter et exporter du trafic au format PCAP, il va être possible d'en injecter. Le script suivant illustre la manière de forger un message CAN et de l'envoyer sur le réseau :

```
# [...]
# send_forged_can_scapy.py

from scapy.all import *
from scapy.contrib.cansocket_native import *

interface = 'slcan0'
sniff_time = 5  # en secondes
out_pcap_filename = 'can.pcap'

print('Création du paquet de lecture du PID vitesse')
pkt = CAN(identifier=0x7df, length=3, data=b'\x02\x01\x0d')
pkt.show()

print('Création d'un socket CAN et envoi')
socket = CANSocket(iface=interface)
response = socket.sr1(pkt)
response = CAN(bytes(response))
speed = response.data[3]
print('La vitesse de véhicule est de {} km/s.'.format(speed))
```

La même chose peut être réalisée en utilisant des messages extraits d'un fichier PCAP. Il est supposé ici que le message à envoyer est présent en première position dans le PCAP enregistré précédemment :

```
# [...]
# replay_can_pcap_scapy.py

from scapy.all import *
from scapy.contrib.cansocket_native import *

interface = 'slcan0'
in_pcap_filename = 'can.pcap'

print('Rejeu d'un fichier pcap')
list_pkt = rdpcap(in_pcap_filename)

socket = CANSocket(iface=interface)
response = socket.sr1(list_pkt[0][CAN])
```

Photo d'illustration.



```
response = CAN(bytes(response))
speed = response.data[3]
print('La vitesse de véhicule est de {}
km/s.'.format(speed))
```

Il est bien entendu possible de surveiller l'exécution de ces scripts en lançant au préalable Wireshark. Il est à noter que les paquets envoyés apparaissent en double sur la capture (mais ne sont bien envoyés qu'en un exemplaire sur le réseau). Il n'existe en effet pas de moyen de distinguer, au niveau du socket, un message provenant du bus d'un émis par votre machine.

Sécurité du protocole

Les expériences précédentes permettent de mettre en évidence que le protocole CAN ne fournit aucun mécanisme de sécurité. Étant un bus de communication, chaque équipement connecté recevra l'intégralité des échanges, où chaque type de message sera identifiable par la valeur de son champ « arbitration ID ». Il est de plus impossible de savoir qui a émis un message et à qui il est destiné. Cela implique que la mise en place d'une politique de filtrage, comme cela peut être fait sur des réseaux IT classiques, n'est pas réalisable.

Une contre-mesure, qui commence à être utilisée par les constructeurs, consiste à segmenter le réseau interne en plusieurs bus CAN distincts. Cela permet par exemple de cloisonner d'un côté les ECU ayant un rôle critique, et de l'autre ceux liés au confort des passagers. Dans ce cas, un équipement se chargera de faire l'interface entre ces multiples réseaux. Il sera alors un sujet d'étude de premier choix, car sa compromission briserait complètement la propriété de cloisonnement et permettrait à un attaquant d'injecter à volonté des messages dans n'importe lesquels de ces réseaux.

Un bon point cependant à ce fonctionnement en broadcast, est l'impossibilité pour un attaquant de réaliser des attaques en homme du milieu (MITM pour « Man In The Middle »). Chaque message émis sera forcément reçu par le destinataire légitime (sauf modification physique du bus bien évidemment). L'attaquant sera donc obligé soit de trouver un biais dans le protocole applicatif, lui permettant d'invalider un message légitime (appelé également MOTS pour « Man On The Side ») au bénéfice des siens, soit de saturer le bus de messages afin d'augmenter les chances de succès de son attaque.

Enfin, le protocole CAN ne propose aucun mécanisme de protection de la confidentialité des échanges, par du chiffrement, et contre le

rejeu, par un code d'authentification de message (MAC). C'est ce qui nous a permis précédemment de rejouer un message et d'en analyser le contenu. De ce fait, toute protection additionnelle ne pourra être mise en place que par les protocoles applicatifs qui seront implémentés.

CONCLUSION

Cet article avait pour vocation de faciliter l'interaction avec un bus CAN, dont l'accès est non trivial depuis nos équipements IT classiques, ainsi qu'à la mise en place d'un environnement de travail familier. La véritable étude de sécurité peut démarrer à partir de ce point. Le protocole CAN ne définissant que les couches basses protocolaires, les vulnérabilités ou les fonctionnements exotiques seront à rechercher dans les couches de transport, telle que l'ISO-TP, permettant d'envoyer des messages de plus de 8 octets, jusqu'au niveau applicatif. Les sujets d'étude sont vastes, allant du simple fuzzing à l'analyse des mécanismes de mise à jour des calculateurs. L'identification des différentes informations échangées par les ECU lors du fonctionnement du véhicule est également une des premières choses qui peut être réalisée si ces messages vous sont simplement accessibles.

Pour se lancer dans ce domaine, la lecture du très célèbre « Car Hacker's Handbook » **[HANDB]** est un incontournable. Il est également intéressant de consulter des sites spécialisés sur le sujet tel que **www.elm327.fr [OBD-II]**, ainsi que les analyses de sécurité, en commençant notamment par les publications des deux chercheurs Charlie Miller et Chris Valasek **[IOACTIVE]**.

Attention cependant à toujours garder à l'esprit que la plus grande prudence est de mise lors des expérimentations. De plus, chaque vulnérabilité découverte se devra d'être communiquée avec le plus de discrétion possible aux acteurs concernés, du fait notamment de la difficulté de mettre à jour de tels systèmes et de l'impact direct que leur exploitation pourrait avoir sur la sécurité d'un grand nombre de personnes. ▌

RÉFÉRENCES

[MISC97] A. Amokrane, « Pentest dans les réseaux CAN : des standards à la pratique », *MISC n°97*, mai-juin 2018

[PIO] Lien vers le projet de PlatformIO : **https://platformio.org/**

[CAN-UTILS] Code source du projet SocketCAN : **https://github.com/linux-can/can-utils/**

[SLCAND] Options de configuration du daemon slcand : **https://elinux.org/Bringing_CAN_interface_up/**

[SCAPY] Code source du projet Scapy : **https://github.com/secdev/scapy/**

[HANDB] Lien vers le Car Hacker's Handbook : **http://opengarages.org/handbook/**

[OBD-II] Page du site d'un constructeur de sondes de diagnostic : **https://www.elm327.fr/norme-obd/modes-obd/**

[IOACTIVE] Lien vers une des publications de Charlie Miller & Chris Valasek sur le sujet : **https://ioactive.com/pdfs/IOActive_Adventures_in_Automotive_Networks_and_Control_Units.pdf**

ACTUELLEMENT DISPONIBLE

GNU/LINUX MAGAZINE N°223

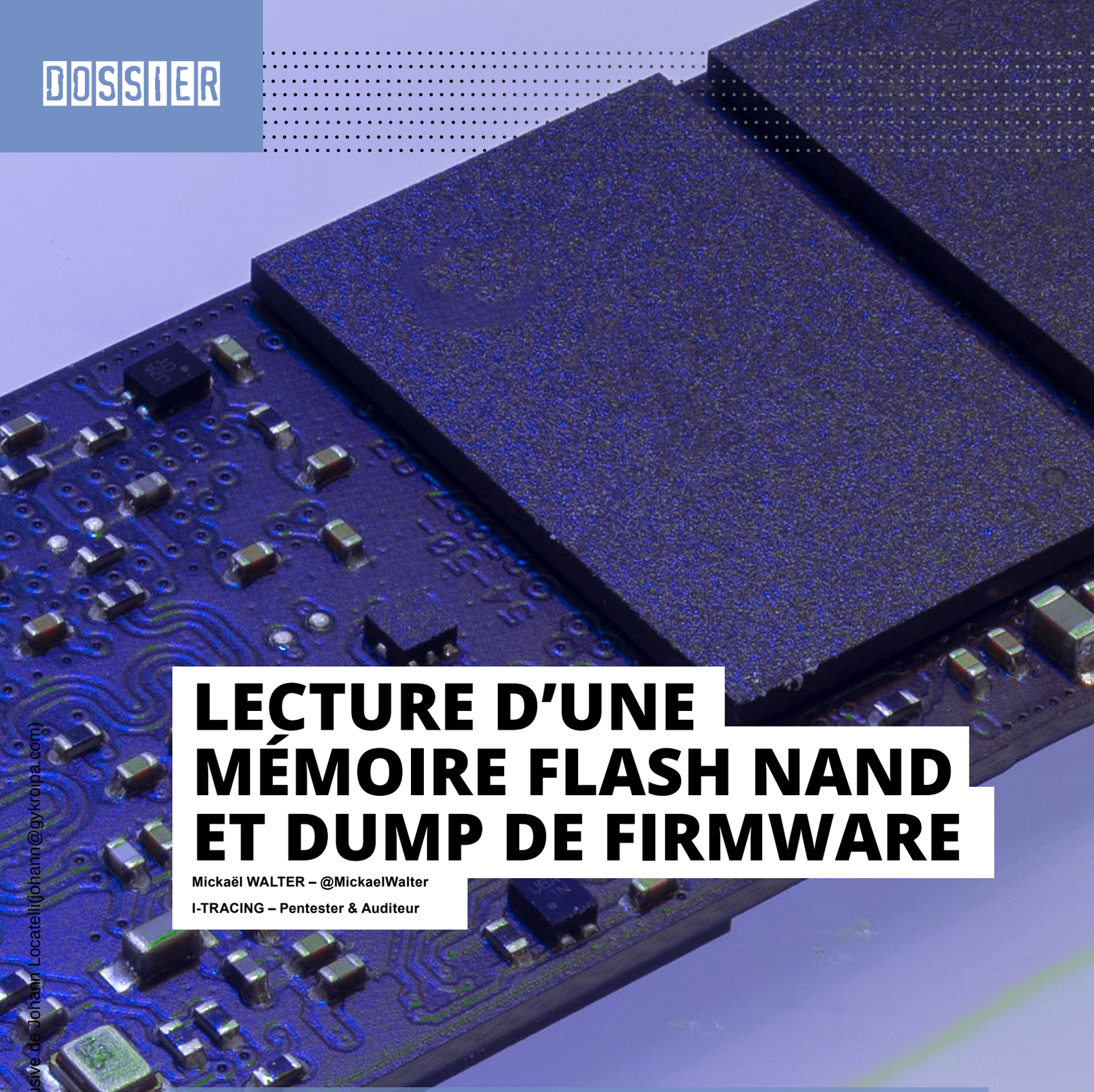
« COMPILER » SES SCRIPTS PYTHON

POUR WINDOWS, LINUX ET MACOS

NE LE MANQUEZ PAS

CHEZ VOTRE MARCHAND DE JOURNAUX ET SUR :

https://www.ed-diamond.com



LECTURE D'UNE MÉMOIRE FLASH NAND ET DUMP DE FIRMWARE

Mickaël WALTER – @MickaelWalter

I-TRACING – Pentester & Auditeur

Photo d'illustration.

Obtenir le firmware est une étape importante de l'analyse d'un système embarqué puisqu'il donne de précieux indices sur le fonctionnement de l'équipement. Il peut parfois s'avérer nécessaire de le récupérer directement sur la mémoire flash de l'équipement.

Lors de l'analyse d'un appareil embarqué, l'obtention du *firmware* facilite la recherche de vulnérabilités et, plus généralement, la compréhension de son fonctionnement. De nombreux fabricants fournissent eux-mêmes le *firmware*, parfois en ligne, à des fins de mises à jour. Pour d'autres, il peut être nécessaire de chercher un accès root afin de réaliser une copie « à chaud ». Cela peut se faire par l'exploitation d'une ou plusieurs vulnérabilités logicielles ou par un accès matériel au travers des ports de débogage souvent présents en hardware.

Pourtant, ces ports ne donnent pas toujours l'accès à une console root ou à la mémoire et l'état du processeur. Une approche complémentaire est alors d'attaquer le matériel « à froid » afin de lire directement le *firmware* à partir des mémoires persistantes.

Nous allons cibler dans cet article le cas particulier des mémoires flash NAND brutes, c'est-à-dire ne disposant pas d'un contrôleur intégré. L'absence de celui-ci, que l'on retrouve en revanche dans les mémoires eMMC ou SD, implique la gestion par le système d'exploitation de plusieurs fonctions très bas niveau.

L'attaque « à froid » d'un système embarqué par la lecture de sa mémoire flash est donc une approche permettant d'obtenir de manière exhaustive l'ensemble des données du *firmware*.

Les méthodes « à chaud » peuvent cependant être d'un plus grand intérêt, au moins dans un premier temps. En effet, la lecture directe d'une mémoire flash implique la pose de sondes ou le démontage de composants avec le risque d'endommager irrémédiablement le sujet d'étude. Il est donc préférable d'opter pour la lecture directe de la mémoire qu'après avoir épuisé les options précédentes.

ACQUISITION DES DONNÉES

Fonctionnement général d'une mémoire flash NAND

La robustesse des mémoires flash NAND et leur faible coût expliquent, entre autres avantages, leur présence répandue en informatique embarquée. Mais celles-ci sont affectées par des limitations qui doivent être prises en compte dans la conception de ce type de système :

- l'écriture use la mémoire et est le principal facteur de vieillissement du composant ;
- les erreurs de lecture sont très courantes et nécessitent une correction par le contrôleur ;
- certains blocs (unités d'effacement) sont inutilisables dès la sortie de l'usine.

Par ailleurs, une mémoire flash brute, qui ne remplit que les fonctions liées au stockage, présente un jeu d'entrées/sorties spécifique à ce type de composant **[ONFI]**.

Lorsqu'une mémoire flash brute n'est pas gérée par un contrôleur dédié, c'est au système d'exploitation d'en tenir compte et d'appliquer les mesures nécessaires à la bonne lecture et à la prolongation de la durabilité de la mémoire.

Commençons par détailler les unités élémentaires d'une mémoire flash NAND SLC (*Single-Level Cell*) :

- l'octet ;
- la page ;
- le bloc ;
- la *spare area* (aussi appelée *Out Of Bounds* ou OOB).

Une page est un ensemble d'octets et constitue l'unité élémentaire de lecture et de programmation (écriture). Elle est accompagnée d'un espace de stockage supplémentaire, la *spare area*, qui regroupe une série d'informations à propos des données utiles.

L'utilisation de cet espace est laissée libre au contrôleur de la mémoire et n'est pas formellement standardisée. Mais de manière pragmatique on y retrouve deux informations : le marqueur de bloc défectueux (*bad block* en anglais) et le résultat du calcul d'un code correcteur sur la page ou sur des sections de la page (les sous-pages). La manière dont le marquage de bloc défectueux en sortie d'usine est effectué est standardisée, à l'instar de l'interface d'entrée/sortie, par un consortium : l'ONFI (*Open NAND Flash Interface*) **[ONFI]**. En revanche, le code correcteur d'erreur ne l'est pas.

Le bloc est quant à lui l'unité élémentaire d'effacement et est constitué d'un groupe de pages. Pour qu'une page puisse être programmée, elle doit être vide (entièrement constituée de 1 binaire). L'opération d'effacement est donc nécessaire et va concerner tout un ensemble de pages à la fois **[PECYCLE]**.

Il existe également un objet de plus, le plan, dans le cas des mémoires MLC (*Multiple-Level Cell*), mais celles-ci ne seront pas traitées ici **[SLCVSMLC]**.

La figure 1 résume le découpage d'une mémoire flash NAND selon le bestiaire que nous avons décrit.

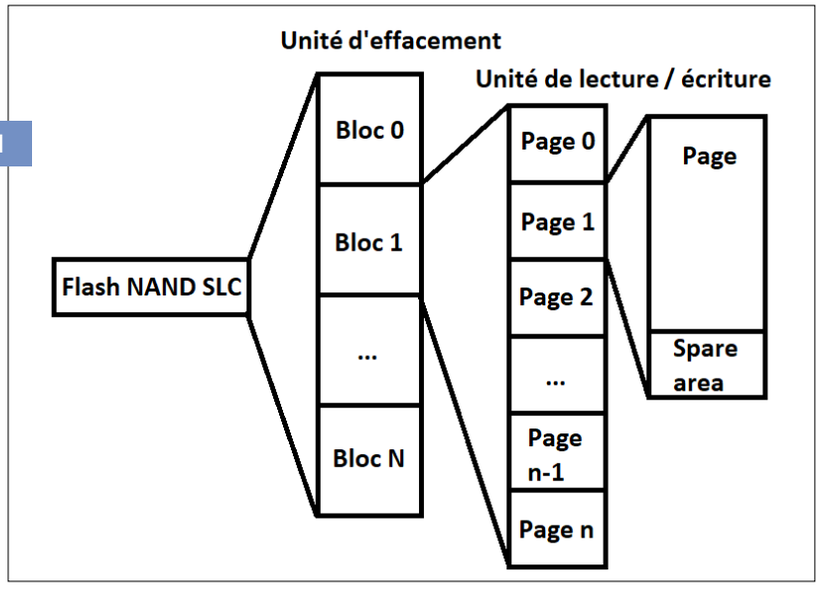


FIGURE 1

Organisation générale des éléments physiques d'une mémoire flash NAND.

Si la lecture d'une page n'affecte que très peu la mémoire, l'écriture a nettement plus d'incidence. Ainsi, en fonctionnement, de nouveaux blocs défectueux apparaissent.

Lecture

» Accès aux broches de la mémoire

La plupart des mémoires flash NAND suivent le standard ONFI. Il décrit, entre autres, le brochage pour différents boîtiers (TSOP-48, BGA-63, etc.) ainsi qu'une série de commandes minimales qui permettent la lecture, l'écriture ou encore l'obtention de métadonnées à propos de la mémoire. Ceci permet d'inter-

changer deux modèles différents de mémoire sur un circuit imprimé sans devoir modifier d'autres composants et facilite le travail d'identification des broches du composant.

Afin de lire la mémoire, deux solutions principales existent pour connecter les entrées/sorties au dispositif de lecture : dessouder le composant et accéder directement à ses broches ou poser des sondes pour s'y interfacer directement sur le circuit.

L'opération de réimplantation du composant peut être délicate, mais l'analyse sans démontage n'est pas non plus toujours possible (par exemple, dans le cas d'un boîtier BGA (*Ball Grid Array*) où les broches se trouvent sous le composant). De la même manière, si un fer à souder permet de détacher une mémoire encapsulée dans un boîtier TSOP-48, d'autres types de boîtiers nécessiteront des outils comme des stations à air chaud **[NANDDIS]**.

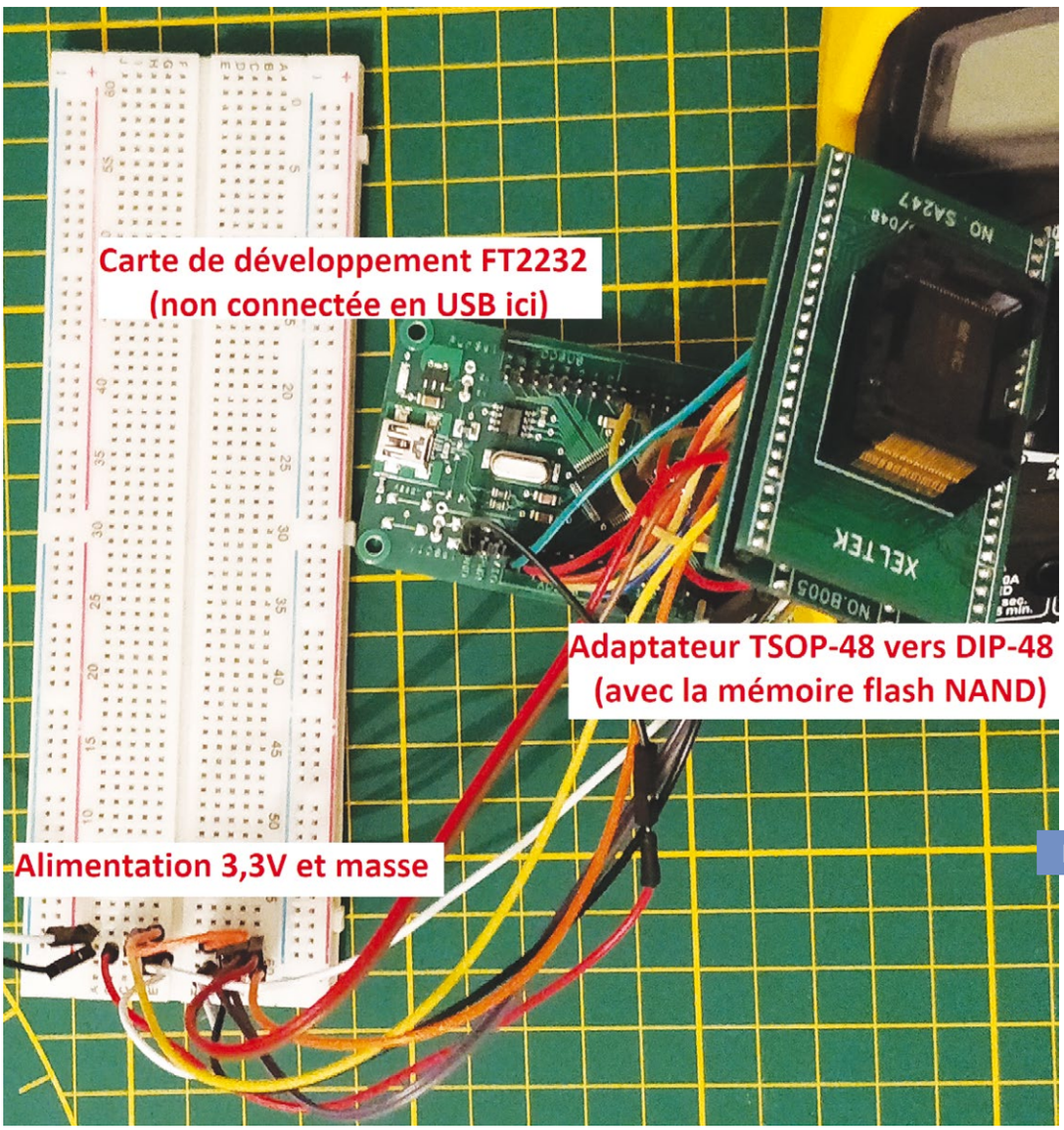


FIGURE 2

Montage de lecture d'une mémoire flash NAND à partir d'un circuit basé sur le FTDI FT2232H.

La méthode de lecture la plus simple repose sur l'utilisation d'un lecteur/programmateur de mémoire flash spécialisé. Mais ce type de matériel étant relativement onéreux, on peut opter pour un montage « artisanal ». Dans ce cas, il va falloir faire les choses à la main.

Heureusement, d'autres s'y sont intéressés avant nous et ont développé une technique de lecture en exploitant un microcontrôleur dédié à l'interfaçage entre l'USB et plusieurs protocoles d'échanges de données avec des composants électroniques : le FTDI FT2232H **[FTDIDS]**. Plusieurs cartes de développement le comprennent, comme par exemple celle de *Dangerous Prototypes* **[DPBOARD]**.

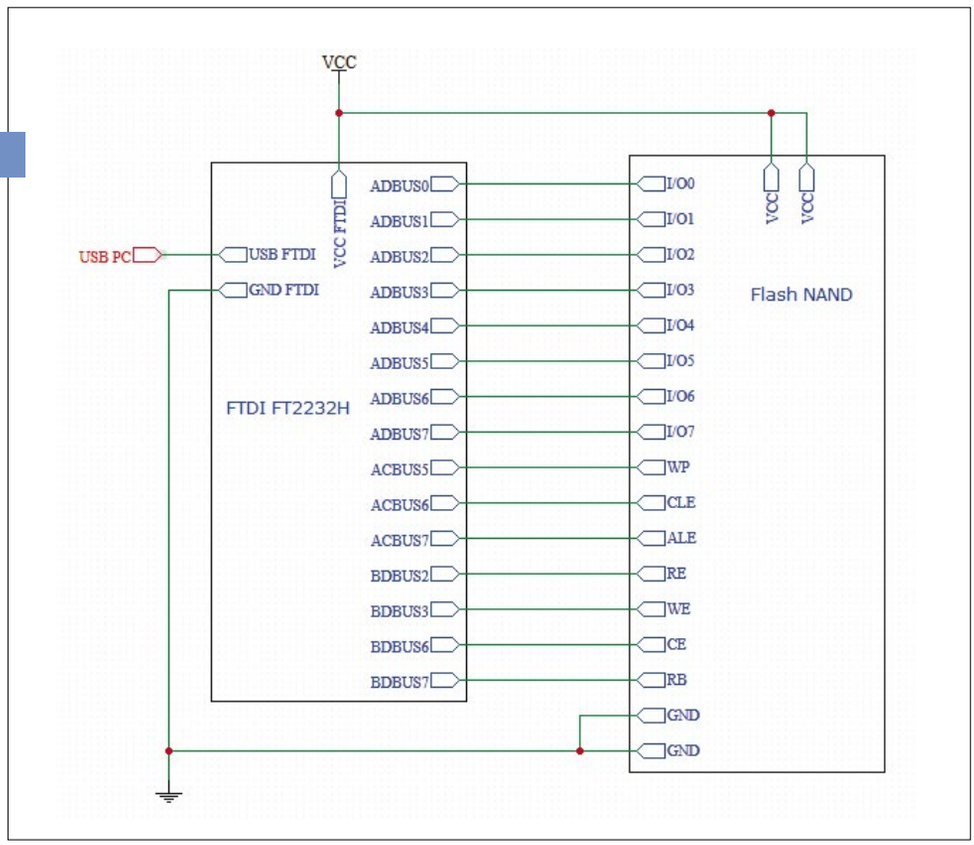
Les résultats de ces travaux se présentent sous la forme d'un programme de copie brute exploitant le FT2232H écrit en C++ **[SPRITESMODS]**, sobrement appelé *ftdinandreader*, ainsi qu'un script Python fonctionnant sur le même principe **[DUMPFLASH]**, *DumpFlash*. Ils sont disponibles sur Internet et peuvent être utilisés indépendamment.

La figure 2, page précédente, donne un aperçu d'ensemble d'un montage bricolé pour lire une mémoire flash NAND en TSOP-48. Il comprend : une alimentation en 3,3V (récupérée sur un lecteur UART), une carte de développement FT2232, un adaptateur TSOP-48 à DIP-48 (habituellement utilisé avec les programmateurs flash, mais qui peut être obtenu indépendamment), et une plaque de prototypage malheureusement incompatible avec l'adaptateur (d'où la présence des fils).

Dans les deux cas, les connexions entre la mémoire et le microcontrôleur sont les mêmes et sont décrites en figure 3.

FIGURE 3

Connexions entre le microcontrôleur FT2232H et la mémoire flash.



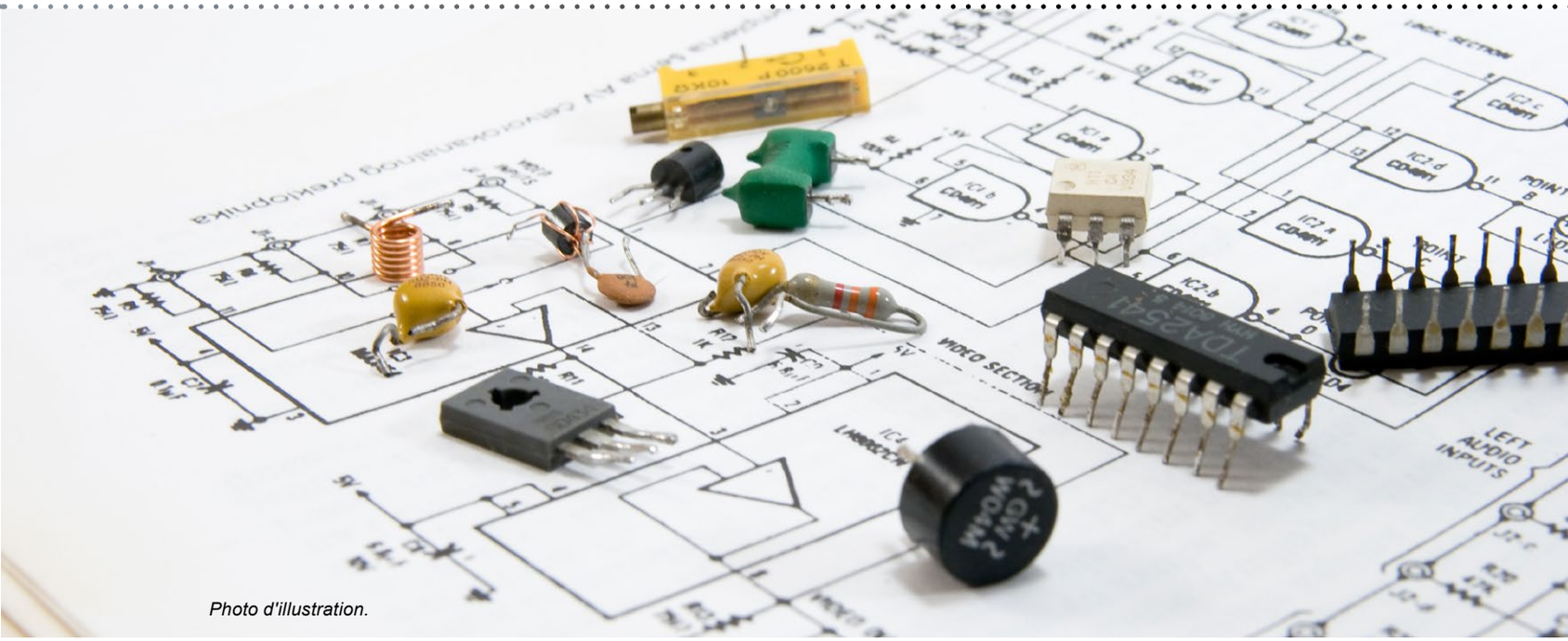


Photo d'illustration.

Le poste utilisé pour brancher en USB le FT2232H doit disposer de la bibliothèque **libftdi1** ainsi que de ses en-têtes de compilation pour le fonctionnement de **ftdinandreader** (**libusb-1.0** ainsi que **pyftdi** pour *DumpFlash*). Ces bibliothèques sont disponibles depuis les gestionnaires de paquets courants (**pyftdi** étant également disponible par **pip**) des distributions Debian et dérivées.

Il est en principe possible d'utiliser Windows pour assurer la connexion avec le FT2232H, les bibliothèques nécessaires étant disponibles pour ce système d'exploitation, même si votre serviteur a préféré GNU/Linux dans ses recherches.

Dans ces conditions, la vitesse de lecture à espérer est relativement faible et se compte en quelques dizaines de kilo-octets par seconde. C'est néanmoins largement suffisant pour des mémoires ayant des capacités de l'ordre du gigabit.

» En pratique

Une fois le programme de lecture installé, compilé et configuré, le *dump* est assez simple à réaliser. Il suffit de connecter la carte comprenant le FT2232H en USB et d'entrer la commande suivante pour vérifier dans un premier temps que tout est bien connecté et fonctionnel :

```
$ sudo ./ftdinandreader -i
FT2232H-based NAND readerNand type: NAND 128MiB 3,3V 8-bit
Manufacturer: Macronix
Size: 128MB, pagesize 2048 bytes, OOB size 64 bytes
Large page, needs 5 addr bytes.
All done.
```

Celle-ci exécute la commande de lecture des informations de base prévue par l'ONFI afin de récupérer les quatre octets de configuration de la mémoire flash SLC. Ces quatre octets sont ensuite interprétés pour afficher les informations de base de la mémoire. Si aucune erreur ne se produit à ce niveau, le matériel est fonctionnel et la lecture de la mémoire est possible.

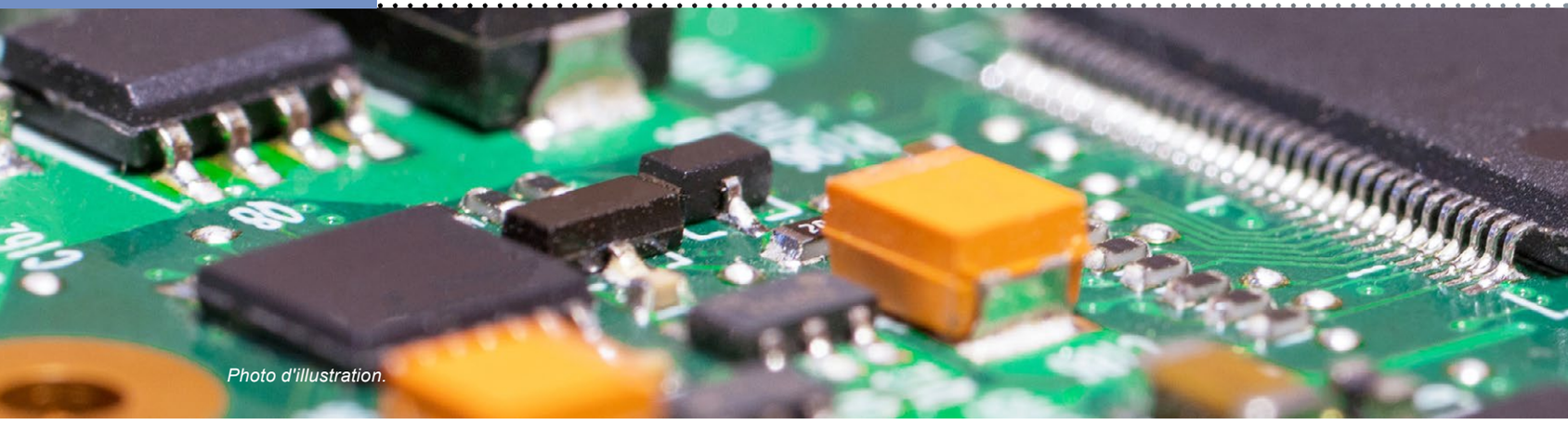


Photo d'illustration.

Il suffit alors de lancer la lecture avec la commande suivante et d'attendre que l'ensemble soit versé dans le fichier image résultant :

```
$ sudo ./ftdinandreader -r raw_nand.dmp -t both # Ici on lance la lecture
complète avec l'inclusion des spare areas
FT2232H-based NAND readerNand type: NAND 128MiB 3,3V 8-bit
Manufacturer: Macronix
Size: 128MB, pagesize 2048 bytes, OOB size 64 bytes
Large page, needs 5 addr bytes.
Reading 65536 pages of 2048 bytes...
All done.
```

L'image ainsi obtenue doit avoir en fin de *dump* une taille supérieure à celle annoncée par le fabricant. Elle doit en effet contenir les *spare areas* qui ne sont pas comptabilisées dans la capacité vendue.

En raison du fonctionnement par blocs et de la présence des *spare areas*, l'image n'est pas exploitable en l'état. De plus, nous avons jusqu'ici éludé la question de la correction des erreurs alors qu'il est certain que l'image contient des bits inversés présents de manière aléatoire (*bit flip*). Nous allons voir dans le prochain paragraphe comment corriger ces erreurs et ne récupérer que les données utiles.

Nettoyage de l'image

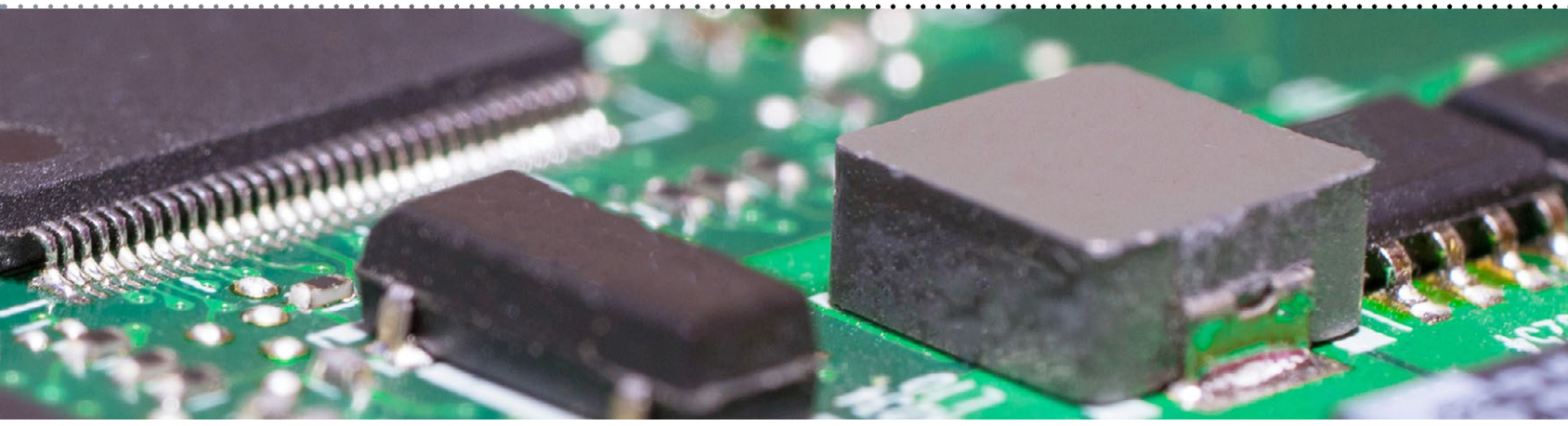
L'une des grosses limitations des mémoires flash NAND évoquées plus haut est l'apparition lors de la lecture d'erreurs sur quelques bits de chaque page. Le nombre d'erreurs et leurs positions sont variables et imprévisibles. C'est pourquoi il est nécessaire d'avoir la possibilité de les détecter et de les corriger.

Le nombre d'erreurs pouvant survenir est cependant borné et les fabricants donnent dans leur documentation la capacité de correction qu'il est recommandé d'avoir a minima pour chaque page.

La manière dont cette correction est faite n'est pas standardisée et de nombreux cas particuliers existent. Cependant, l'un des rôles prévus pour la *spare area* de chaque page est justement le stockage du résultat du code correcteur par le contrôleur.

Comme son nom l'indique, le rôle du code correcteur est de localiser les éventuelles erreurs dans les données reçues par le contrôleur. Il permet donc de détecter la position des bits qui seront à rectifier pour la bonne lecture des données.

Il existe plusieurs algorithmes permettant de corriger plus ou moins de bits, mais on peut citer la famille des codes *Bose-Chaudhuri-Hocquenghem* (BCH). C'est cet algorithme qui est utilisé par le noyau Linux lorsque



la charge de la correction incombe au logiciel **[KERNELBCH]**. Ce qui simplifie alors grandement le travail, dans ces cas, puisque nous pouvons alors nous reposer sur le noyau pour réaliser la correction en simulant une mémoire flash virtuelle.

Pour ce faire, on crée une mémoire NAND virtuelle similaire à celle examinée en utilisant les octets de configuration correspondants proposés par les développeurs du module noyau MTD **[MTDNANDSIM]**. Le package **mtd-utils** permet, sous des distributions Debian et dérivées, d'obtenir les outils de simulation et de manipulation de mémoire flash s'ils ne sont pas déjà disponibles de base.

Les commandes suivantes permettent d'activer et de configurer une mémoire flash virtuelle à l'aide du module noyau **nandsim** :

```
$ # Simulation d'une mémoire NAND de 128Mo avec des pages de 2048 octets
$ sudo modprobe nandsim first_id_byte=0xec second_id_byte=0xa1 third_id_
byte=0x00 fourth_id_byte=0x15
```

De cette manière, la mémoire flash virtuelle créée se trouve en RAM et est rendue disponible par un fichier périphérique : **mtd<x>** (avec **x** le numéro du device). Dans le cas d'une capacité conséquente, mieux vaut diriger les données vers un fichier. Cela peut se faire à l'aide de l'option **cache_file=<votre-fichier>**.

Il suffit ensuite d'y écrire les données de l'image en prenant soin de désactiver la correction d'erreurs et d'inclure les *spare areas* afin de s'assurer que c'est bien le résultat du code correcteur d'erreurs en place qui sera pris en compte par le module, et non pas un nouveau calcul sur les données erronées.

Les mémoires flash NAND brutes ne fonctionnant pas comme des périphériques de stockage habituels, il est nécessaire d'utiliser des commandes dédiées, car un simple **dd** échouera :

```
$ sudo flash_erase /dev/mtd0 0 0 # Effacement complet de la mémoire
virtuelle
Erasing 128 Kibyte @ 7fe0000 -- 100 % complete

$ sudo nandwrite -n -o /dev/mtd0 ./raw_nand.dmp # Ecriture de l'image
brute de la NAND dans la mémoire virtuelle sans code correcteur
d'erreur et en comprenant les spare areas
Writing data to block 0 at offset 0x0
Writing data to block 1 at offset 0x20000
[...]
```

Pour réaliser la correction et retirer les *spare areas*, il suffit de lire la mémoire, mais en laissant le module réaliser les corrections cette fois-ci :

```
$ sudo nanddump -f corrected_nand.dmp /dev/mtd0 # Dump de la
NAND virtuelle avec correction et retrait des spare areas
ECC failed: 0
ECC corrected: 0
Number of bad blocks: 0
Number of bbt blocks: 0
Block size 131072, page size 2048, OOB size 64
Dumping data starting at 0x00000000 and ending at 0x08000000...
```

Évidemment cette méthode ne fonctionne pas à tous les coups en raison de l'absence de standard sur la correction d'erreurs, mais devrait être applicable à une majorité de cas. Sinon il va falloir se renseigner au maximum, voire réaliser une ingénierie inverse, sur l'algorithme utilisé par le matériel ou le logiciel de contrôle.

ORGANISATION ET LECTURE DES DONNÉES

Agencement en mémoire flash d'un système GNU/Linux

L'agencement en mémoire flash d'un système embarqué est spécifique et n'a pas de raison particulière d'être linéaire.

Le code d'amorçage de second niveau étant lui-même appelé par celui de premier niveau intégré au SoC, il doit se trouver à un emplacement précis dans la mémoire. Il s'agit du seul emplacement réellement déterminable à l'avance. Il est situé généralement en début de mémoire, car les fabricants garantissent la plupart du temps que le premier bloc est valide.

On peut ensuite trouver mélangées les données utiles dans différentes parties. Il est possible d'y trouver des systèmes de fichiers dont *Unsorted Block Image File System* (UBIFS), qui est conscient des contraintes d'une mémoire flash NAND, ou encore le système *squashfs* qui a l'avantage de pouvoir stocker de manière compressée des données qui ont vocation à rester constantes (numéro de série, éléments d'identification de l'équipement, etc.). Sur d'autres mémoires, on peut également trouver JFFS (*Journaling Flash File System*) et JFFS2 qui sont les prédécesseurs de UBI.

À l'inverse, il n'est pas judicieux d'utiliser de partition de *swap* sur une mémoire flash NAND pour des raisons évidentes : l'écriture récurrente de données causerait une usure prématurée. La même idée tend à organiser le dossier temporaire (`/tmp`) dans une partition entièrement située en RAM.

Il est intéressant de se pencher sur la manière qu'a GNU/Linux d'organiser la fragmentation des partitions et leur imbrication sur la mémoire afin d'assurer le *wear-leveling*. Nous pourrons alors mieux comprendre comment réassembler et lire ces données pour enfin accéder aux systèmes de fichiers.

Photo d'illustration.

La stack UBI/UBIFS

» Le mécanisme de bloc UBI

Unsorted Block Images (UBI) est un mécanisme de gestion des blocs d'effacement/programmation dédié aux mémoires flash NAND non managées. Il a la charge, entre autres, de la gestion du *wear-leveling*, qui consiste à répartir l'écriture sur l'ensemble de la mémoire afin d'obtenir une usure uniforme des blocs. Il prend aussi en compte les blocs défectueux.

UBI est spécifique à cette application. Il n'est pas compatible avec d'autres types de mémoire comme les mémoires managées (eMMC, SD, etc.) ou les supports de stockage plus « classiques » comme les disques durs.

Le principe de base du fonctionnement d'UBI est de mettre en relation des blocs logiques (*Logical EraseBlock* – LEB) avec les blocs physiques (*Physical EraseBlock* – PEB). Cela permet de rendre transparentes pour les couches supérieures toutes les tâches liées à la gestion des blocs et en particulier leur agencement en mémoire. En effet, les LEB d'un même volume paraissent continus alors que les PEB sont désordonnés dans la mémoire.

UBI définit deux types de volumes : les volumes statiques et les volumes dynamiques. Les volumes statiques ont un mode de fonctionnement en lecture seule et sont particulièrement utiles pour les systèmes de fichiers *squashfs* évoqués au paragraphe précédent. Par ailleurs, les volumes dynamiques permettent l'accès en lecture/écriture.

Afin de fonctionner, UBI réserve deux espaces de 64 octets en début de chaque bloc physique non défectueux pour y placer deux en-têtes. Ces deux en-têtes sont identifiés EC (*Erase Counter*) et VID (*Volume Identifier*) **[UBIHDR]**.

Les deux en-têtes sont aisément repérables en raison de leurs *magic numbers* « UBI# » et « UBI! ». Ceci permet d'identifier très rapidement la présence de PEB UBI dans une image de mémoire flash NAND avec une simple recherche de chaîne de caractères.

Chacun des deux en-têtes remplit des rôles précis. L'en-tête EC porte le compteur d'effacement du bloc physique qui a pour but d'approximer son niveau d'usure et donc d'assurer la répartition d'usure, il est incrémenté à chaque effacement du PEB. L'en-tête VID quant à lui comporte l'essentiel des métadonnées pour un PEB, notamment son lien avec le LEB correspondant.

Toutes ces informations peuvent être obtenues à partir de l'image d'une mémoire flash NAND à l'aide de l'utilitaire *ubi_reader* **[UBIREADER]** en mode explicite.

Les outils *MTD Utils* comprennent des programmes utilitaires permettant d'activer le système UBI sur une mémoire flash directement connectée ou virtualisée avec *nandsim* **[UBI]**.

Pour monter une image de mémoire flash et accéder à ses volumes UBI, la série de commandes suivantes permet de charger les volumes dans **/dev** :

```
$ sudo modprobe ubi # Insertion du module noyau UBI
$ # Attachement du device virtuel UBI (l'option -O indique
l'offset du VID, à vérifier dans l'image, la valeur de 2048 est la
plus courante)
$ sudo ubiattach -O 2048 -p /dev/mtd0
UBI device number 0, total 1024 LEBs (130023424 bytes, 124.0 MiB),
available 99 LEBs (12570624 bytes, 12.0 MiB), LEB size 126976
bytes (124.0 KiB)
```

En cas d'erreur, la commande **dmesg** aidera à obtenir des informations à propos des problèmes rencontrés par l'utilitaire.

En cas de succès, la commande **ubiattach** renverra un message similaire à celui ci-dessus et créera des fichiers de périphérique sous **/dev** avec une nomenclature de la forme **ubi_<x>_<y>** (**x** est un nombre identifiant le système UBI attaché et **y** un numéro de volume). Il s'agit des différents volumes UBI.

Une fois les volumes identifiés et accessibles depuis **/dev**, les données utiles peuvent directement être obtenues par un **dd** classique :

```
$ # Récupération triviale des données à l'aide d'UBI et de dd
$ sudo dd if=/dev/ubi0_0 of=vol0_dump.dmp
61+1 records in
61+1 records out
31648 bytes (32 kB, 31 KiB) copied, 0.00706087 s, 4.5 MB/s
```

Le plus dur est désormais fait et il ne reste plus qu'à identifier et lire les systèmes de fichiers.

» Le système de fichiers UBIFS

Les systèmes de fichiers « classiques » (comme ext4) ne tiennent pas compte des particularités des mémoires flash NAND et peuvent poser problème.

UBI File System (UBIFS) a été développé avec des fonctionnalités répondant aux contraintes des systèmes embarqués et des mémoires flash NAND brutes. Il est important de ne pas confondre ces deux éléments de la stack UBI/UBIFS qui ont des rôles bien différents.

Ces fonctionnalités comprennent le support des coupures de courant brusques, la compression à la volée ou encore le contrôle d'intégrité **[UBIFS]**.

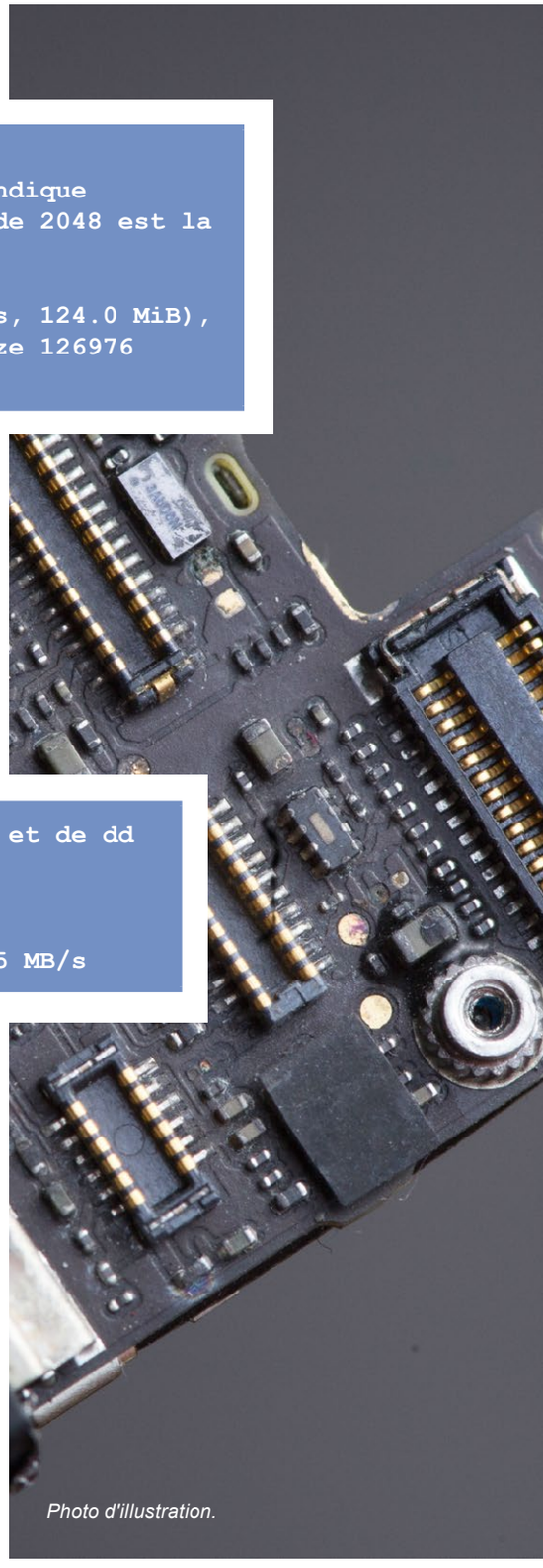
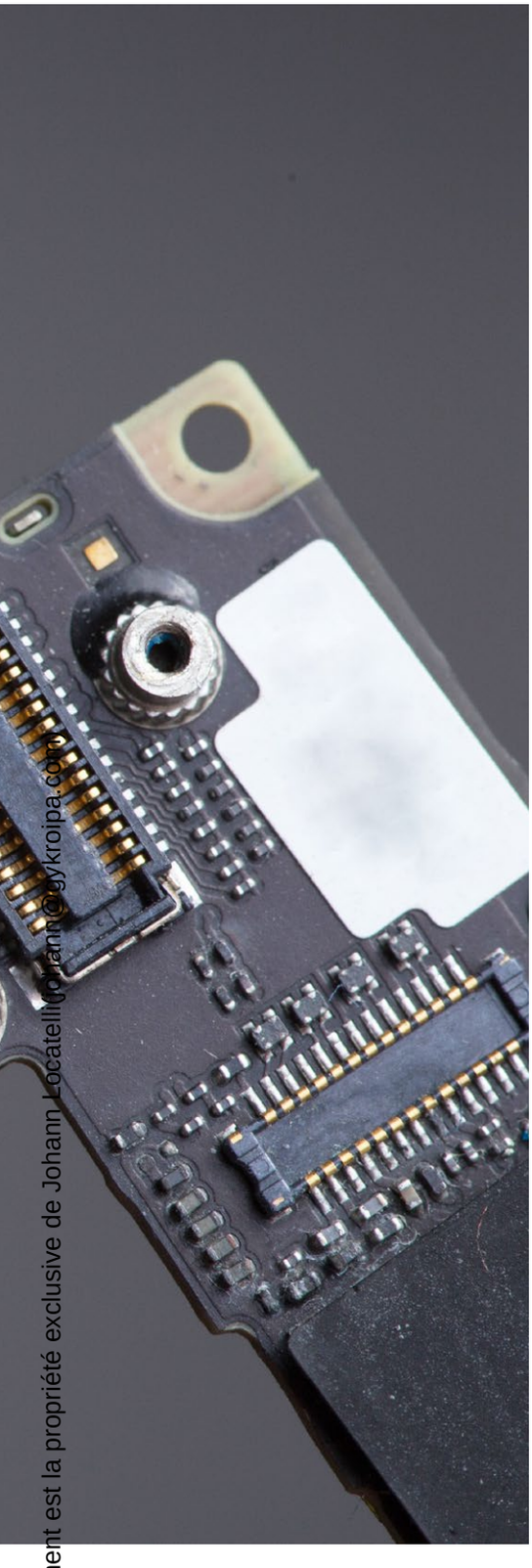


Photo d'illustration.



Nous n'allons pas détailler son fonctionnement et simplement monter le volume UBIFS pour enfin accéder aux fichiers du *firmware* :

```
$ # On peut déterminer à l'avance s'il s'agit bien
d'un volume UBIFS à l'aide de file
$ sudo file -Ls /dev/ubi0_6
/dev/ubi0_6: UBIfs image, sequence number 1,
length 4096, CRC 0xaf03c5e5

$ sudo mkdir /media/ubifs_vol # Le montage d'un
système UBIFS est trivial
$ sudo mount -t ubifs /dev/ubi0_6 /media/ubifs_vol
```

Le processus est ridiculement simple après toutes les autres étapes déjà passées.

Si toutefois des problèmes persistent, il est possible de les traquer avec `dmesg` et l'outil *ubi_reader* déjà évoqué plus haut.

CONCLUSION

Dans cet article, nous avons surtout évoqué les outils proposés par la communauté GNU/Linux pour prendre en charge ces mémoires aux contraintes très particulières. Cependant, les technologies de gestion sont assez disparates et il est probable qu'un travail de recherche supplémentaire soit nécessaire pour l'adapter à chaque cas particulier.

Le mécanisme de blocs UBI n'est pas toujours utilisé et nous pourrions également évoquer la gestion directe par Linux du *Memory Technology Device* (MTD).

Au final, les difficultés de la lecture d'une mémoire flash se situent à plusieurs niveaux : la connexion aux broches du matériel, la disponibilité du matériel de lecture, la correction des erreurs de lecture et enfin la compréhension de l'agencement des données.

Mais la plupart des mémoires flash ont un fonctionnement similaire et respectent l'ONFI. On y retrouve pages, blocs, *spare areas* et brochages identiques. Autant de constantes qui facilitent le travail de lecture.

On peut également envisager, à la suite de la lecture, l'écriture de mémoire flash et le *patching* du *firmware*, par exemple pour y placer une porte dérobée permettant l'analyse en fonctionnement de l'équipement. L'opération est nettement plus délicate, mais faisable **[NANDRW]**.

Dans tous les cas, l'analyse directe du fonctionnement de telles mémoires et des mécanismes de gestion est riche en enseignements et permet de mieux comprendre les liens entre matériel et logiciel.

REMERCIEMENTS

Un grand merci à Michaël SOUCHET et Mathieu FERRANDEZ pour leur relecture et leurs encouragements, aux relecteurs et contributeurs de *MISC* pour leur patience, et aussi à tous ceux référencés plus haut ou non pour avoir aidé un profane du hardware à décortiquer le sujet.

RÉFÉRENCES

[NANDDIS] Oh Wook Jeong, BlackHat US 14, Reverse Engineering Flash Memory for Fun and Benefit : **https://www.blackhat.com/docs/us-14/materials/us-14-Oh-Reverse-Engineering-Flash-Memory-For-Fun-And-Benefit-WP.pdf**

[PECYCLE] Understanding Flash: Blocks, Pages and Program / Erases : **https://flashdba.com/2014/06/20/understanding-flash-blocks-pages-and-program-erases/**

[SLCVSMLC] SLC vs. MLC: An Analysis of Flash Memory : **http://www.supertalent.com/datasheets/SLC_vs_MLC%20whitepaper.pdf**

[ONFI] Intel Corporation, Micron Technology Inc., Phison Electronics Corp., Western Digital Corporation, SK Hynix Inc. et Sony Corporation, « Open NAND Flash Interface Specification revision 4.1 », 2017

[SPRITESMODS] Blog de SpritesMods : **http://spritesmods.com/?art=ftdinand**

[DUMPFLASH] Dépôt GitHub d'Oh Wook Jeong : **https://github.com/ohjeongwook/DumpFlash/**

[DPBOARD] La description de la carte de développement FT2232 de Dangerous Prototypes : **http://dangerousprototypes.com/docs/FT2232_breakout_board**

[FTDIDS] Future Technology Devices International Limited : « FT2232H Dual High Speed USB to Multipurpose UART/FIFO IC Datasheet Version 2.5 », 2016

[KERNELBCH] Code source du code correcteur d'erreur du noyau Linux : **https://github.com/torvalds/linux/blob/master/drivers/mtd/nand/raw/nand_bch.c**

[MTDNANDSIM] Référence de nandsim : **http://linux-mtd.infradead.org/faq/nand.html**

[UBIHDR] Spécification des en-têtes UBI : **https://github.com/torvalds/linux/blob/master/drivers/mtd/ubi/ubi-media.h**

[UBIREADER] Dépôt GitHub de Jason Pruitt : **https://github.com/jrspruitt/ubi_reader/**

[UBI] Documentation de référence d'UBI : **http://linux-mtd.infradead.org/doc/ubi.html**

[UBIFS] Documentation de référence d'UBIFS : **http://www.linux-mtd.infradead.org/doc/ubifs.html**

[NANDRW] Fork du projet de SpritesMods pour intégrer l'écriture : **https://github.com/bkerler/NANDReader_FTDI**

ACTUELLEMENT DISPONIBLE

GNU/LINUX MAGAZINE HORS-SÉRIE N°100

NE LE MANQUEZ PAS

CHEZ VOTRE MARCHAND DE JOURNAUX ET SUR :

https://www.ed-diamond.com

ATTAQUES DES ÉQUIPEMENTS MOBILES : DU GPRS AU LTE

Sébastien DUDEK
(@FIUxIuS)

Les équipements de l'écosystème Internet of Things (IoT) utilisent diverses technologies telles que la radiocommunication mobile, perçue aujourd'hui comme un moyen éprouvé pour transporter de l'information partout dans le monde. Dans cet article, nous verrons en pratique comment attaquer ces équipements, ainsi que leurs infrastructures suivant différents scénarios en boîte noire.

Photo d'illustration.

INTRODUCTION

Contexte

La téléphonie mobile ou radiotéléphonie, fondée sur la radiocommunication, est apparue en France en 1986 avec le déploiement des premiers réseaux première génération (1G) France Télécom et des postes Radiocom 2000 utilisant ce réseau analogique. Cette technologie était une révolution, car elle lançait la téléphonie vers un nouvel air avec le principe de Réseau haute Densité (RHD, appelé en anglais *handover*) permettant de changer de cellule dynamiquement et de se déplacer. À l'époque, on pouvait déjà équiper sa voiture d'un téléphone Radiocom 2000 **[5]**. En décembre 1987, pour favoriser la croissance de la radio téléphonie, l'instance régulatrice des Postes de Télécommunications a accordé une autorisation d'exploitation pour d'autres opérateurs comme la Société française de radiotéléphonies (SFR) ayant développé par la suite sa propre « Ligne SFR » **[1]**.

Par fréquence porteuse, cette première génération occupait une très large bande (400 MHz), contrairement aux bandes de seconde génération utilisant en partie une technique de répartition en fréquence FDMA (*Frequency Division Multiple Access*) de 25 MHz en combinaison avec le TDMA (*Time-Division Multiple Access*) en Europe, ou même les réseaux 3G utilisant des techniques de multiplexage par code WCDMA (*Wideband Code Division Multiple Access*), ainsi que les réseaux 4G FDMA (*Frequency-Division Multiple Access*) et OFDM (*Orthogonal Frequency-Division Multiplexing*) de 5 à 20 MHz.

La première génération fut abandonnée au profit de la norme GSM le 28 juillet 2000 en Europe. À ce jour, il est possible de constater que l'introduction de la 3G et de la 4G n'a pas eu encore d'impact sur les réseaux 2G encore très largement utilisés. Mais on suppose qu'avec l'entrée de la 5G, en cette nouvelle année 2019, cela risque de changer. En effet, aux États-Unis, certains opérateurs ont déjà commencé à abandonner les réseaux 2G et 3G pour se consacrer à la 5G, qui n'est encore qu'une réalité commerciale **[6]**.

Les techniques de modulations, de codage et répartitions cellulaires ont permis d'améliorer les réseaux mobiles jusqu'à obtenir un réseau 4G fiable et rapide. Suivant les évolutions, de nouvelles offres et équipements basés sur la téléphonie mobile sont apparus :

- interphones ;
- tableaux de bord de voitures ;
- certains compteurs électriques « intelligents » ;
- boîtes aux lettres en libre service ;
- frigos connectés ;
- caméra et alarmes connectées ;
- etc.

En figure 1 (page suivante), nous pouvons observer une architecture de type M2M (*Machine-to-Machine*) utilisée pour centraliser les communications d'équipements comme les interphones.

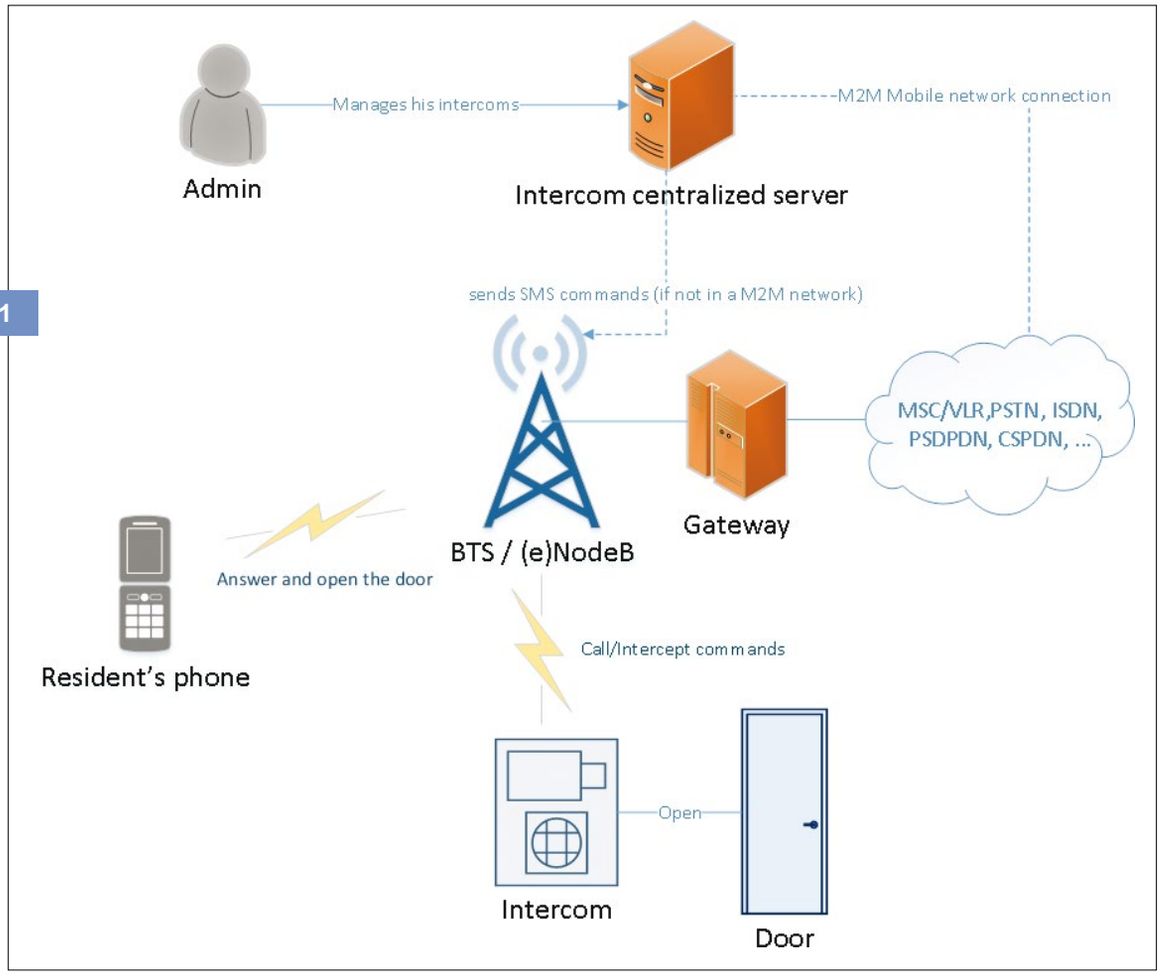


FIGURE 1

Architecture M2M pour des interphones connectés.

Sécurité des communications

Avec l'introduction de la radio logicielle et des premiers USRP de Ettus, des projets comme OpenBTS **[2]** de Range Networks et ceux d'osmocom ont rendu accessible l'univers, très fermé, de la télécommunication mobile. Ces projets ayant pour but principal d'ouvrir le monde de la radiotéléphonie, ont aussi été utilisés par des chercheurs en sécurité pour mettre en pratique des attaques comme l'interception et les attaques sur l'algorithme de chiffrement A5/1.

Même si l'introduction de l'algorithme vulnérable A5/2 hors d'Europe pendant une certaine période reste encore un mystère, les opérateurs ont su répondre en partie aux nouvelles menaces. Les normes GPRS, 3G et 4G ont été développées pour utiliser des algorithmes de chiffrement plus robustes tels que KASUMI (algorithme UEA-1), Snow-3G (UEA-2) deuxième algorithme possible pour UMTS et utilisé pour LTE (128-EEA1). LTE permet aussi d'utiliser du chiffrement AES 128 bits (128-EEA2) en plus de Snow-3G.

Contrairement aux réseaux GSM et GPRS, les réseaux UMTS et LTE forcent l'authentification mutuelle (voir figure 2) et introduisent le contrôle d'intégrité ainsi qu'une plus grande protection de

Photo d'illustration.

la signalisation. Cependant, la 3G reste toujours plus « permissive » avec l'utilisation de carte SIM (*Subscriber Identity Module*) simple, contrairement à la 4G. De plus, certaines exceptions quant à l'authentification mutuelle peuvent être observées sur des *firmwares* de modem/bandes de base (*baseband*) défaillants **[4] [28]**.

	GSM	3G	4G
Client authentication	YES	YES	YES
Network authentication	NO	Only if USIM is used (not SIM)	YES
Signaling integrity	NO	YES	YES
Encryption	A5/1	KASUMI | SNOW-3G	SNOW-3G | AES | ZUC...

FIGURE 2 *Sécurité des différentes technologies mobiles dans le cas d'une interception.*

À venir : la 5G

La technologie 5G très proche du LTE-A(*dvanced*), mais qui promet plus de rapidité théorique (10Gbps - 100 Gbps) et un spectre plus étendu (600 MHz - 6 GHz) pourrait avoir un rôle plus important dans l'univers IoT. En effet, l'argument mis en avant par la norme est la possibilité de communication point-à-point comme le V2X (*Vehicle-to-everything*) **[7]** destinée aux voitures autonomes, qui pourront se connecter à :

- une infrastructure (V2I) ;
- un réseau (V2N) ;
- un véhicule (V2V) ;
- et même à un piéton (V2P)...

À l'heure où cet article est écrit, la norme 5G est encore en phase de test et redéfinition/correction. Mais avec un regard critique, nous pouvons déjà nous questionner quant à la relation de confiance entre les différents systèmes, mais aussi l'impact sur la confidentialité des données.

MATÉRIEL

Radio logicielle

Les équipements de radio logicielle courants permettant de s'interfacer avec les équipements mobiles sont les suivants :

Équipement	Fréquences	Échantillonnage max. CAN/CNA (taux, largeur)	Logiciels supportés	Précision/stabilité d'horloge	Canaux TX/RX	Prix
USRP B2x0	70 Mhz - 6 GHz	61.44 Msps, 12 bits	■ 2G : OpenBTS ou OsmoBTS+OsmoTRX ■ 3G : OpenBTS-UMTS ■ 4G : srsLTE ■ 5G : OpenAirInterface	±2 ppm sans GPSDO	■ B200 : 1 Tx + 1 Rx ■ B210 : 2 Tx + 2 Rx	~800€ min.
bladeRF	300 MHz – 3.8 GHz	40 Msps, 12 bits	■ 2G : YateBTS ■ 4G : srsLTE ■ 5G : OpenAirInterface (attention : version 2.0 pas encore bien supportée)	±1 ppm	1 Tx + 1 Rx	~400€ min.
LimeSDR	100 kHz - 3.8 GHz	61.44 Msps, 12 bits	■ 2G : OpenBTS/ OsmoBTS avec OsmoTRX ■ 4G : srsLTE ■ 5G : OpenAirInterface	±2.5 ppm	2 Tx + 2 Rx	~300€ min.
XTRX	30 MHz - 3.7 GHz	120 Msps SISO / 90 Mss MIMO, 12 bits	■ 2G : OpenBTS avec OsmoTRX (encore expérimental)	± 0.5 ppm sans GPS / ± 0.01 ppm avec lock GPS	2 Tx + 2 Rx	~260€ min.

Le choix du matériel dépend généralement du budget et des supports de chaque périphérique. Pour un faible budget, les tendances vont vers le LimeSDR et le bladeRF, mais la gamme USRP B2x0 reste encore abordable pour un grand nombre d'usages. Malgré son faible prix, le nouveau XTRX, basé sur le *chipset* LMS7002M semblable au LimeSDR, est prometteur, mais manque encore de supports.

NOTE

La précision temporelle est une grandeur souvent exprimée en partie par million (ppm). Plus cette grandeur est faible, plus le signal a de chance de rentrer dans la fenêtre de réception. Cette grandeur est très importante dans la téléphonie mobile. Pour compenser le manque de précision d'un périphérique, une horloge de référence externe type GPSDO [9] peut être utilisée afin d'éviter de remplacer le quartz [8] de l'équipement radio logicielle.

Photo d'illustration.

Alternatives

D'autres équipements ou logiciels commerciaux peuvent aussi se révéler utiles :

- sysmoBTS **[10]** pour le GSM et GPRS ;
- femtocells 3G/LTE modifiées comme sysmoNITB **[11]** ;
- LTE LabKit de Yate pour la partie LTE **[12]** ;
- Amarisoft LTE **[24]** ;
- la version commerciale de srsLTE incluant Cat-NB1 pour IoT ;
- ou encore des équipements spécialisés comme les CMU200 **[13]**.

NOTE

Pour la 3G, OpenBTS-UMTS manque de fonctionnalités offertes par l'USIM (chiffrement et authentification). Si l'utilisation de ces fonctionnalités est nécessaire, il est intéressant de se tourner vers des solutions existantes telles que les femtocells vulnérables/modifiées [11][14] ou des équipements spécialisés.

ATTRACTION MOBILE

Explications

Généralement, les modems de ces équipements recherchent les cellules les plus favorables auxquelles se connecter. Pour fournir une haute disponibilité, ces modems implémentent plusieurs piles protocolaires.

Afin d'intercepter un équipement, 3 méthodes peuvent être envisagées :

- *downgrade attack* : repli vers 4G → 3G jusqu'à 2G ;
- isolation avec *shield* de Faraday ;
- utilisation d'une carte (U)SIM personnalisée.

Méthodes d'attractions

» Affaiblissement avec du « smart »-jamming

Ce type d'attaque est générique et ne requiert pas directement d'accès à l'équipement cible.

De façon naïve, nous pouvons utiliser un brouilleur (*jammer*) 2G/3G/4G commercial et retirer l'antenne 2G pour ne brouiller que les bandes 3G et 4G. Mais ce type de brouillage n'est que peu précis et peut laisser passer des réseaux UMTS (ex : bande B8) et LTE (ex : bande B3).

Pour pallier ce problème, l'attaque de *jamming* peut être améliorée avec un minimum d'intelligence (« smart »-jamming). Pour ce faire, 2 étapes sont nécessaires :

- monitorer et collecter les cellules avec leur indice (e/u)ARFCN courant (*Absolute Radio-Frequency Channel Number*) ;
- brouiller avec les fréquences associées aux (e/u)ARFCNs collectés des opérateurs ciblés.

De la même manière que le projet osmocomBB, un outil comme **modmobmap** **[16] [17]** peut être utilisé afin de récolter les informations nécessaires des cellules 2G/3G et 4G à l'aide d'un simple *smartphone* comme suit :

```
$ sudo python modmobmap.py -m servicemode -s <chemin Android SDK>
=> Requesting a list of MCC/MNC. Please wait, it may take a while...
Found 3 operator(s)
{u'20810': u'F SFR', u'20820': u'F-Bouygues Telecom', u'20801':
u'Orange F'}
[+] Unregistered from current PLMN
[+] New cell detected [CellID/PCI-DL_freq  (XXXXXXXXXX)]
 Network type=3G
 PLMN=208-10
 Band=8
 Downlink UARFCN=3011
 Uplink UARFCN=2786
[+] New cell detected [CellID/PCI-DL_freq  (3XX-1675)]
 Network type=4G
 PLMN=208-10
 Band=3
 Downlink EARFCN=1675
[...]
[+] Cells save as cells_1536076848.json # with an CTRL+C interrupt
```

Le fichier **cells_1536076848.json** conserve les informations des cellules que nous pouvons utiliser avec un outil de brouillage comme **modmobjam** **[18] [19]**. Après avoir adapté l'outil a son équipement radio logicielle, celui-ci s'utilise en 2 étapes avec la partie *jammer* à lancer en premier :

```
$ python GRC/jammer_gen.py
```

Puis doit être lancé en second le script **smartjam_rpcclient.py** permettant d'envoyer des commandes RPC au brouilleur en spécifiant la liste des cellules collectées précédemment :

```
$ python smartjam_rpcclient.py -f <Modmobmap path>/cells_1536076848.
json -w 208-1
```

De cette manière, nous pouvons brouiller les bandes 3G et 4G d'un opérateur cible pour attirer l'équipement vers notre fausse station GPRS.

NOTE

Des antennes de transmission supplémentaires seront sûrement nécessaires afin de brouiller plus de 5 fréquences quasi simultanément.

Faraday shield

Connue pour protéger les équipements du bruit électromagnétique, la cage de Faraday, du moins son concept, est aussi utilisée pour éviter les pollutions externes d'un équipement. Le terme « Faraday shield » est plus approprié dans le cas des ondes électromagnétiques comme le signal mobile. Pour atténuer le signal électromagnétique, sa conception dépend de plusieurs éléments importants :

- fréquence ;
- longueur d'onde ;
- puissance de réception et transmission ;
- l'épaisseur du *shield* ;
- distance entre le récepteur et le transmetteur.

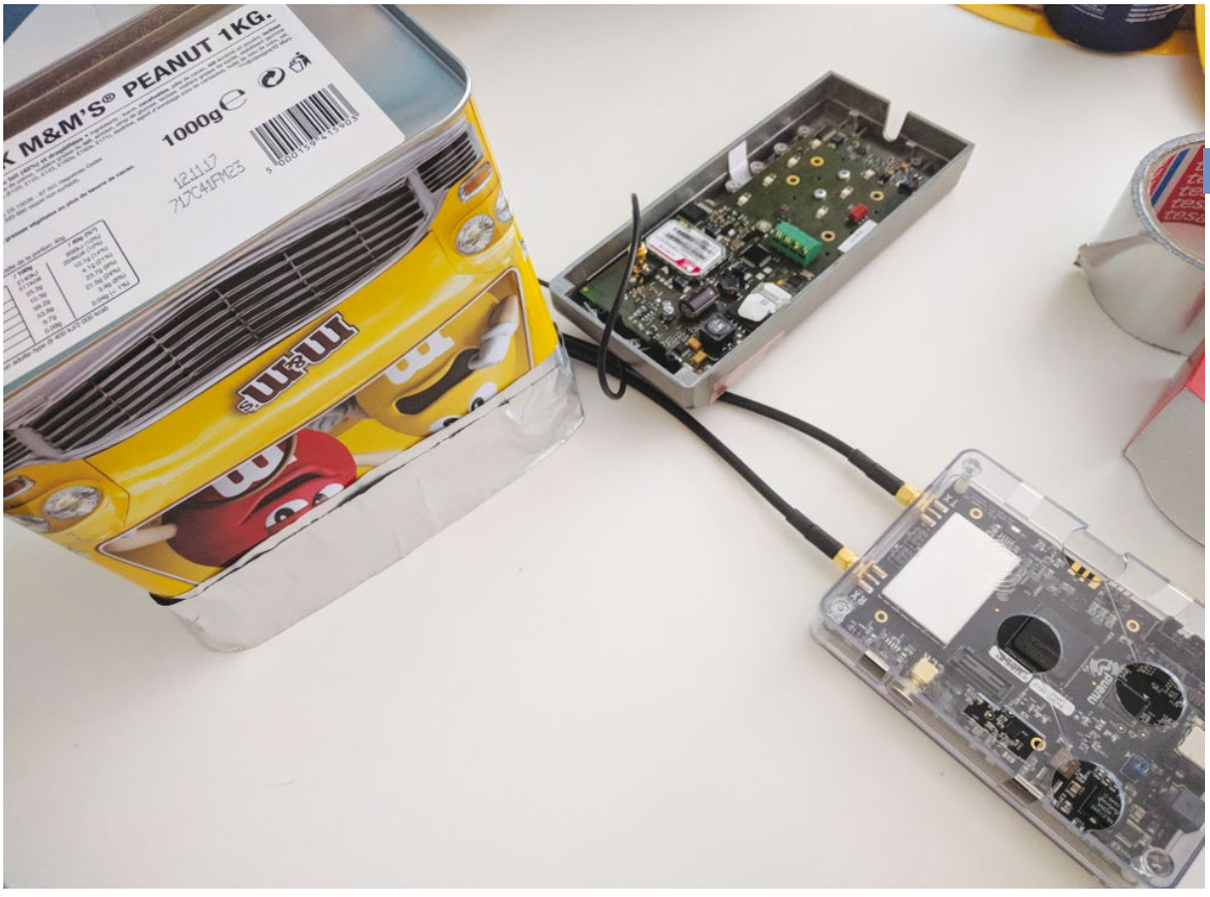


FIGURE 3

Utilisation d'une boîte de Faraday M&Ms pour de l'interception.

Malgré la faible accessibilité d'un *Faraday shield* performant, des alternatives plus abordables existent avec un peu d'imagination et bricolage. En effet, une simple boîte métallique peut être envisagée pour atténuer suffisamment le signal, si elle est suffisamment épaisse et éloignée des antennes d'opérateurs. En effet, nous pouvons voir, en figure 3 (page précédente), qu'une simple boîte de *M&Ms* et de scotch métallisé pour boucher les ouvertures permet d'isoler l'interphone et de l'attirer rapidement à la fausse station GPRS. De plus, pour ne pas prendre trop de place, seules les antennes des deux périphériques ont été mises dans la boîte.

NOTE

Notons que pour des équipements de type véhicule connectés, les parkings peuvent aussi se révéler très utiles pour des attaquants [20].

» Carte (U)SIM personnalisée

Lorsque cela est possible, l'utilisation de carte (U)SIM personnalisée peut être un gain de temps considérable lors d'attaques mobiles. En 2G, l'utilisation des cartes (U)SIM étrangères interdisant le *roaming*, ou alors avec un *prefered PLMN* reprogrammé, peuvent aussi être de bonnes alternatives. Rappelons aussi qu'en 3G, l'authentification et le chiffrement sont « désactivés » sans mode USIM.

La 4G quant à elle force l'utilisation des cartes USIM. Dans ce cas, des cartes telles que les sysmoUSIM peuvent être entièrement configurées, mais leur coût est relativement élevé. Ces cartes peuvent être configurées avec les outils PySIM et sysmo-usim-utils pour y modifier les secrets Ki (*subscriber key*) et OP/c (*Operator Variant Algorithm Configuration field*), ainsi que le MCC/MNC local comme suit :

```
$ sudo python pySim-prog.py -p0 -t sysmoUSIM-SJS1 -a 50024782 -x 001 -y
01 -i 9017000000***** -s 898821100000002******
[...]
 > Ki       : 6abb9ae663f9889eddaae298cdcb4ec6
 > OPC      : 074a3a73ed3c54e1960e9e5732ff35b1
 > ACC      : None
```

NOTE

Le coût de l'achat de carte USIM personnalisable peut être réduit, comme certains vendeurs chinois mettent à disposition des cartes que l'on peut trouver su Taobao en cherchant les mots clés « openlte sim » [21] [22].

Il est nécessaire de vérifier qu'un code PIN n'est pas rentré par le périphérique. Pour ce faire, l'utilisation d'outil comme le SIMtrace (voir figure 4) pourrait être utile afin de récupérer le PIN et éviter que le périphérique ne bloque notre carte personnalisée, mais aussi pouvoir réutiliser la carte M2M fournie pour attaquer les infrastructures.

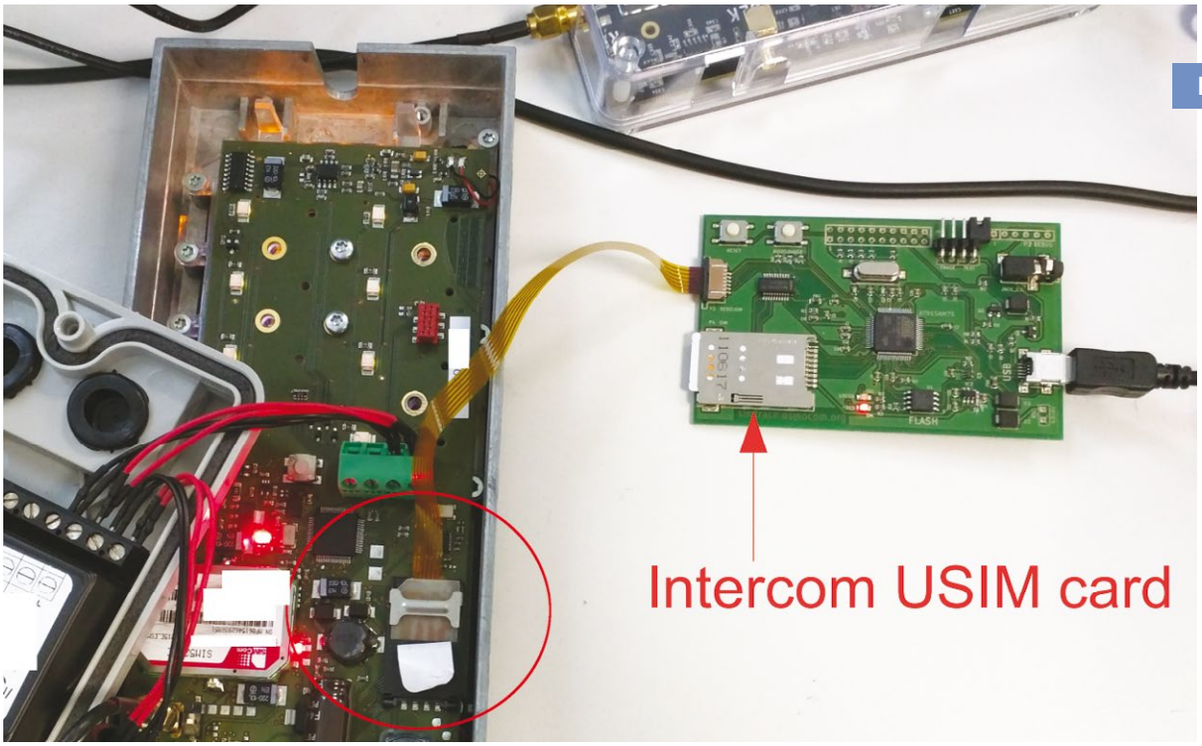


FIGURE 4

Utilisation du SIMtrace pour capturer le code PIN de la SIM.

INTERCEPTIONS

GPRS

En démarrant une BTS (*Base Transceiver Station*) avec l'outil YateBTS, par exemple, avec les options GPRS **[23]**, la cible peut être piégée avec les méthodes d'attraction citées plus haut. Une fois le périphérique piégé, nous pouvons observer son statut au sein du SGSN (*Serving GPRS Support Node*) comme suit :

```
mbts sgsn list
 GMM Context: imsi=20801XXXXXXXXXXXX ptmsi=0xd3001 tlli=0xc00d3001
state=GmmRegisteredNormal age=5 idle=1 MS#1,TLLI=c00d3001,8d402e2e
IPs=192.168.99.1
```

Nous pouvons observer qu'une IP a été assignée au périphérique (**192.168.99.1**) nous permettant de contacter le périphérique et observer les échanges avec ses serveurs comme illustrés en figure 5.

	Source	Destination	Protocol	Length	Time	Info
1	192.168.99.1	8.8.8.8	DNS	64	0.000000000	Standard query 0x11d8 A gsm. .info
2	8.8.8.8	192.168.99.1	DNS	80	0.037753523	Standard query response 0x11d8 A gsm. .info A 91.
3	192.168.99.1	91.121	TCP	48	0.419114786	80 → 60001 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 WS=1
4	91.121.	192.168.99.1	TCP	48	0.425593982	60001 → 80 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=146
5	192.168.99.1	91.121.	TCP	40	0.855774038	80 → 60001 [ACK] Seq=1 Ack=1 Win=16384 Len=0
6	192.168.99.1	91.121.	TCP	117	1.120101836	80 → 60001 [PSH, ACK] Seq=1 Ack=1 Win=16384 Len=77
7	91.121.	192.168.99.1	TCP	40	1.126491129	60001 → 80 [ACK] Seq=1 Ack=78 Win=29312 Len=0
8	91.121.	192.168.99.1	TCP	60	1.129285601	60001 → 80 [PSH, ACK] Seq=1 Ack=78 Win=29312 Len=20
9	91.121.	192.168.99.1	TCP	40	1.129573587	60001 → 80 [FIN, ACK] Seq=21 Ack=78 Win=29312 Len=0
10	192.168.99.1	91.121.	TCP	40	1.637377595	80 → 60001 [ACK] Seq=78 Ack=21 Win=16364 Len=0
11	192.168.99.1	91.121.	TCP	40	1.698825585	80 → 60001 [ACK] Seq=78 Ack=22 Win=16384 Len=0
12	192.168.99.1	91.121.	TCP	40	1.722705944	80 → 60001 [FIN, ACK] Seq=78 Ack=22 Win=16384 Len=0
13	91.121.	192.168.99.1	TCP	40	1.728877051	60001 → 80 [ACK] Seq=22 Ack=79 Win=29312 Len=0

FIGURE 5

Capture d'une communication entre l'interphone et le serveur au niveau SGSN.

Dans le cas d'un interphone connecté répandu dans Paris, certaines des requêtes peuvent être facilement rejouées, comme celle responsable de la synchronisation horaire :

```
[...]
In [7]: s.connect((ip, port))
In [8]: s.send(binascii.hexlify("[...]11e1540XX[...]"))
In [9]: data = s.recv(1024)
Out[11]: '2018/09/07 15:09:01\n'
```

Mais certains services peuvent chiffrer les données avec une clé, comme le service du port TCP 5001 dans le cas de notre interphone. Nous verrons plus tard l'importance des attaques *hardwares* dans ce type de cas.

UMTS et LTE

» 3G : limites et solutions

Certains modems ne supportent qu'un nombre limité de piles protocolaires (3G ou 4G seulement).

Pour cela, d'autres outils comme OpenBTS-UMTS pour la 3G et srsLTE existent, ainsi que des solutions commerciales plus spécialisées, comme mentionné plus haut.

Cependant, OpenBTS-UMTS ne supporte pas (encore) les fonctionnalités de chiffrement et d'authentification du mode USIM. L'utilisation de simples SIM est donc requise, mais certaines cartes programmables peuvent désactiver les fonctionnalités USIM comme les cartes sysmoUSIM comme suit :

```
$ sudo python sysmo-usim-tool.sjs1.py -a 772***** -c
[...]
   ==> USIM application disabled
```

Alternativement, il faudra se tourner vers des solutions moins accessibles comme le CMU200, ou alors une femtocell vulnérable/modifiée.

» Interception LTE

La 4G force l'utilisation de cartes USIM. La connaissance des secrets (OP/c et Ki) est nécessaire pour que le réseau authentifie l'équipement et génère le quadruplé suivant :

- RAND : challenge généré par le HSS (*Home Subscriber Server*) dans le HLR/AuC permettant de générer les prochains vecteurs d'authentification ;
- XRES : résultat demandé pendant le *challenge/response* avec le terminal ;
- AUTN : jeton d'authentification du réseau auprès de l'UE ;
- KASME : clé de dérivation des clés de chiffrement et d'intégrité.

Avant, il est nécessaire de vérifier que le mode USIM est bien activé :

```
$ sudo python sysmo-usim-tool.sjs1.py -a 50024782 -m
[...]
   ==> USIM application enabled
```

Photo d'illustration.

Les secrets connus peuvent ensuite être renseignés dans la base de données du cœur de réseau LTE EPC (*Evolved Packet Core*) comme suit avec srsLTE :

```
# vi /root/.srs/user_db.csv
[...]
ue3,9017000000*****,b5997ac4a912e9c6216e13951029c674,opc,83e5d3f22da411
072508f675d2e9e9d9,9001,000000000062,7
```

Si toutes les conditions sont bien remplies, l'EPC devrait indiquer que l'authentification s'est bien déroulée et une IP devrait être assignée comme suit :

```
[...]
UE Authentication Accepted.
[...]
SPGW Allocated IP 172.16.0.2 to ISMI 9017000000*****
```

Il ne reste plus qu'à intercepter les communications de la même manière qu'avec le GPRS.

ATTENTION !

Les équipements IoT LTE utilisent généralement des modems compatibles Cat M1 et NB-IoT, implémentés uniquement par les solutions commerciales de srsLTE et Amarisoft. Alternativement, il est possible de se tourner vers des piles protocolaires de repli (UMTS, voir GPRS) si possible.

Aller plus loin avec la 5G

Pour les intéressés, des outils comme OpenAirInterface5G **[25]** développé par Eurecom, existent. Bien que la partie EPC soit sujette à une licence, des EPC de sources ouvertes tels que NextEPC **[26]** et pycrate_core **[37]** peuvent servir de base et être réadaptés.

Difficultés rencontrées en audit

Beaucoup de bonnes surprises peuvent être trouvées avec l'interception d'équipement IoT (données en clair, etc.), mais des mécanismes de sécurité peuvent aussi perturber et ralentir les tests :

- *whitelist* des connexions ;
- utilisation de certificat client.

Généralement, la carte (U)SIM fournie avec le périphérique cible peut être utilisée sur un modem, ou téléphone portable, afin d'abuser de sa connexion internet et ainsi contacter les serveurs distants. Dans d'autres cas, la carte (U)SIM est embarquée et difficile d'accès. Mais cela n'empêche pas les attaques côté client afin de trouver des vulnérabilités sur l'équipement. En effet, la figure 6 illustre la capture d'un client web de tableau de bord automobile avec une version d'Android obsolète, que l'on pourrait attaquer avec des réponses spécialement forgées.

FIGURE 6

Capture d'une requête envoyée par le client web d'un tableau de bord automobile.

Cependant, si un certificat client est utilisé pour se connecter à l'infrastructure et que ce certificat est correctement vérifié, l'attaque de ces équipements et de ses infrastructures peut devenir complexe. Des attaques *hardwares* devront alors être entreprises afin de récupérer les secrets nous permettant de poursuivre les tests.

NOTE

Les communications (SPI, I²C, etc.) entre le modem et la (U)SIM embarquée peuvent être observées et abusées avec une modification matérielle pour tenter des attaques de type *relay* [27].

ATTAQUES HARDWARES

Cas d'un interphone connecté

Le premier réflexe à avoir est l'identification des interfaces. En figure 7, nous pouvons identifier différents composants d'un interphone connecté :

- *slot* (U)SIM (jaune) ;
- modem 3G (bleu) ;

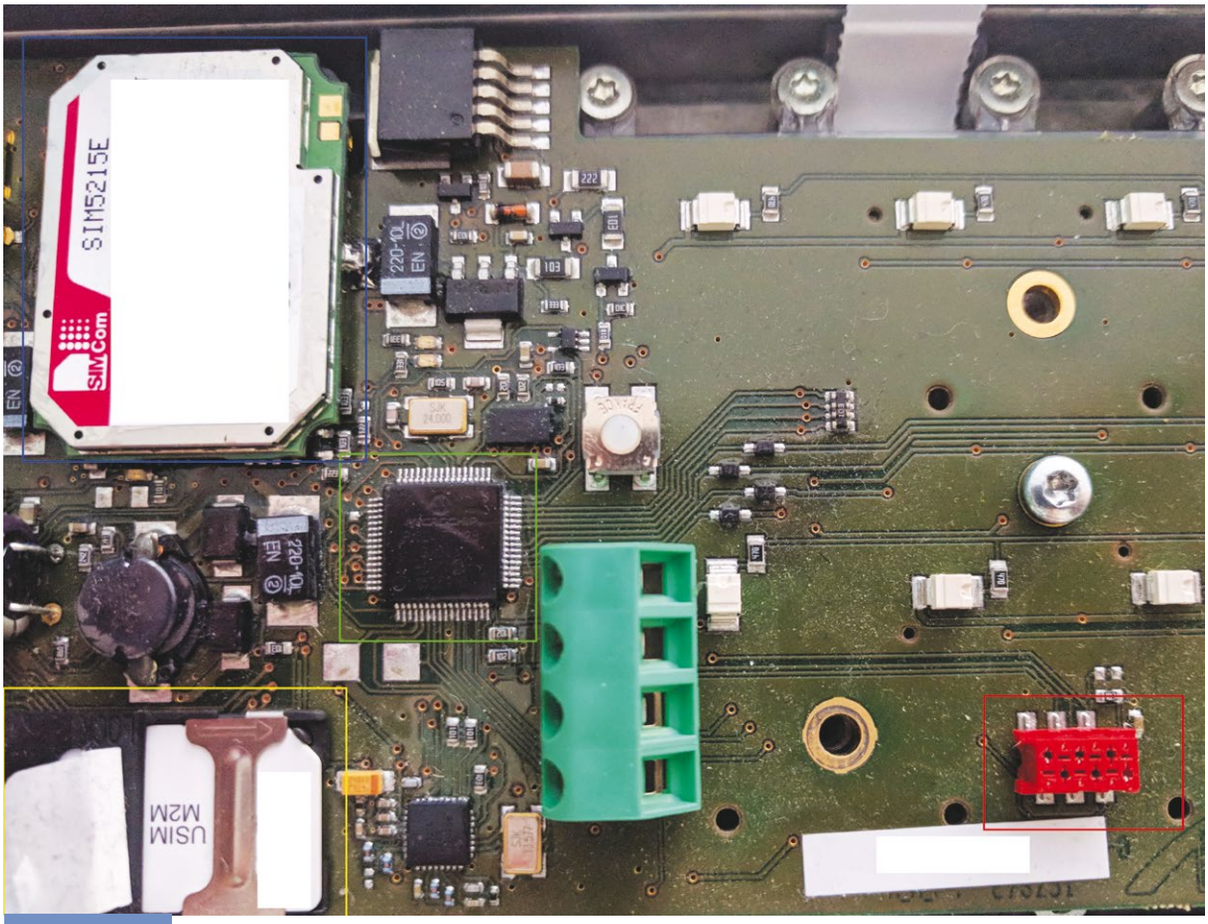


FIGURE 7 *Composants d'un interphone connecté.*

- microcontrôleur (vert) ;
- interface inconnue (rouge).

Afin d'en connaître plus sur cette interface, il faut se référer à la documentation constructeur du microcontrôleur : Microchip - PIC24FJ128 – GA006 (voir figure 8, page suivante).

Des tests de continuité permettent de rapidement identifier des ports relatifs au ICSP (*In-Circuit Serial Programmer*) et d'y connecter un programmeur dédié PICkit 3 (voir figure 9, page 69).

L'utilisation du programmeur avec le logiciel MPLAB-X permet de retrouver l'image du *firmware* de l'interphone, ainsi que son code désassemblé PIC24F (voir figure 10, page suivante), à étudier pour en extraire les secrets manquants.

Autres interfaces rencontrées

Sur des équipements embarqués, bon nombre d'interfaces peuvent être rencontrées :

- UART (*Universal Asynchronous Receiver/Transmitter*) : s'interfacer sur un *bootloader* (ex : uBoot), mais aussi la console d'un équipement si l'accès série est configuré par le système ;

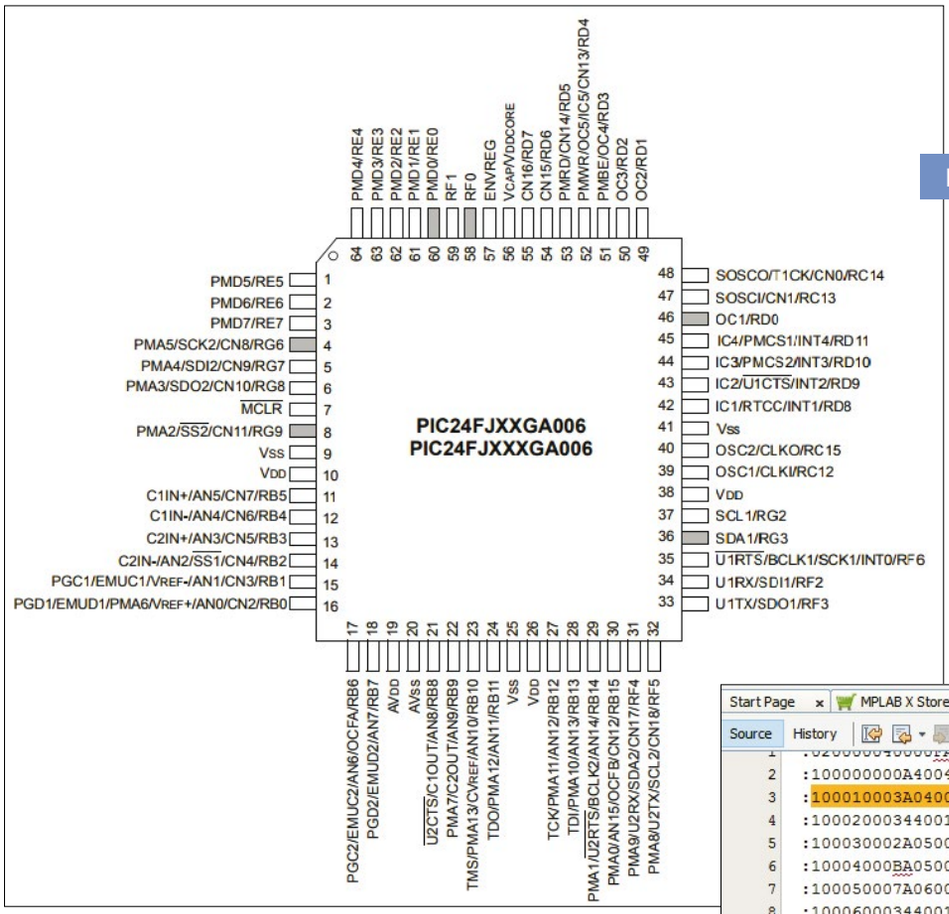


FIGURE 8

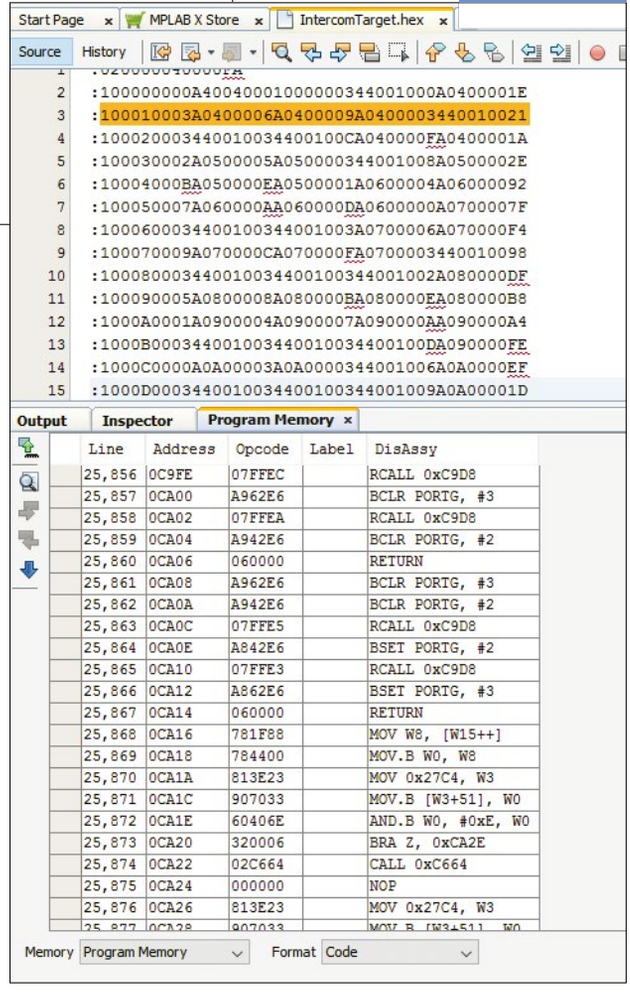
Schéma des différentes pattes du microcontrôleur.

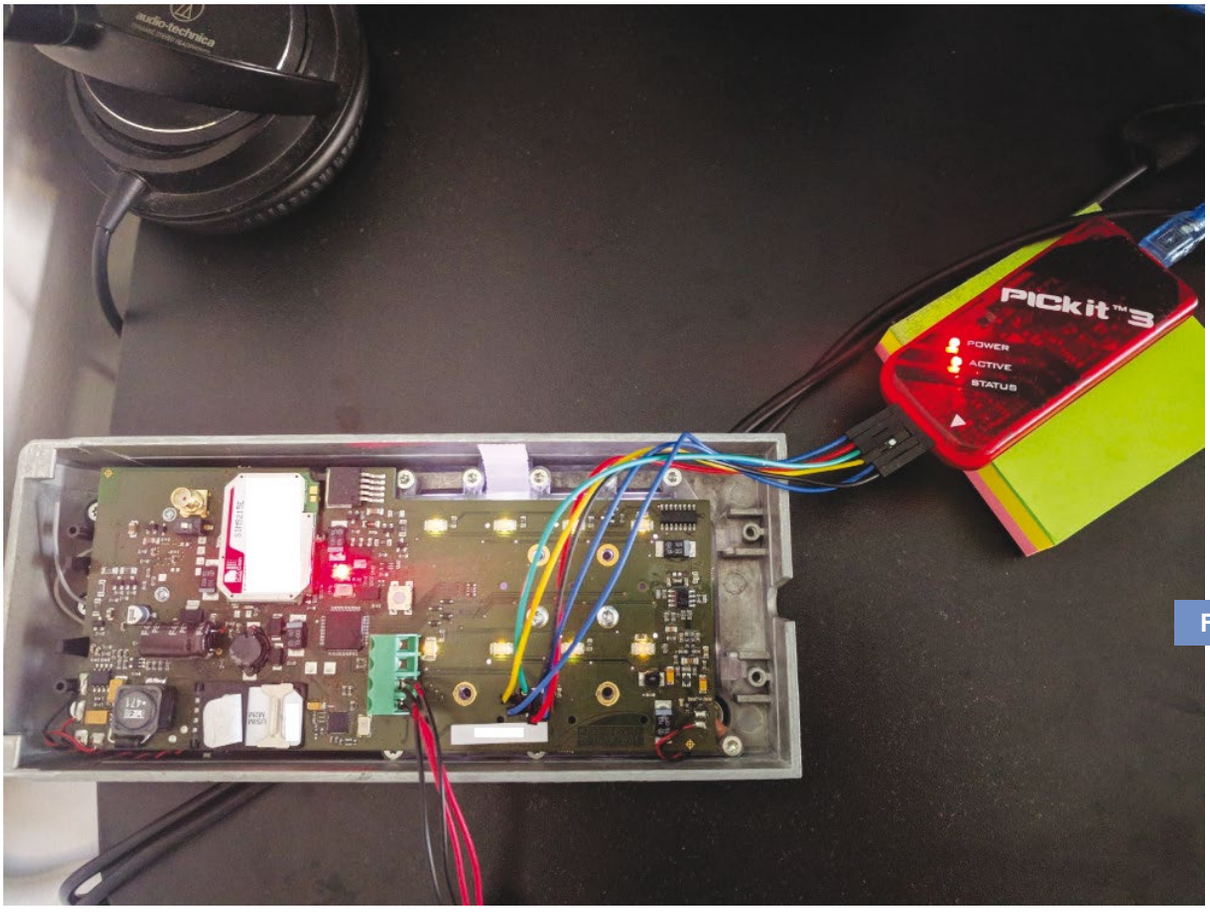
- JTAG (*Joint Test Action Group*) : communiquer avec les différents composants (ex. : mémoire flash) de la carte et tester les différentes communications **[33]** ;
- SPI (*Serial Peripheral Interface*): communication microcontrôleurs <-> autres périphériques (cartes mémoires, sondes, etc.) ;
- I²C : reliant les microcontrôleurs, EEPROMs et autres modules/périphériques ;
- et bien d'autres...

Ces interfaces peuvent être identifiées grâce aux documentations, tests de continuité et observation des pistes. Mais dans le cas d'une interface JTAG, l'identification des différentes pattes peut être une tâche fastidieuse. Pour cela, il existe des outils comme le JTAGulator **[34]** (~300€) pour identifier les différents *pins* (voir figure 11, page 70).

Extraction et désassemblage du firmware de l'interphone.

FIGURE 10





Utilisation du PICkit3 pour les microcontrôleurs PIC.

FIGURE 9

Des alternatives comme le BUSSide **[36]**, utilisant simplement un SoC ESP8266 NodeMCU (~8€) peuvent aussi permettre d'identifier des interfaces JTAG et UART, comme illustré en figure 12, page 71.

Après identification des différentes interfaces, des outils permettant de contacter ces interfaces existent comme le Bus Pirate v3 (v4 peu mature), le Shikra, ou encore le BusVoodoo supportant 14 protocoles **[35]**.

Contournement des protections

Dans le cas de l'interphone connecté, nous pouvions extraire le *firmware* du microcontrôleur sans problème, car aucune protection (*fuse bits*) n'avait été configurée à cet effet (voir la figure 13, page 71).

Mais dans beaucoup de cas, il peut en être autrement. Pour contourner ces protections, certains *bypasses* sont possibles, comme pour les PIC18F552, où certains blocs de code (*entry point* par exemple) peuvent être effacés et réécris afin de lire les autres blocs. Ce type d'attaque a permis d'extraire les clés d'authentification DES et chiffrement 3DES des cartes iCLASS **[38]** sur ces microcontrôleurs.

D'autres attaques dites de « glitching » (injections de fautes : voltage, *reset*, EMI, UV-C, etc.) plus ou moins invasives, permettent de provoquer des changements de chemin d'exécution et donc de gagner

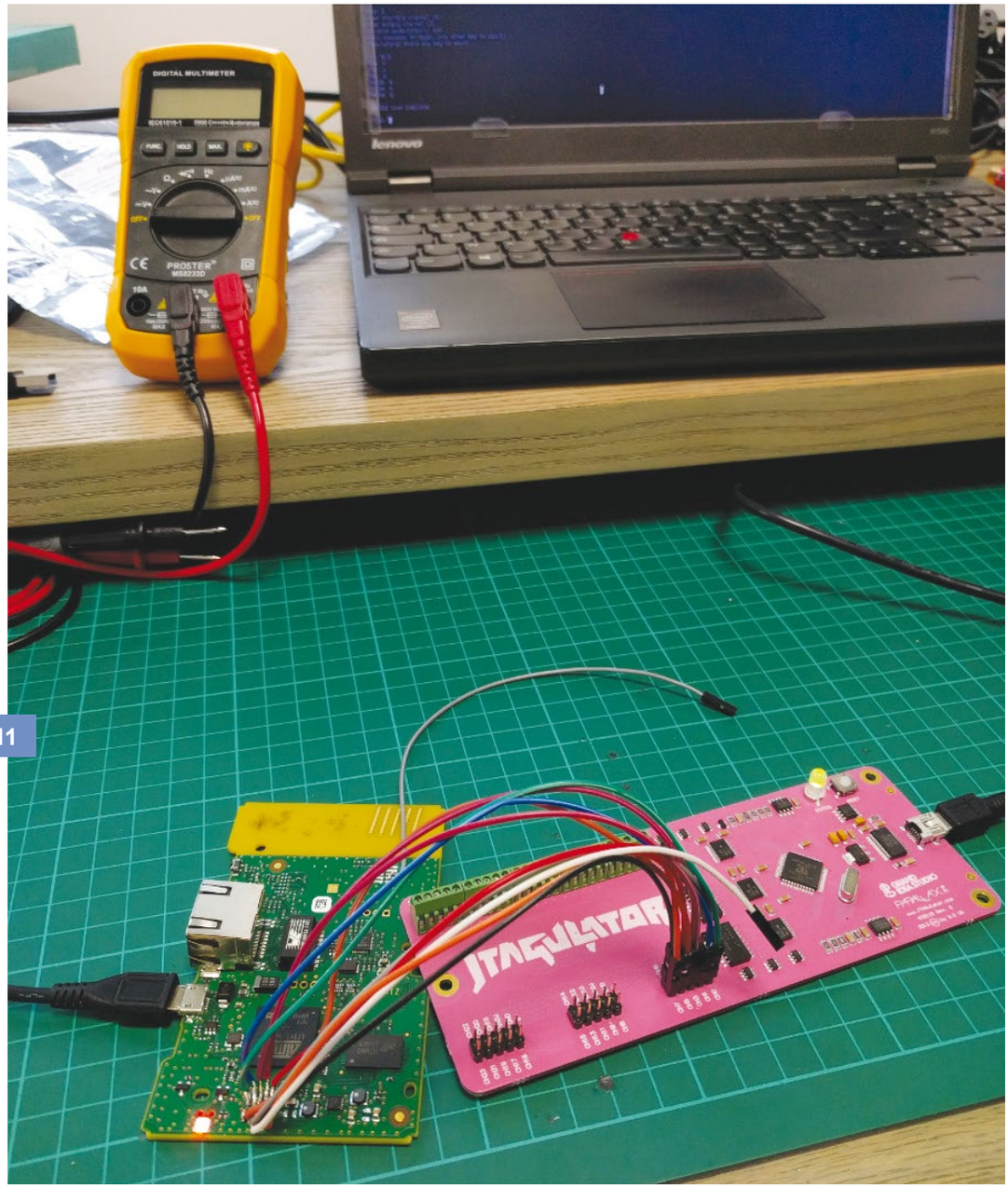


FIGURE 11

Bruteforce des pins JTAG avec un JTAGulator.

des accès sur le microcontrôleur. En effet, nous pourrons citer le *Cold-Boot stepping attacks* sur la série STM32F0 pouvant être reproduite avec peu de moyens **[30]**, mais aussi les attaques UV-C (UltraViolet-C) plus complexes **[29].**

Ces attaques peuvent prendre une toute autre ampleur avec l'utilisation de FPGA et permettre, par exemple, d'obtenir un retour UART sur les STM32F103 **[31]**. De plus, sur les séries STM32F1 et STM32F3, d'autres chercheurs ont récemment montré qu'il est possible d'abaisser la protection RDP2 (*no access*) de ces microcontrôleurs vers RDP1 (accès aux registres et RAM) pour lire la RAM du *hardware wallet* **TREZOR** (voir figure 14, page 72) lors d'une mise à jour de *firmware* **[32]**.

Et lorsque les interfaces sont manquantes, d'autres attaques plus intrusives peuvent aussi être entreprises :

- *sniffing* de bus ;
- *chip-off* de mémoire flash, comme en figure 15, page suivante ;
- et autres attaques encore plus sophistiquées.

CONCLUSION

Cet article a permis d'énumérer différentes méthodologies et moyens alternatifs pour attaquer les périphériques IoT utilisant les réseaux mobiles. L'attaque de *downgrade* vers réseaux 2G reste la méthode la plus générique pour réaliser des captures, car encore beaucoup de systèmes sont dotés d'une pile protocolaire 2G. Mais avec le LTE pour l'IoT, ainsi que la venue de la 5G, on

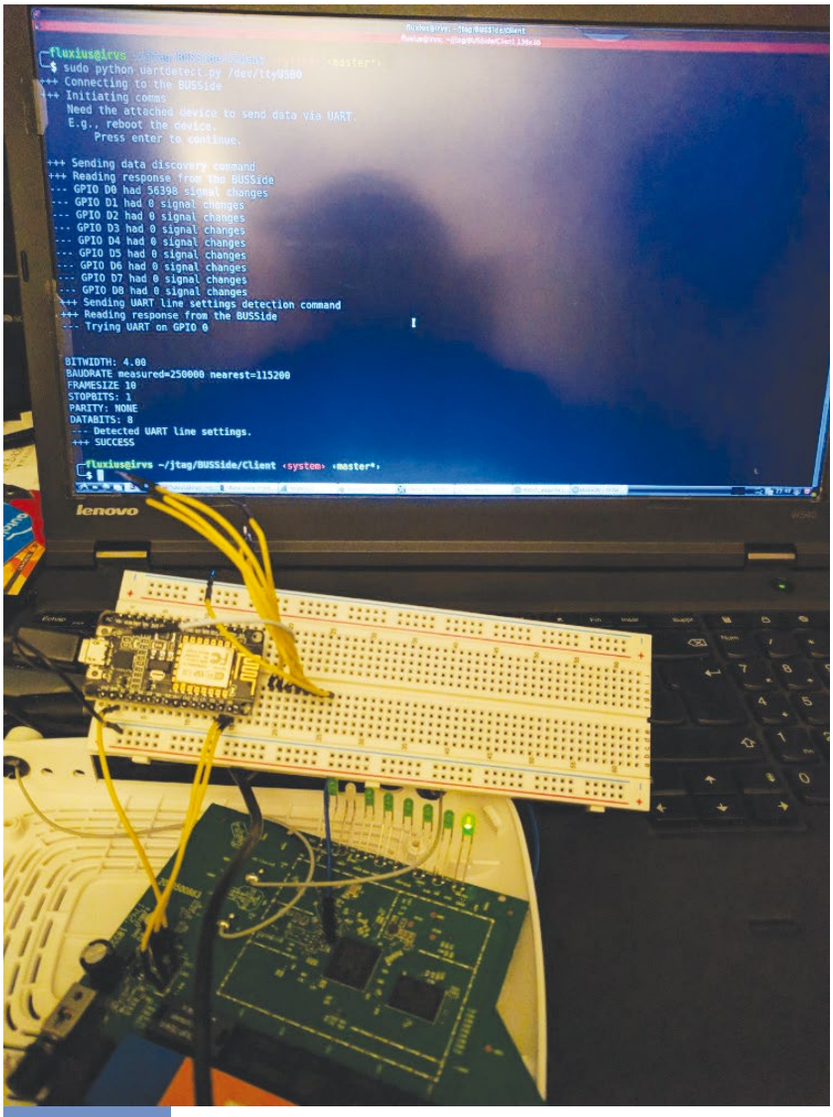


FIGURE 12 *Utilisation du BUSSide avec un ESP8266 pour bruteforcer les pins JTAG et UART.*

Output | Inspector | Configuration Bits ×

Address	Name	Value	Field	Option	Category	Setting
157FC	CONFIG2	7ABE	POSCMOD	HS	Primary Oscillator Select	HS Oscillator mode selected
			OSCIOFNC	OFF	Primary Oscillator Output Function	OSC2/CLKO/RC15 functions as CLKO (FOSC/2)
			FCKSM	CSDCMD	Clock Switching and Monitor	Clock switching and Fail-Safe Clock Monitor are disabled
			FNOSC	PRI	Oscillator Select	Primary Oscillator (XT, HS, EC)
			IESO	OFF	Internal External Switch Over Mode	IESO mode (Two-Speed Start-up) disabled
157FE	CONFIG1	3EF8	WDTPS	PS256	Watchdog Timer Postscaler	1:256
			FWPSA	PR128	WDT Prescaler	Prescaler ratio of 1:128
			WINDIS	ON	Watchdog Timer Window	Standard Watchdog Timer enabled,(Windowed-mode is disabled)
			FWDTEN	ON	Watchdog Timer Enable	Watchdog Timer is enabled
			ICS	PGx1	Comm Channel Select	Emulator/debugger uses EMUC1/EMUD1
			GWRP	OFF	General Code Segment Write Protect	Writes to program memory are allowed
			GCP	OFF	General Code Segment Code Protect	Code protection is disabled
			JTAGEN	OFF	JTAG Port Enable	JTAG port is disabled

FIGURE 13 *Configuration extraite du microcontrôleur de l'interphone connecté.*

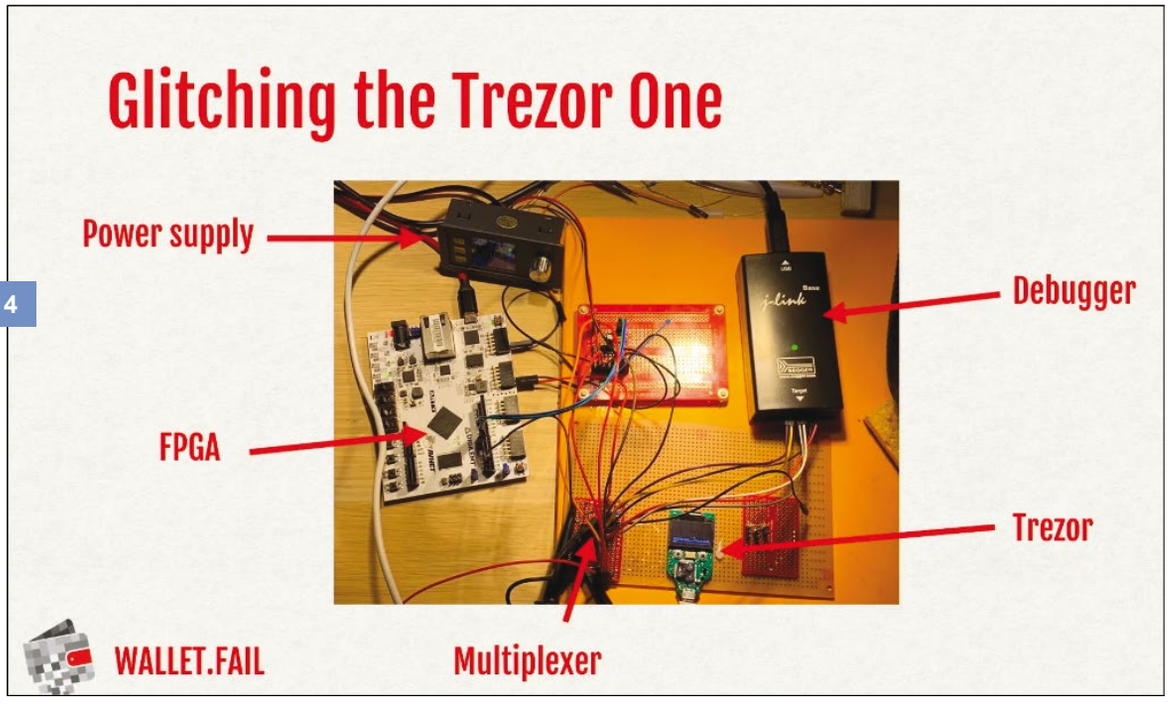


FIGURE 14

Downgrade du hardware wallet TREZOR (source : https://wallet.fail/).

peut s'attendre que ces périphériques soient à l'avenir moins accessibles. Un budget plus important va donc être nécessaire avant que des groupes puissent ouvrir d'autant plus les normes. Malgré cela, les attaques hardwares resteront non seulement une solution complémentaire, mais aussi une alternative pour aborder ce type d'équipement.

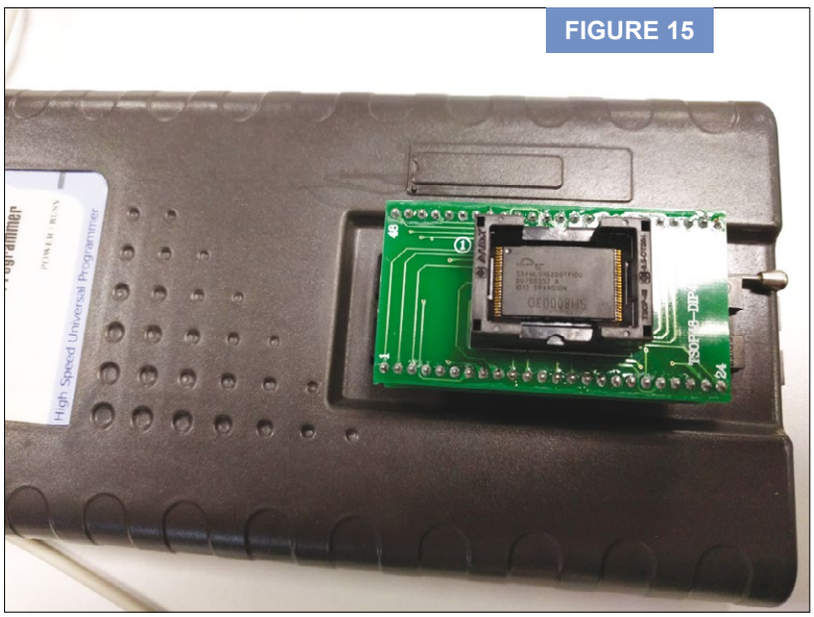


FIGURE 15

Extraction du firmware de flash au format TSOP48 avec programmateur.

REMERCIEMENTS ET RECONNAISSANCES

Je tiens à remercier mon entreprise Synacktiv, collègues et clients pour les différentes opportunités de mettre en pratique et améliorer les différentes attaques mobiles et *hardware*. Je tiens aussi à saluer les chercheurs et consultants sécurité, qui, à travers les discussions ou/et présentations, apportent toujours beaucoup de pointeurs, astuces et conseils dans l'univers mobile comme Christophe Devine et Benoit Michau ainsi que Sławomir Jasek, Raphaël Rigo, Philippe Paget, Kévin Redon et Silvio Cesare qui n'hésitent pas à échanger sur

les attaques de *glitching* et attaques *hardware* intéressants à moindre coût. Je tiens aussi à souligner l'importance des communautés ouvrant en grande partie l'univers mobile au public comme Range Networks, osmocom, SRS, Eurecom et bien d'autres.

Pour finir, je tiens à remercier les lecteurs qui sauront, je l'espère, apprécier cet article et sûrement donner leurs retours constructifs sur le sujet. ▌

RÉFÉRENCES

[1] Réseaux GSM par Xavier Lagrange, Philippe Godlewski, Sami Tabbane

[2] Projet OpenBTS : **http://openbts.org/**

[3] Projet Osmocom : **http://osmocom.org/**

[4] Analyse de sécurité des modems mobiles par Benoît Michau et avec la contribution de Christophe Devine : **https://www.sstic.org/media/SSTIC2014/SSTIC-actes/Analyse_securite_modems_mobiles/SSTIC2014-Article-Analyse_securite_modems_mobiles-michau.pdf**

[5] Radiocom 2000, collection historique : **http://collectionhistorique.orange.com/fr/nos-objets/Dossier-nos-objets/Radiocom-2000**

[6] AT&T announces it will build a fake 5G network : **https://www.theverge.com/2017/4/25/15425414/att-5g-evolution-network-lte-advanced-misleading-marketing**

[7] 5G V2X, 5G Automotive Association : **http://www.3gpp.org/ftp/information/presentations/Presentations_2017/A4Conf010_Dino%20Flore_5GAA_v1.pdf**

[8] ClockTamer installation : **https://wiki.myriadrf.org/ClockTamerUSRPInstallation**

[9] GPS disciplined oscillator, Wikipédia : **https://en.wikipedia.org/wiki/GPS_disciplined_oscillator**

[10] Sysmo NITB 2G starter kit : **https://www.sysmocom.de/products/lab/2Gstarterkit/index.html**

[11] Sysmo NITB 3.5G starter kit limited edition : **https://www.sysmocom.de/products/lab/3g5starterkit/index.html**

[12] LTE LabKitTM - Lab EnodeB and LTE Callbox : **https://yatebts.com/products/lte_lab_kit/**

[13] Exploitation Of A Modern Smartphone Baseband par Marco Grassi, Muqing Liu et Tianyi Xie : **https://paper.bobylive.com/Meeting_Papers/BlackHat/USA-2018/us-18-Grassi-Exploitation-of-a-Modern-Smartphone-Baseband-wp.pdf**

[14] Hacking Femtocells par Kevin Redon et présenté par Ravishankar Borgaonkar : **https://www.isti.tu-berlin.de/fileadmin/fg214/Papers/conf_t2_2010.pdf**

[15] Projet osmocomBB : **http://bb.osmocom.org/trac/**

[16] Modmobmap, BeeRump 2018 par Sébastien Dudek : **https://www.rump.beer/2018/slides/modmobmap.pdf**

[17] Projet Modmobmap : **https://github.com/Synacktiv/Modmobmap**

[18] Modmobjam, RUMP SSTIC 2018 par Sébastien Dudek : **https://www.synacktiv.com/ressources/sstic_rump_2018_modmobjam.pdf**

[19] Projet Modmobjam : **https://github.com/Synacktiv/Modmobjam**

[20] Penthertz: the use of radio attacks in red team and pentests, Security PWNing 2018 par Sébastien Dudek : **https://www.synacktiv.com/ressources/SecurityPWNing2018-PentHertz_using_radio_during_redteam_tests.pdf**

[21] USIM chinoises personnalisables, Taobao : **https://item.taobao.com/item.htm?spm=a21wu.10013406.0.0.568e3659Nt3blw&id=564374220567**

[22] Build a LTE Network with srsLTE and Program Your Own USIM Card : **https://cyberloginit.com/2018/05/03/build-a-lte-network-with-srslte-and-program-your-own-usim-card.html**

[23] Using GPRS with Local breakout in YateBTS : **https://wiki.yatebts.com/index.php/Using_GPRS_with_Local_breakout_in_YateBTS**

[24] Produits Amarisoft : **https://www.amarisoft.com/products-lte-ue-ots-sdr-pcie/#network**

[25] Projet OpenAirInterface5G : **https://gitlab.eurecom.fr/oai/openairinterface5g**

[26] Projet NextEPC : **http://nextepc.org/**

[27] Mobile Authentication Subspace Travel, HITB 2015, par Marku Vervier : **https://conference.hitb.org/hitbsecconf2015ams/wp-content/uploads/2014/12/WHITEPAPER-Mobile-Authentication-Subspace-Travel.pdf**

[28] How to not break LTE crypto, SSTIC 2016, par Benoi Michau et Chistophe Devine : **https://www.sstic.org/media/SSTIC2016/SSTIC-actes/how_to_not_break_lte_crypto/SSTIC2016-Article-how_to_not_break_lte_crypto-michau_devine.pdf**

[29] Shedding too much Light on a Microcontroller's Firmware Protection : **https://www.aisec.fraunhofer.de/content/dam/aisec/ResearchExcellence/woot17-paper-obermaier.pdf**

[30] Dumping the flash of the Syscan 2015 badge par Emilien Girault : **https://gist.github.com/egirault/7b3fe7041e1bf5e2258ed5df7083f14d**

[31] UART controlled glitch on an iCE40 icestick : **https://threadreaderapp.com/thread/1006286104939593730.html**

[32] Attaque des hardware wallets par Dmitry Nedospasov, Josh Datko et Thomas Roth : **https://wallet.fail/**

[33] JTAG Explained (finally!) : **https://blog.senr.io/blog/jtag-explained**

[34] JTAGulator : **http://www.grandideastudio.com/jtagulator/**

[35] BusVoodoo : **https://bus.cuvoodoo.info/manual/index.html#_protocols**

[36] BUSSide : **http://busside.com.au/**

[37] Projet Pycrate : **https://github.com/P1sec/pycrate**

[38] Heart of Darkness - exploring the uncharted backwaters of HID iCLASSTM security : **https://www.openpcd.org/images/HID-iCLASS-security.pdf**

ACTUELLEMENT DISPONIBLE

LINUX PRATIQUE N°111

TIREZ LE MEILLEUR DE VOTRE LIGNE DE COMMANDES !

NE LE MANQUEZ PAS

CHEZ VOTRE MARCHAND DE JOURNAUX ET SUR :

https://www.ed-diamond.com

DES TRACEURS GPS BIEN TROP INDISCRETS

Pierre BARRE & Chaouki KASMI

Mobile & Telecom Lab/Xen1thlabs DARKMATTER LLC, Abu Dhabi/EAU

Le suivi d'individus et de biens à forte valeur ajoutée est depuis plusieurs années réalisé par l'utilisation de traceurs GPS. Ces équipements ont pour fonction de transmettre les coordonnées de géolocalisation périodiquement à un serveur distant via un module radio mobile et une carte SIM. Il a été également constaté que certaines de ces balises intègrent un microphone fournissant un accès distant à une fonction d'espionnage. Considérant différents vecteurs d'attaque, nous nous sommes intéressés aux méthodes de compromission possibles démontrant un trop faible niveau de sécurité de ces solutions. Nous proposons dans cet article de décrire la méthodologie appliquée pour l'évaluation du niveau de sécurité de différentes balises GPS disponibles sur le marché. Plusieurs vulnérabilités sont décrites démontrant les risques liés à l'emploi de cette technologie pour le suivi des biens sensibles et des trajets de personnes importantes.

Photo d'illustration.

L'utilisation de traceurs GPS dans les domaines privés et publics, parfois à la limite de la légalité, a vu une très forte hausse. Avec l'objectif de suivre un bien ou une personne, le traceur GPS a pour fonction de transmettre périodiquement ou à la demande les coordonnées GPS de sa localisation. Un traceur GPS se connecte via le réseau mobile à l'aide d'une carte SIM/USIM (en fonction des modèles) afin de transmettre à intervalle régulier les informations obtenues par son module de réception GPS ainsi que par message texte vers un numéro préenregistré lorsque celui-ci est sollicité par le propriétaire du traceur. Certains traceurs dits « évolués » intègrent en plus en fonction des modèles : une fonction d'enregistrement de l'environnement sonore sur demande, des interfaces radio complémentaires (ex. Wifi) ainsi que des détecteurs de mouvement. La configuration de ces traceurs est réalisée par la transmission de messages texte (SMS) dont les détails sont fournis dans la documentation technique transmise par le constructeur au moment de l'achat. Le propriétaire peut ainsi consulter la position du traceur via une application smartphone, via la transmission de SMS ou directement via l'application web. Les données sont transmises à un serveur distant où une application va traiter et mettre en forme les coordonnées GPS sur une carte géographique afin que l'utilisateur prenne connaissance de la position du boîtier de la plus simple des façons. Les coordonnées GPS peuvent également être envoyées par SMS directement à l'utilisateur au format texte afin de permettre un suivi décentralisé. L'utilisateur peut ensuite utiliser ces coordonnées sur un service tel que Google Maps afin de prendre connaissance du lieu géographique. Il a également été constaté que certains de ces traceurs GPS intègrent des fonctionnalités dites « évoluées » comme par exemple un détecteur de mouvement et/ou un microphone ainsi que des capacités de gestion de mise en fonctionnement et d'arrêt de véhicules à distance.

Le niveau de sécurité de ces équipements de suivi et d'espionnage est donc stratégique afin de ne pas fournir à un tiers malveillant les mêmes accès. Il est logiquement nécessaire d'évaluer le niveau de sécurité avec les outils adéquats et d'estimer le risque introduit par l'usage de ces solutions. Peu d'études ont été consacrées à l'analyse complète des traceurs GPS et des infrastructures. L'étude la plus complète que nous recommandons est l'étude TrackMagedon **[1]** mettant à jour des problèmes de sécurité dans les interfaces de gestion démontrant que les données des utilisateurs sont stockées et disponibles pour n'importe quel tiers malveillant ayant quelques connaissances en sécurité des technologies web.

Nous proposons dans cette étude une analyse en profondeur des vecteurs d'attaque possibles ainsi que les résultats des tests effectués sur différents boîtiers du commerce. Plusieurs vulnérabilités critiques démontrent que le niveau de sécurité est faible et que des millions de boîtiers aujourd'hui sur le marché mettent en péril la vie privée des personnes équipées du dispositif (avec ou sans consentement) et potentiellement la résilience des véhicules intégrant des dispositifs évolués permettant l'arrêt à distance de leur moteur. Par ailleurs, nous avons pu constater que les informations remontées vers des serveurs distants intègrent également le numéro des stations de base (CellID) en complément permettant donc aux constructeurs de cartographier les réseaux de télécommunications de différents pays.

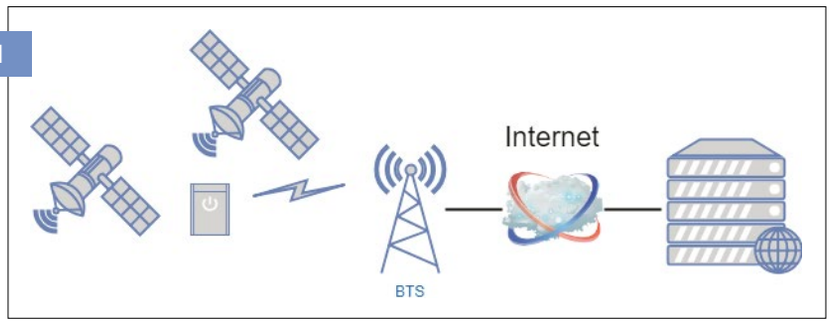
SCÉNARIOS D'ATTAQUE

Un schéma représentant l'architecture générale et les points d'interaction est proposé. Comme il peut être constaté, les vecteurs d'attaque sont nombreux et la surface d'attaque est assez conséquente : les interfaces radio (GPS, réseaux mobiles), les serveurs distants, les applications pour smartphone ainsi que les

protocoles de gestion et de configuration. Par ailleurs, les mécanismes de mise à jour quand ils existent sont également des vecteurs d'attaque intéressants à exploiter. En fonction de l'objectif d'un attaquant, les techniques et outils (majoritairement open source) seront à adapter. Nous proposons dans cette section de traiter les points d'intérêt pour un attaquant en fonction de son objectif sans motiver les lecteurs à réaliser ces différentes attaques du fait de leur illégalité.

FIGURE 1

Schéma de principe de communication d'un traceur GPS.



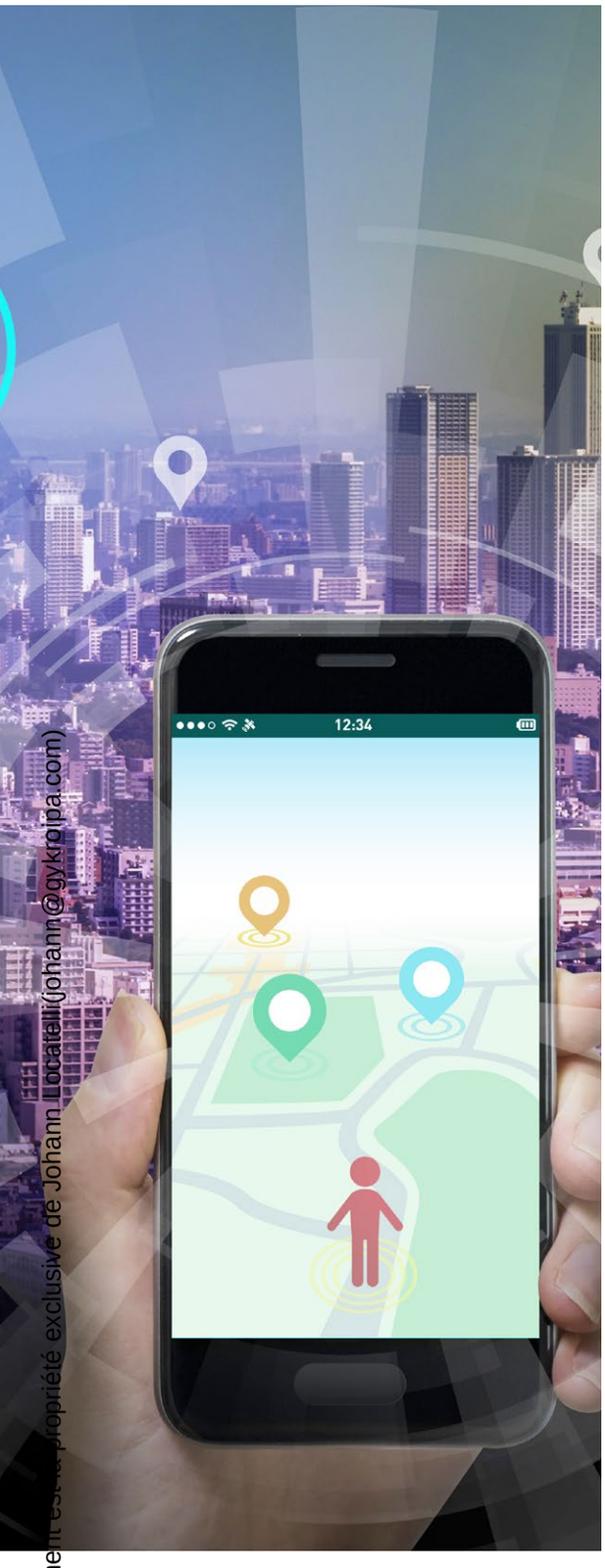
Brouillage et leurre GPS

Un attaquant dont l'objectif est de bloquer la transmission des coordonnées GPS vers les serveurs distants peut ainsi brouiller le signal à l'aide de dispositifs à bas coût disponibles sur Internet. Cette attaque de faible niveau technique est cependant difficile à couvrir si le système de contrôle vérifie le statut du boîtier et/ou corrèle le manque de données avec des informations tierces comme les CellID. Afin de réduire la détection par le système de contrôle qu'une action malveillante est en cours, l'attaquant peut leurrer une constellation GPS pour modifier ou étendre la zone de couverture du dispositif afin de changer la position géographique. Un article complet sur la méthodologie et les outils portant sur le leurrage GPS est inclus dans ce numéro du *MISC* et nous renvoyons le lecteur vers celui-ci. Il est cependant à noter ici que la complexité de l'attaque est faible puisque les outils et leurs utilisations sont très bien documentés.

Attaque du protocole de gestion

Avec l'objectif de reconfigurer et de remonter les données vers sa propre infrastructure, un attaquant peut chercher à changer les adresses des serveurs de gestion et forcer les traceurs à transmettre les coordonnées GPS vers cette infrastructure. Afin d'éviter d'être détecté, il pourra ensuite transmettre les données en les modifiant à la volée vers l'infrastructure légitime. Cette attaque nécessite évidemment de connaître le protocole de management et le numéro de la SIM provisionnée dans le traceur et exploiter des vulnérabilités du protocole de gestion. La complexité de cette attaque est directement liée au niveau de sécurité du protocole de gestion et de son implémentation.

Photo d'illustration.



Attaque des serveurs distants et des applications web et pour smartphone

Un attaquant ayant pour objectif de compromettre l'intégralité des données issues des traceurs GPS pour un constructeur donné pourra directement chercher à attaquer l'infrastructure et les applications de gestion à distance mises à disposition des utilisateurs. En se procurant un traceur, l'attaquant pourra réaliser la rétroconception des protocoles, des firmwares et du matériel. Ensuite, une analyse des mécanismes d'authentification, d'identification et des APIs disponibles permettra de trouver des vecteurs de compromission. La richesse des informations accessibles fera que ce scénario d'attaque est le plus efficace puisqu'il permet potentiellement d'attaquer un très grand nombre de traceurs.

Dans la suite de cette étude, nous proposons les plans expérimentaux et les résultats associés démontrant le faible niveau de sécurité des traceurs GPS et des infrastructures associées. Afin de rendre cette étude la plus large possible, nous nous sommes procuré les traceurs GPS ayant été les plus vendus sur Internet.

ÉVALUATION DE MODULES DU COMMERCE

Analyse de la remontée de données

Dans un premier temps, une analyse du trafic réseau a été réalisée afin de comprendre le type de données échangées ainsi que les protocoles de communication. Pour cela, nous nous sommes placés en boite noire et nous avons mis en place une BTS 2G/2.5G (GPRS) sans authentification/chiffrement (configuration de base de la station 2G) utilisant Yatebts. Il faut noter que la majorité des traceurs ne supporte que les cartes SIM (les auteurs se sont arraché les cheveux avec des cartes uSIM).

Pour des contraintes règlementaires, nous plaçons notre dispositif RF dans une cage de Faraday. Chacun de nos traceurs GPS est équipé d'une carte SIM et un numéro unique est attribué. Une fois que la BTS est opérationnelle, nous avons sniffé l'interface GPRS (`sgsntun`) pour récupérer le trafic réseau de chaque traceur GPS. Finalement, nous avons utilisé les commandes SMS préconisées par le constructeur afin de paramétrer chacun de nos traceurs.

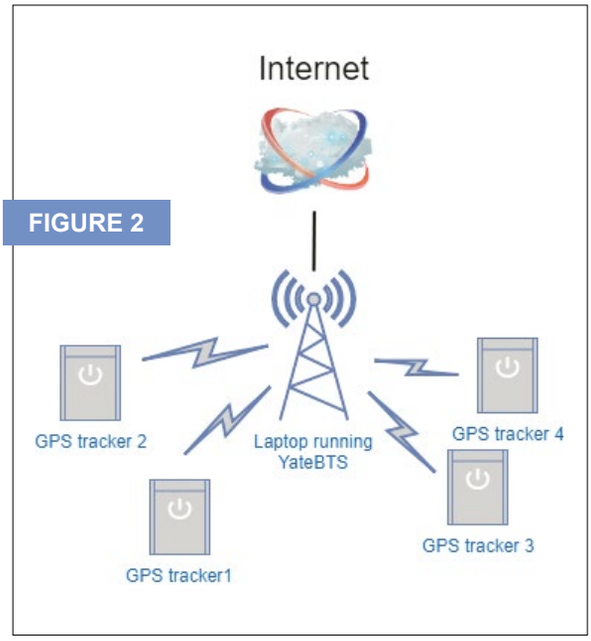


FIGURE 2

Mise en place d'un BTS permettant l'interception des flux réseaux en 2/2.5G.

Durant cette phase de configuration, nous avons pu constater les premiers éléments critiques suivants :

- tous les traceurs envoient les coordonnées vers des IPs en Chine ;
- le trafic n'est pas chiffré et est facilement identifiable (noms de domaine spécifiques, IPs spécifiques, ports spécifiques) – une implémentation de DPI est triviale à effectuer pour ce type de protocole ;
- le mot de passe d'une plateforme est véhiculé en clair, les autres plateformes utilisent le numéro de série des traceurs (prédictibles) comme token sans système d'authentification, permettant à un attaquant d'envoyer des informations forgées ;
- il est possible, via un SMS, d'éditer la configuration du traceur et de spécifier un autre serveur de management (« *reg mon_ip »), permettant à un attaquant d'effectuer un MITM simple via un serveur sur Internet relayant le trafic (UDP ou TCP – selon les traceurs), avec par exemple **balance** (1) pour du TCP :

```
balance -b ::ffff:mon_ip 8841 203.130.62.29:8841
```

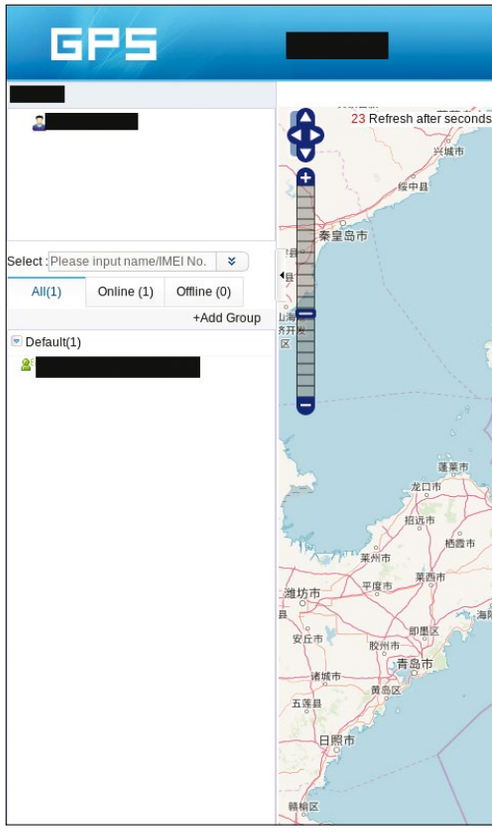
- il est possible de définir un numéro de téléphone master. En cas d'usurpation de **Caller-ID**, cette authentification est contournable. De plus, même via un numéro de téléphone master défini, il est possible d'envoyer des commandes via SMS et d'avoir des informations sur le traceur en retour, permettant ainsi de détecter des traceurs sur un ensemble de numéros de téléphone (cette méthode ne sera par contre pas discrète).

Les traceurs utilisant un module GPS U-BLOX se connectent à un service TCP sur le port 56447/tcp. Par défaut, le client envoie son login, password, latitude, longitude et altitude en clair au début de la session TCP :

```
cmd=full;user=XXXXXX@gmail.com;pwd=XXXXXX;lat=
22.680193;lon=114.146846;alt=0.0;pacc=100.00
```

Le serveur répond en spécifiant un blob propriétaire :

```
u-blox a-gps server (c) 1997-2009 u-blox AG
Content-Length: 2856

Content-Type: application/ubx


.b..0......
```



Le client envoie ensuite régulièrement des informations à un deuxième serveur (8011/tcp) en indiquant sa position :

```
*HQ,17000XXXXX,V1,115112,A,2240.8116,N,11408.8108,E,000.0,000.00,100119
,FFFFFFFF#
*HQ,17000XXXXX,NBR,094111,310,26,02,1,1000,10,23,100119,FFFFFFFF#
*HQ,17000XXXXX,LINK,115112,22,0,6,0,0,100119,FFFFFFFF#
*HQ,17000XXXXX,NBR,115117,310,26,02,1,1000,10,22,100119,FFFFFFFF#
```

On peut détecter différentes commandes suivant le numéro de série (17000XXXXX). 2240.8116 correspond à la latitude 22.408116, 11408.8108 à la latitude 11.4088108. Il n'y a pas d'authentification, ce qui nous a permis de créer notre propre client GPS envoyant des coordonnées GPS sous contrôle. On peut voir que notre traceur GPS se trouve actuellement à Pyongyang en Corée du Nord (Figure 3).

Un second traceur (le plus économique) ne possède aucun élément d'authentification permettant au serveur de vérifier la provenance des données collectées (en SMS ou sur les données émises en 2G). Les données envoyées vers la plateforme de management sont toujours de la forme :

```
\x79\x79\x00 I \xf2 [S/N-ASCII][BLOB-ASCII] \x01\x7e [BLOB-ASCII]
```

Un point intéressant est qu'il envoie en Chine l'intégralité des SMS reçus sur son interface radio – permettant au gestionnaire de la plateforme de savoir exactement ce que veut faire l'utilisateur (Figure 4, page suivante).

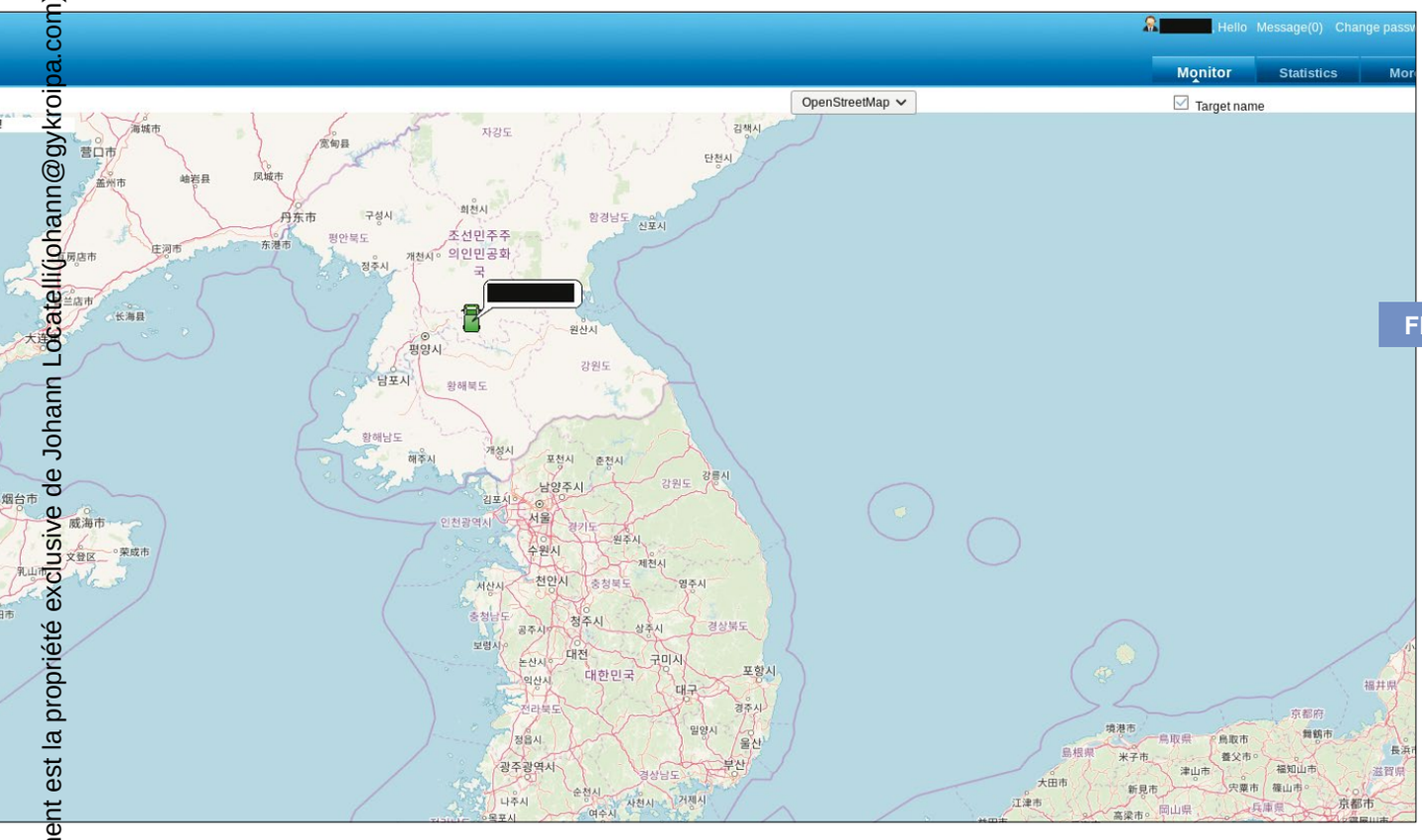


FIGURE 3

Résultat d'un envoi de coordonnées GPS forgées – les auteurs ne sont pas allés en Corée du Nord pour cet article.

FIGURE 4

Paquet TCP envoyé par le traceur à la plateforme de gestion distante.

```
▸ Frame 316: 119 bytes on wire (952 bits), 119 bytes captured (952 bits)
  Raw packet data
▸ Internet Protocol Version 4, Src: 192.168.99.2, Dst: 203.130.62.29
▸ Transmission Control Protocol, Src Port: 58651, Dst Port: 8841, Seq: 118, Ack: 31, Len: 79
▾ Data (79 bytes)
    Data: 79790049f2363930323137313232363132343633383938 38...
    [Length: 79]

0000  45 00 00 77 6f 16 40 00  7f 06 5f 20 c0 a8 63 02   E··wo·@· ··_ ··c·
0010  cb 82 3e 1d e5 1b 22 89  00 8e 28 10 3f e5 55 4d   ··>···"· ··(·?·UM
0020  50 18 2a 80 12 b6 00 00  79 79 00 49 f2 36 39 30   P·*····· yy·I·690
0030  32 31 37 31 32 32 36 31  32 34 36 33 38 39 38 38   21712261 24638988
0040  32 31 31 30 30 30 30 30  30 32 37 36 34 30 35 46   21100000 0276405F
0050  01 7e 31 39 30 31 30 39  31 30 35 34 31 37 0a 2b   ·~190109 105417·+
0060  34 34 30 30 32 35 32 33  39 01 06 53 74 61 74 75   44002523 9··Statu
0070  73 00 05 30 f3 0d 0a                               s··0···
```

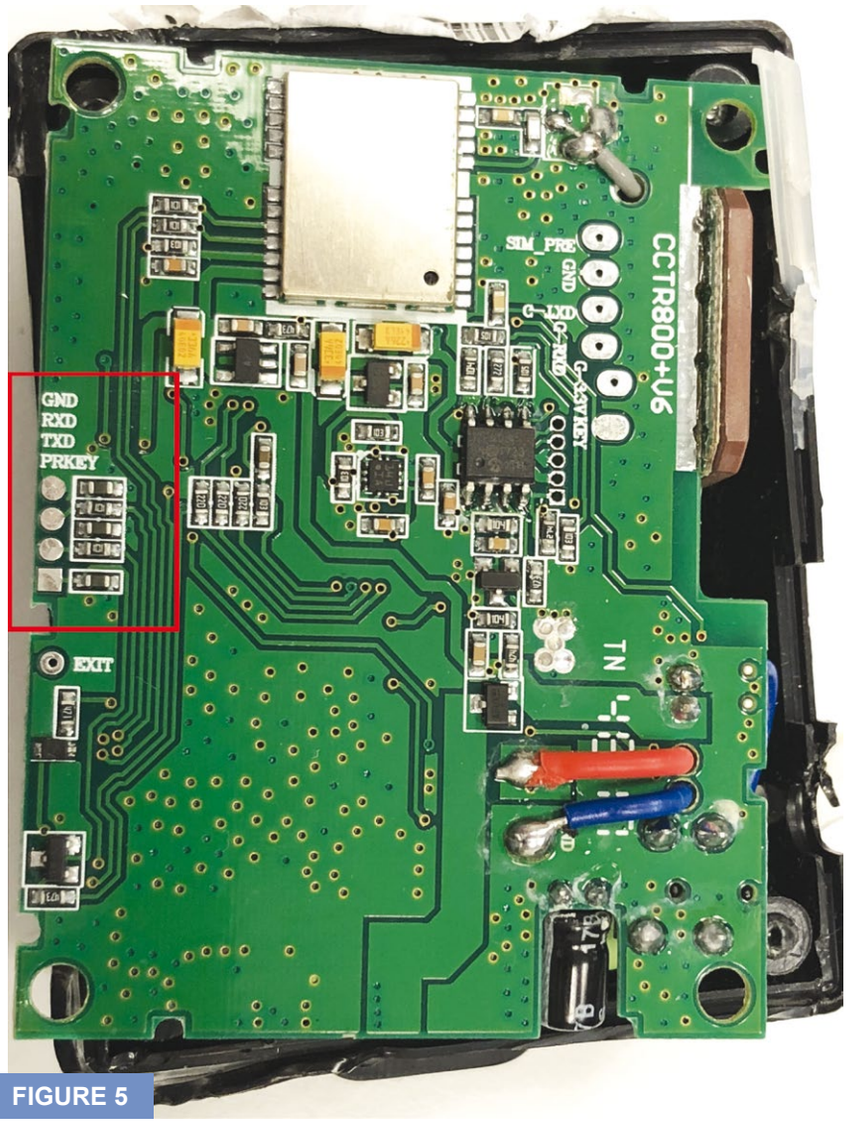


FIGURE 5

Carte d'un traceur GPS - l'interface série est entourée.

Suite à l'envoi d'un SMS « Status » vers le traceur GPS, celui-ci envoie un paquet TCP vers 203.130.62.29:8841/tcp, IP géolocalisée en Chine, mais se trouvant en réalité aux Émirats Arabes Unis et annoncée par l'opérateur Etisalat, contenant le message « Status », avec d'autres informations : 690217122612463 correspondant au S/N du traceur GPS et +440025239 étant le numéro de l'émetteur du SMS. Cela marche pour tout message SMS, même s'il ne correspond pas à un mot-clé spécifique permettant la gestion du traceur. Une utilisation détournée du traceur permet, via la redéfinition du serveur de management, d'avoir un système relais de SMS à bas coût (< 15 euros), solution idéale pour les expats recevant des SMS de validation vers un numéro français.

De manière générale, la sécurité réseau des traceurs GPS est très mauvaise, le trafic est en clair et est facilement compréhensible. Un attaquant peut envoyer de fausses informations sur l'infrastructure de management GPS à condition de posséder le numéro de série du traceur.

Photo d'illustration.

Rétroconception du matériel

L'ouverture des traceurs a permis de mettre en évidence la simplicité de l'électronique utilisée. On trouve principalement de vieux chipsets MediaTek (MT6261 ARM) et des chipsets GPS (par exemple U-BLOX). Des interfaces de debug sont disponibles permettant l'extraction des firmwares et des éléments de configurations par défaut. On notera la simplicité de cette partie puisqu'aucune protection contre les attaques physiques n'est intégrée (Figure 5, ci-contre).

Une analyse a été effectuée sur le dump d'un des firmwares. Pour celui-ci, le système embarqué est Nucleus RTOS et la taille de l'OS est conséquente pour du matériel embarqué (4Mo). Le reverse de ce même dump a permis de révéler la présence de codes SMS backdoors dans différents traceurs GPS. Ces codes SMS ne semblent pas être tous fonctionnels – le firmware est visiblement utilisé sur de nombreux traceurs GPS aux fonctionnalités variables (Figure 6).

```
OM:0006987F                     DCB    0
OM:00069880 aReboot             DCB "*reboot*",0
OM:00069889                     DCB    0
OM:0006988A                     DCB    0
OM:0006988B                     DCB    0
OM:0006988C                     DCB    0
OM:0006988D                     DCB    0
OM:0006988E                     DCB    0
OM:0006988F                     DCB    0
OM:00069890                     DCB 0x2B ; +
OM:00069891                     DCB    0
OM:00069892                     DCB    0
OM:00069893                     DCB    0
OM:00069894 a3646655            DCB "*3646655*",0
OM:0006989E                     DCB    0
OM:0006989F                     DCB    0
OM:000698A0                     DCB    0
OM:000698A1                     DCB    0
OM:000698A2                     DCB    0
OM:000698A3                     DCB    0
OM:000698A4                     DCB 0x2C ; ,
OM:000698A5                     DCB    0
OM:000698A6                     DCB    0
OM:000698A7                     DCB    0
OM:000698A8 aImeiset            DCB "imeiset",0
OM:000698B0                     DCB    0
```

FIGURE 6

Suites de strings correspondant aux commandes supportées par le traceur GPS.

L'analyse du firmware n'a pas révélé de fonctionnalités cachées majeures, mais a plutôt permis de voir que le développement semble avoir été fait rapidement et que de nombreuses commandes ne fonctionnent pas. Nous n'avons pas non plus identifié de fonctionnalités de mise à jour du firmware – il s'agit donc de matériel jetable.

Analyse sécurité des applications mobiles

Le constructeur fournit des applications pour smartphones iOS/Android, permettant d'avoir accès aux informations remontées par le traceur via l'intermédiaire d'une carte mondiale. Une rétro-ingénierie statique et dynamique a été effectuée sur le client Android. Par ailleurs, l'analyse statique via l'utilitaire jadx **[2]** a permis de trouver un système d'échange d'informations via une API accessible en http.

En vérifiant avec l'analyse dynamique, il est apparu que les échanges entre le smartphone et le serveur distant se font bien via http à travers un système d'API propriétaire (**http://m.999gps.net/OpenAPIV2.asmx**) – on peut voir l'identifiant 82383 – correspondant au traceur – dans la requête :

```
<v:Envelope xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns:d="http://www.w3.org/2001/XMLSchema"
xmlns:c="http://schemas.xmlsoap.org/soap/encoding/" xmlns:v="http://schemas.xmlsoap.org/soap/envelope/"><v:Header
/><v:Body><GetTracking xmlns="http://tempuri.org/" id="o0" c:root="1"><DeviceID i:type="d:int">82383</
DeviceID><TimeZone i:type="d:string">UTC</TimeZone><MapType i:type="d:string">Google</MapType></GetTracking></
v:Body></v:Envelope>
HTTP/1.1 200 OK
Cache-Control: private, max-age=0
Content-Type: text/xml; charset=utf-8
Content-Encoding: gzip
Vary: Accept-Encoding
Server: Microsoft-IIS/7.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Sun, 13 Jan 2019 07:17:09 GMT
Connection: close
Content-Length: 438

<?xml version="1.0" encoding="utf-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/
XMLSchema"><soap:Body><GetTrackingResponse xmlns="http://
tempuri.org/"><GetTrackingResult>{"state":"0","deviceUtcDate":"2019-01-12
16:43:24","latitude":"39.056417","longitude":"126.257200","olatitude":"39.056417","olongitude":"126.257200","speed
":"0.00","course":"0","isStop":"1","icon":"27_0","distance":"0","acc":"0"}</GetTrackingResult></
GetTrackingResponse></soap:Body></soap:Envelope>
```

FIGURE 7

Requête http envoyée depuis l'application Android vers les APIs.

Il est aussi possible d'accéder au WSDL (*Web Services Description Language*) de l'API en rajoutant **?WSDL** à l'adresse afin de récupérer un descriptif complet du service. Finalement, le constructeur fournit une interface de debug en ligne (Figure 8).

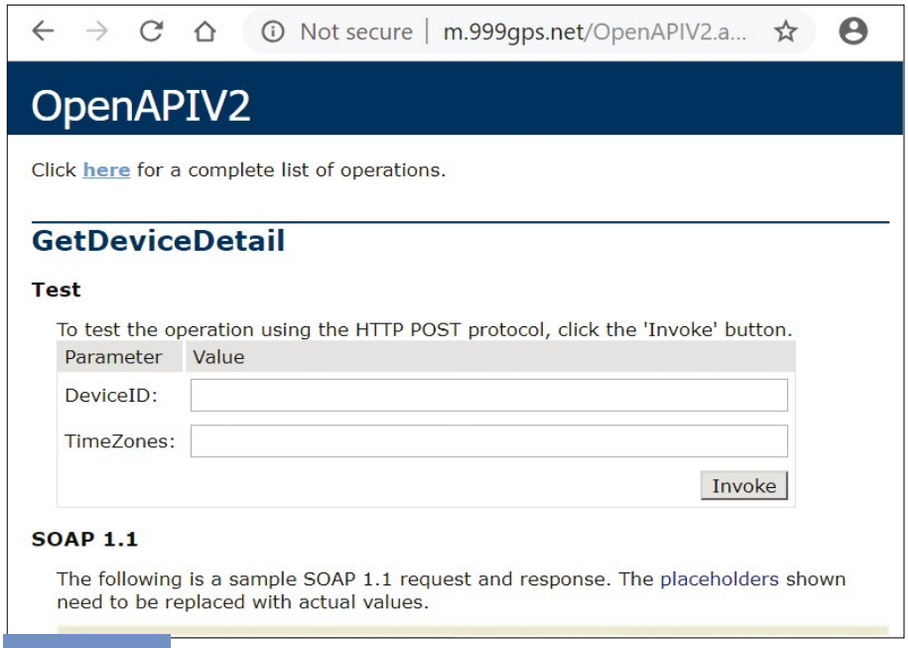


FIGURE 8 *Interface de debug sur les APIs.*

En analysant le trafic, il est apparu que l'authentification est inexistante et que rejouer la requête analysée précédemment par Wireshark en changeant l'identifiant permet de récupérer les coordonnées d'un autre traceur. Cette vulnérabilité avait été déjà signalée par l'équipe de Trackmaggedon **[1]** en janvier 2018, mais n'a jamais été corrigée par le constructeur.

Photo d'illustration.

Sécurité des sites web de gestion des traceurs et attaques avancées

Lors de l'achat d'un traceur, il est fourni un identifiant et un mot de passe pour un site web permettant d'accéder aux informations remontées par le traceur. En analysant ces sites web, il est apparu de nombreux problèmes :

- Par défaut, le nom d'utilisateur et le mot de passe sont les 7 derniers caractères du numéro de série du traceur GPS. Les utilisateurs ne semblent pas être au courant et ne changent pas forcément les mots de passe. Il est ainsi possible pour un attaquant de provisionner l'ensemble des comptes disponibles.

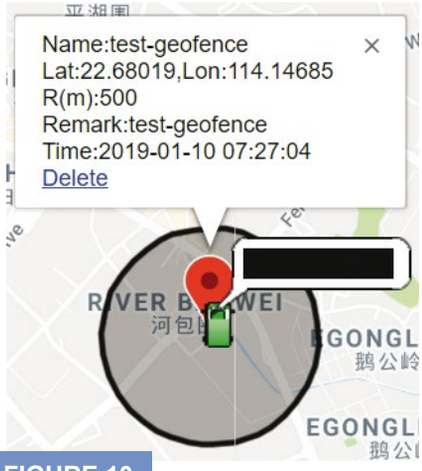


FIGURE 10 *Définition d'une Geofence.*

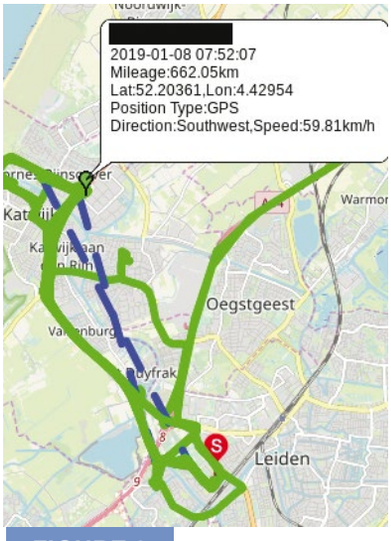


FIGURE 9 *Traceur GPS en Hollande.*

- Il a été possible, depuis un compte d'un traceur GPS, d'accéder aux coordonnées d'un autre traceur GPS en notre possession en indiquant l'ID de notre traceur dans la requête http – il s'agit d'une vulnérabilité de type *insecure direct object reference*, à condition de posséder une session valide (c'est-à-dire de posséder juste un compte sur la plateforme). De plus, il est aussi possible d'ajouter une *geofence* à d'autres traceurs, provoquant une alerte si le véhicule sort de la zone ou permettant d'aller jusqu'à l'arrêt du moteur.

L'interface permet d'avoir accès à l'intégralité de l'historique des déplacements des traceurs.

CONCLUSION

Dans le cadre de cette étude, nous nous sommes intéressés à la sécurité de traceurs GPS dont la popularité est grandissante. Nous avons pu constater que les différents scénarios d'attaque peuvent être facilement exploités afin d'obtenir les informations de localisation des traceurs GPS de façon illégitime et que la protection des échanges de données était pour l'intégralité des traceurs quasiment inexistante. Les interfaces de configuration via SMS dont l'authentification est soit inexistante, soit contournable via l'usurpation du CallerID exposent les utilisateurs à une atteinte à leur vie privée, mais aussi à des problèmes du fait des fonctionnalités avancées de certains modèles comme l'arrêt du véhicule à distance. De nombreux problèmes de sécurité ont été trouvés dans les APIs et les sites web.

Évidemment, les différentes vulnérabilités ne permettent pas un unique type d'exploitation. Les informations étant extrêmement complètes, comportant les informations du réseau de télécommunications environnant et les coordonnées GPS, celles-ci permettent pour un attaquant de cartographier de façon efficace les infrastructures critiques de téléphonie mobile d'un pays. Les trajets et les lieux sont également des informations stratégiques puisque corrélés aux horaires de travail, un attaquant est capable de connaître quand et où une cible travaille ainsi que son lieu de résidence. L'intégralité des données est déjà présente sur des serveurs de pays tiers rappelant le bug logiciel du transfert des coordonnées GPS des Cell-IDs dans iOS **[3]**.

Enfin, nous avons pu constater que la protection des données est très faible et la notion de GDPR est inexistante pour les fournisseurs de traceurs GPS. Finalement, des résultats complémentaires seront présentés durant la conférence Hack In Paris 2019.

REMERCIEMENTS

Ce travail a été effectué durant notre temps personnel de recherche. Nous souhaitons remercier DARKMATTER LLC de nous permettre de publier nos travaux portant sur la sécurité des objets connectés. Les opinions et les résultats présentés dans cet article n'engagent que leurs auteurs. Nous souhaitons aussi remercier Eiman Al Shehhi, Mobile & Telecom Lab, DARKMATTER LLC, pour son soutien durant l'analyse technique.

RÉFÉRENCES

[1] **https://0x0.li/trackmageddon/**

[2] **https://github.com/skylot/jadx**

[3] **https://www.apple.com/newsroom/2011/04/27Apple-Q-A-on-Location-Data/**

LEURRAGE DU GPS PAR RADIO LOGICIELLE

G. GOAVEC-MEROU, J.-M FRIEDT – FEMTO-ST temps-fréquence, Besançon

F. MEYER – OSU Theta, Observatoire de Besançon

Le système de navigation GPS est avant tout un système de dissémination de temps utilisé comme référence dans une multitude d'applications nécessitant une synchronisation de sites géographiquement distants. Nous démontrons ici comment une implémentation en radio logicielle des trames émises par les satellites permet de leurrer un récepteur en position et en temps (sortie 1 PPS).

Photo d'illustration.

INTRODUCTION

Navstar, devenu aujourd'hui GPS, est un système de géolocalisation basé sur la trilatération de signaux émis par une constellation de satellites. Conçu à des fins militaires, le segment civil n'est aucunement protégé contre les attaques. Cependant, ces attaques nécessitaient jusqu'à récemment un équipement peu accessible au commun des mortels. La situation change rapidement avec la disponibilité d'émetteurs programmables par radio logicielle.

Depuis la désactivation de son mode de résolution dégradée SA (*Selective Availability*) en mai 2000 **[1] [2]**, GPS s'est peu à peu insinué dans nombre d'activités quotidiennes, pour devenir omniprésent, ne serait-ce que par notre obsession à consulter les informations géoréférencées de notre téléphone mobile. Une étude anglaise estime **[3]** à 5 milliards de livres les pertes associées au brouillage de GPS pendant 5 jours, une tâche triviale et sans intérêt technique, mais qui met en évidence l'omniprésence des systèmes de navigation par satellite dans les infrastructures critiques d'un pays (penser navigation aérienne, synchronisation d'horloges et de transports ferroviaires, livraisons de colis...). Bien plus grave, nous nous proposons d'exposer ici le leurrage (*spoofing*) de GPS **[4] [5]** : alors que brouiller nécessite un bête émetteur un peu puissant et se détecte immédiatement par une perte de service, le leurrage est plus subtil, car il introduit une information erronée pour un utilisateur qui croit avoir une information valable, et donc ne se rend pas compte de l'attaque **[6] [7]**.

Notre objectif est dans un premier temps de résumer les grandes lignes du fonctionnement de GPS **[8]** : nous insisterons sur le fait que le positionnement nécessite avant tout un transfert précis de temps pour permettre une trilatération. Nous démontrerons ensuite l'attaque sur divers récepteurs allant des téléphones mobiles aux récepteurs GPS grand public tels que U-Blox (qui équipent par exemple les drones DJI – nous laissons le lecteur imaginer la portée de l'attaque). Quelques contre-mesures triviales limitent la portée de l'attaque, mais ne l'interdisent pas, et nous verrons que même les GPS intégrés dans des véhicules sont leurrés sous réserve de prendre un peu soin à la qualité du signal émis. Nous conclurons avec quelques stratégies de contre-mesure envisagées.

PRINCIPE DE GPS

GPS, comme les autres systèmes de navigation par satellite (GLONASS russe, Galileo européen – de façon générale GNSS pour *Global Navigation Satellite Systems*), est formé d'une constellation de satellites en orbite à une vingtaine de milliers de kilomètres au-dessus de la surface de la Terre. La mécanique céleste – les lois de Kepler – impose un certain nombre de conditions sur les propriétés orbitales qui seront au cœur de notre capacité à contrer les attaques de leurrage si nous nous en donnons les moyens. En particulier, un premier paramètre que nous introduisons dès le début de cette discussion est le décalage Doppler introduit

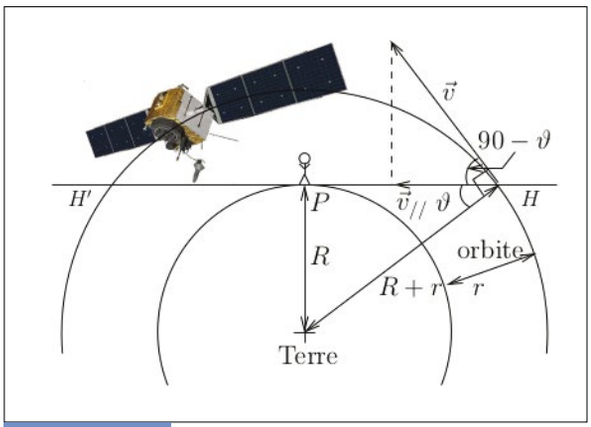


FIGURE 1 *Schéma de l'orbite d'un satellite et notations utilisées dans le texte.*

par le mouvement des satellites. En nous inspirant des notations de la fig. 1, nous constatons que lorsque le satellite apparaît au-dessus de l'horizon HH', l'angle ϑ est donné par $\sin(\vartheta) = R / (R+r)$ avec $R = 6400$ km le rayon de la Terre et $r = 20000$ km l'altitude du satellite sur son orbite. De ce fait, la projection du vecteur tangentiel de la vitesse v est $v_{\parallel} = |\vec{v}|\sin(\vartheta) = v \cdot R / (R+r)$. Compte tenu de la troisième loi de Kepler qui nous dit que le ratio du carré de la période au cube du rayon de l'orbite est constant, et sachant que les satellites en orbite géostationnaire, donc de période de 24 h, sont à une altitude de 36000 km, nous déduisons la période d'un satellite GPS de $T = 12$ h. Compte tenu de cette période et de la distance parcourue le long de l'orbite, nous déduisons une vitesse tangentielle de $2\pi(R+r) / T \approx 13800$ km/h = 3840 m/s. Nous en déduisons la vitesse radiale maximale lorsque le satellite est en H ou H' de $|\vec{v}| = 3840 \times 6400 / 26400 = 930$ m/s et donc un décalage Doppler δf maximal de $\delta f = f_0 \cdot v/c$ avec $f_0 = 1575{,}42$ MHz la fréquence de la porteuse et $c = 3 \cdot 10^8$ m/s la célérité de la lumière : $|\delta f| < 4.9$ kHz. Cette limite sur le décalage Doppler est imposée par la physique céleste et ne peut en aucun cas être enfreinte : nous verrons qu'elle nous amène une première protection contre le leurrage des signaux GPS.

Le signal de la porteuse porte une information doublement codée : d'une part un message rapide (1 Mb/s) encode le numéro de satellite émettant l'information, et d'autre part le message de navigation transmis par chaque satellite à bas débit (50 bits/s) est superposé sur ce code. Nous avons détaillé ces divers encodages dans **[8]**, dans lequel nous nous étions arrêté à l'obtention des bits du message de navigation, sans en étudier le contenu. Toute la difficulté de leurrer GPS est de minutieusement reconstruire chaque trame

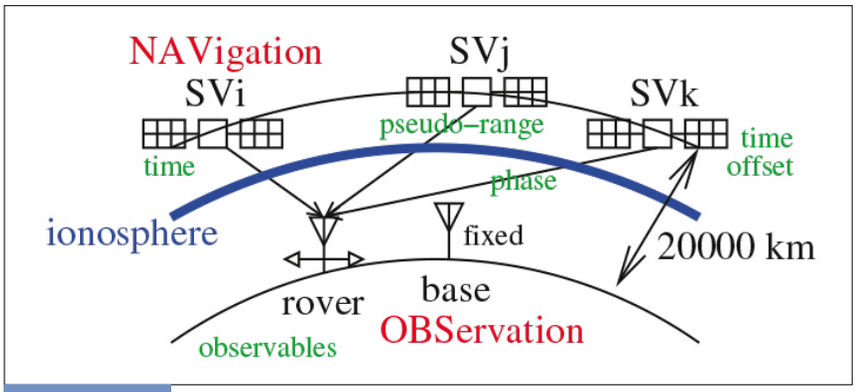
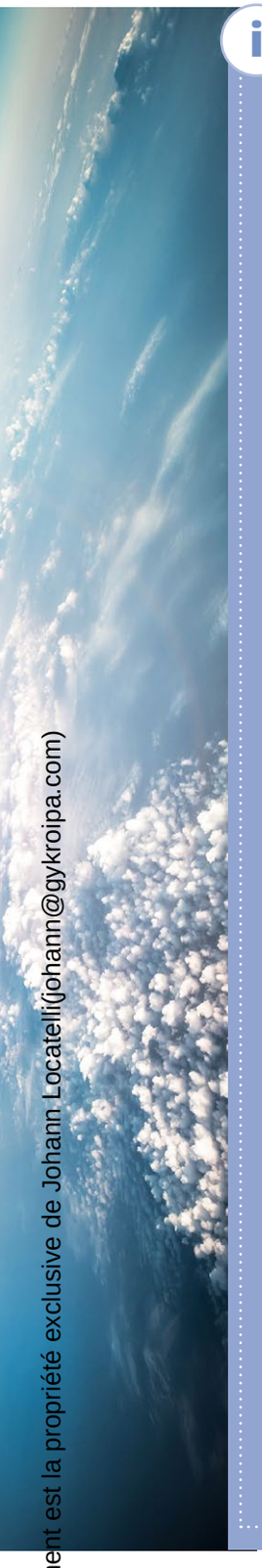


FIGURE 2 *Segment spatial de GPS avec les éphémérides des véhicules spatiaux (SV), distingués par leur code pseudo-aléatoire, décrites par les fichiers de navigation, et segment au sol décrit par les fichiers d'observation RINEX.*

Photo d'illustration.



NOTE

[10] définit un pseudo-range comme « *The pseudo-range (PR) is the distance from the receiver antenna to the satellite antenna including receiver and satellite clock offsets (and other biases, such as atmospheric delays): PR=distance+c*(receiver clock offset–satellite clock offset + other biases)* ». Il s'agit donc d'une estimation brute de la distance récepteur-véhicule spatial, indépendamment de toute correction de délai de propagation de l'onde dans les diverses couches atmosphériques.

Un exemple de mesure, extrait d'un fichier RINEX généré à partir d'un récepteur U-Blox monofréquence (L1) avec mesure de phase, est :

```
> 2017 12 22  5 57 46.0010000  0 12
G12  22028410.605   115760077.968        3307.683            46.000
G18  21024975.970   110486988.127        1088.977            45.000
G24  20360955.102   106997530.988        -437.104            49.000
J 1  37731461.503   198280150.949         713.158            45.000
J 2  37863385.498   198973438.883        -372.958            45.000
G15  21655567.700   113800790.795       -1526.538            48.000
G20  22301572.946   117195549.217       -2991.357            44.000
R16  21729491.824   116075061.937        3857.002            41.000
R15  19336042.668   103325965.774        -387.583            43.000
R 4  20461570.837   109570805.433       -1945.881            44.000
R14  21528858.057   114761049.343       -3182.688            38.000
R 5  19671117.697   105153436.303        1676.938            38.000
```

La première lettre indique la constellation, avec G pour GPS, R pour le GLONASS russe, et J pour les satellites QZSS japonais en orbite géosynchrone, entre 32000 et 38000 km.

Un rapide survol de la deuxième colonne de ces mesures nous conforte sur leur validité : la constellation des satellites GPS (véhicules spatiaux dont le nom commence par « G ») orbite à 20000 km au-dessus de la Terre. Les pseudo-ranges sont donc compris entre une vingtaine de milliers de km, et cette altitude ajoutée au diamètre terrestre de 2 x 6400 km (évidemment, un satellite GPS aux antipodes de l'observateur n'est pas visible, mais c'est un pire cas). Ici, les distances aux satellites sont comprises entre 20000 km et 22000 km pour GPS, un peu moins pour GLONASS, en accord avec nos attentes. Les Japonais (ces mesures ont été acquises depuis Sendai, au Japon) proposent un système de localisation basé sur des orbites géosynchrones avec des satellites à des altitudes plus élevées : ici encore les mesures sont en accord avec nos attentes, puisque nous observons 37800 km. Aucun satellite européen Galileo (nom commençant par « E ») n'est visible dans cette acquisition. Dans la 4ème colonne, les décalages Doppler sont eux aussi dans la gamme des valeurs décrites dans le texte. La 5ème colonne indique la puissance du signal, et la 3ème colonne une information de phase de la porteuse plus compliquée à analyser.

La conversion des pseudo-ranges décrits dans un fichier RINEX vers une information de datation ou de position est prise en charge par l'excellente bibliothèque d'outils libres `rtklib` (www.rtklib.com), dont l'utilisation dépasse le cadre de cet article.

en respectant les paramètres physiques de la transmission pour faire croire aux récepteurs les plus pointilleux que le signal est émis de l'espace. Les diverses trames du message de navigation sont décrites en détail dans **[9]** : ces quelques pages n'ont évidemment pas la prétention de reprendre les plus de 600 pages de ces deux ouvrages dont la compréhension est fondamentale. En particulier, ces documents expliquent comment passer des paramètres orbitaux des satellites (transmis dans les messages de navigation) et de la date des transmissions vers les *pseudo-ranges* tenant compte de la position du récepteur au sol. Le *pseudo-range* est l'élément clé du positionnement de l'utilisateur sur Terre, et l'information brute traitée par le récepteur au sol pour le positionner par trilatération. Ces *pseudo-ranges* sont en particulier transmis comme donnée brute dans les fichiers RINEX (*Receiver Independant EXchange Format*), format standardisé **[10]** pour échanger les informations entre récepteurs GNSS (fig. 2, page 90).

Il est important de maîtriser ce concept de fichier RINEX, car c'est grâce à ces fichiers de référence qu'un utilisateur peut améliorer, en post-traitement, l'estimation de la position de son récepteur en intégrant un certain nombre de corrections telles que le délai ionosphérique – retard de l'onde électromagnétique introduit par la densité variable d'électrons dans l'ionosphère. Pour cette raison, les utilisateurs de récepteurs d'un peu plus haut de gamme que les récepteurs grand public qui ne fournissent que les informations traitées au format NMEA (trop tard pour retraiter les données et en améliorer la résolution) peuvent télécharger les éphémérides de précision améliorée des satellites (observation des paramètres orbitaux au lieu de leur prévision) ainsi que diverses corrections, et ce grâce aux services de l'IGS (*International GNSS Service*) qui collecte les mesures précises de stations de référence distribuées à la surface de la Terre. Les deux types de fichiers RINEX sont les observations (extension finissant par o) issues du récepteur au sol qui permettent de corriger les observations faites par un utilisateur sur le terrain – ces fichiers ne nous intéressent pas ici – et les fichiers de navigation (finissant par n) qui donnent les paramètres orbitaux des satellites, indépendamment de toute notion de localisation au sol (fig. 2). Ce second jeu de données, décrivant les paramètres orbitaux des satellites de la constellation et ici acquis en traitant les messages de navigation transmis par les satellites, sera aussi disponible en version améliorée en précision sur divers sites chargés de disséminer les produits de l'IGS et répertoriés à **https://kb.igs.org/hc/en-us/articles/202054393-IGS-FTP-Sites** – par exemple **ftp://cddis.gsfc.nasa.gov/gnss/products/** – pour fournir les informations d'entrée pour générer les signaux de leurrage du GPS.

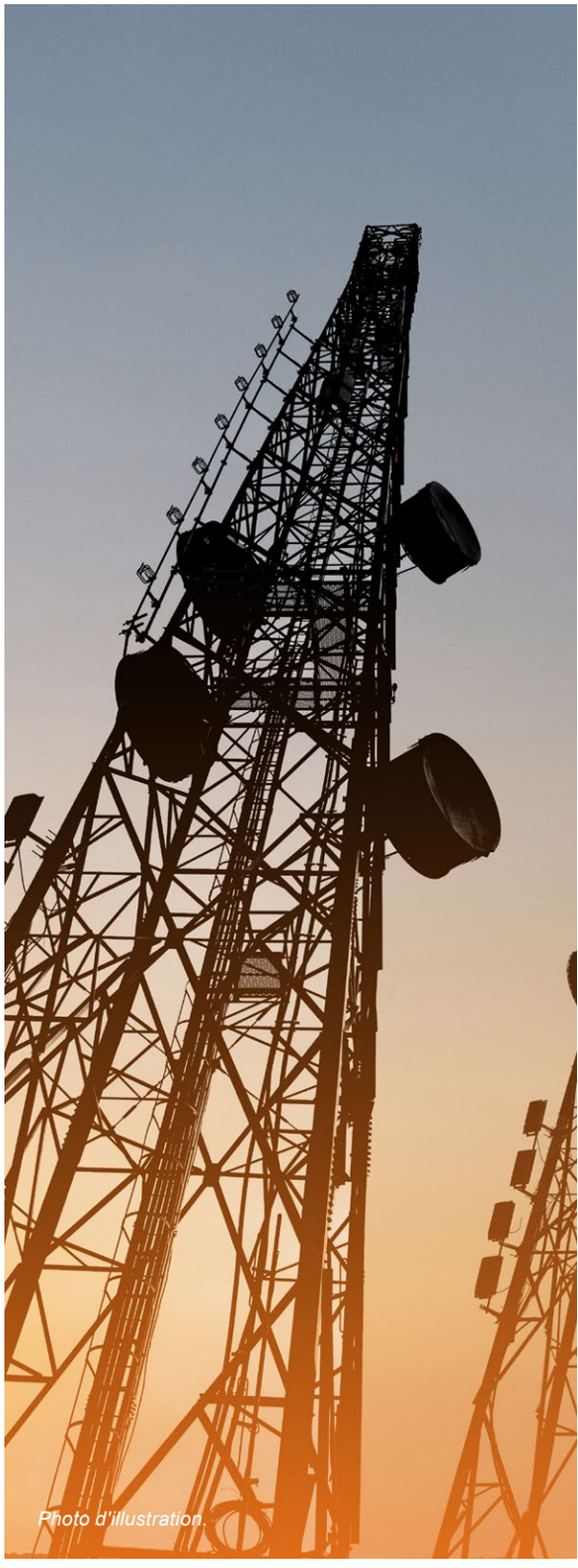
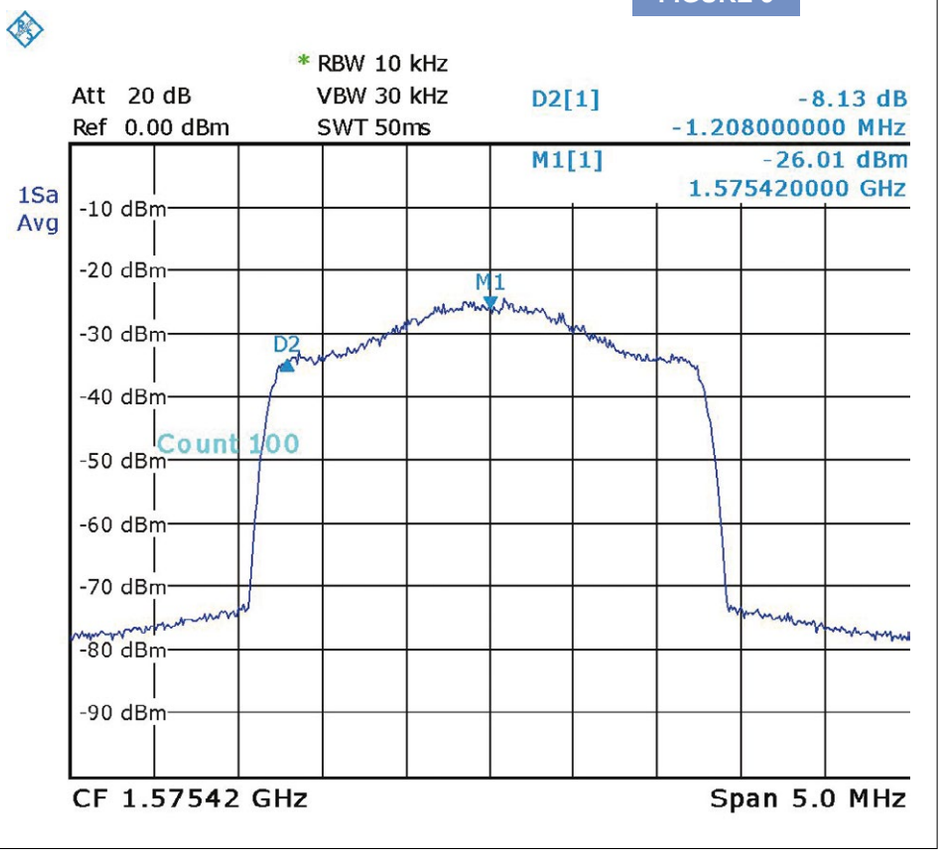


Photo d'illustration.

FIGURE 3



Spectre du signal émis par la PlutoSDR réglée sur un gain de 0 dB : la porteuse à 1575,42 MHz est étalée spectralement sur ± 1MHz par modulation en phase selon la séquence pseudo-aléatoire caractéristique de chaque satellite, induisant un niveau de -30 dBm environ dans la bande.

Un second paramètre imposé par la physique de la constellation spatiale d'émetteurs radiofréquences est la puissance reçue au sol, déterminée par le bilan de liaison et en particulier les pertes de propagation imposées par la conservation de l'énergie – encore une fois un principe physique que nous ne saurions contourner lors de nos tentatives de leurrage. La norme décrivant GPS n'explicite pas la puissance émise depuis l'espace, mais la puissance reçue au sol : **[11, p.14]** nous informe que GPS doit fournir au sol une puissance du signal de -160 dBW = -130 dBm sur la porteuse L1 à 1575,42 MHz. Alors que nous avions décrit dans **[8]** comment ce signal se trouve sous le bruit thermique et ne peut donc être visible sur un analyseur de spectre en l'absence d'une antenne de fort gain telle qu'un radiotélescope, ce signal est remonté de 30 dB lors de la compression d'impulsion réalisée par la corrélation du signal acquis avec le code (connu) de chaque satellite. Le premier point important de cette analyse est que le niveau de signal reste excessivement faible au niveau du récepteur et tout émetteur au sol pourra très facilement dépasser cette puissance pour éblouir le récepteur. Au contraire, nous verrons que certains récepteurs vérifient le niveau des signaux reçus pour rejeter ceux présentant une puissance excessive et qui ne sauraient donc venir de l'espace.

GPS exploite un signal de 2 MHz de bande passante, donc toute attaque de leurrage nécessitera une source d'une telle bande passante. Nous utilisons la PlutoSDR de Analog Devices, disponible pour 85 euros chez Mouser (depuis qu'elle est épuisée chez Farnell). Ce circuit est capable d'émettre jusqu'à 0 dBm (1 mW) et d'atténuer sa sortie pour abaisser cette puissance.

Ainsi, à titre de comparaison, notre émetteur servant à l'attaque peut émettre jusqu'à 1 mW (nous vérifions que l'atténuation 0 dB signifie une puissance émise de 0 dBm en mesurant le niveau de signal d'une porteuse émise continûment), que nous observons atteindre une puissance de -30 dBm après étalement du spectre par modulation de phase (fig. 3). Cette observation est en accord avec l'étalement de spectre sur 1023 bits qui abaisse la puissance crête de 10 $\log_{10}$ (1023) = 30 dB. Ceci est valable pour les divers satellites

de la constellation, dont les signaux se somment après propagation. Les pertes de propagation en espace libre (*Free Space Propagation Loss* – FSPL) sont de FSPL = 20 $\log_{10}$ (d) + 20 $\log_{10}$ (f) - 147,55, avec la constante à la fin de cette équation donnée par 20$\log_{10}$ (c/4/π) avec c=3 • 10^8 m/s la célérité de l'onde électromagnétique dans le vide. À f=1575,42 MHz, ces pertes s'élèvent à 20 $\log_{10}$ (d) + 36 dB.
Si nous émettons 0 dBm, alors les pertes nécessaires à atteindre les -130 dBm de la norme sont FSPL = 130 = 20$\log_{10}$ (d) + 36 dB, qui seraient atteintes pour une distance d = $10^{(130-36)/20}$ = 50 km. En pratique, nous émettons 20 dB de moins (option -A -20 du logiciel de synthèse des trames GPS que nous décrirons ci-dessous), soit une portée de l'attaque de l'ordre de 5 km. En ne prenant pas la norme, mais le bilan de liaison en espace libre entre le satellite qui émet 25 W (**http://gpsinformation.net/main/gpspower.htm**) avec un gain d'antenne de 13 dBi et les 182 dB de pertes de propagation le long des 20000 km qui séparent le satellite de la surface de la Terre, la puissance au sol est -125 dBm. Si nous désirons avoir au moins 3 dB de puissance de plus que le « vrai » signal, alors les 8 dB de différence avec le calcul précédent réduisent la portée de notre attaque à 5 km x $10^{(-8/20)}$ = 2 km, garantissant le peu d'impact sur l'environnement de travail de nos tests : nous avons vérifié que, probablement compte tenu de la médiocrité de l'antenne dipôle attaquée en sortie de la PlutoSDR sans balun **[A]**, le signal GPS dépassait notre émission à une cinquantaine de mètres de l'émetteur.

LOGICIEL POUR DÉPLOYER L'ATTAQUE DE LEURRAGE

Ayant sélectionné la plateforme matérielle respectant les contraintes de fréquence de porteuse (1575,42 MHz), de bande passante (2 MHz) et de puissance émise, il nous reste à rédiger le logiciel de synthèse des signaux. Le travail n'est pas complexe, mais nécessite un soin particulier pour implémenter toutes les étapes : nous nous appuyons sur **github.com/Mictronics/pluto-gps-sim** pour démontrer l'attaque. Ce logiciel est impressionnant de concision puisqu'il implémente toute la séquence, de la lecture du fichier de paramètres orbitaux RINEX à la génération des messages de navigation en passant par toutes les transformations de coordonnées imposées par la mécanique céleste, dans un millier de lignes de code parfaitement lisibles (et donc modifiables pour injecter nos propres paramètres dans les messages transmis).

L'objectif de l'attaque par leurrage est de générer des signaux représentatifs de ceux émis par la constellation de satellites. Étant donné que tous les satellites communiquent sur la même fréquence de porteuse de 1575,42 MHz, le seul travail est de générer le flux de données complexes I/Q, somme des contributions des divers satellites, avec la modulation en phase du code de chaque satellite décalé en fréquence par l'effet Doppler associé à la position du satellite dans le ciel, et les messages de navigation permettant de positionner le récepteur sur Terre avec un retard introduit par la propagation du signal du satellite au sol tel que représenté par chaque pseudo-range. Afin de ne pas être perturbés par les vrais satellites de la constellation qui émettent en continu, nous devons absolument générer un signal de leurrage correspondant aux satellites visibles en un instant et lieu donné : faute de respecter cette contrainte, le récepteur recevra un mélange de « vrais » signaux et de « faux » signaux et ses chances de se fixer sur la position erronée sont réduites. Nous avons vu qu'à 20000 km d'altitude, les satellites mettent 12 h pour effectuer une orbite, donc exploiter la configuration valable quelques heures avant l'attaque reste pertinent. Le lieu introduit lors de l'attaque ne doit par ailleurs pas trop différer du site physique du récepteur pour que ce dernier voie une constellation similaire à celle des messages émis.


Photo d'illustration.

La liste des satellites et leurs paramètres orbitaux tels qu'émis dans les messages de navigation des divers satellites sont publiés à **cddis.nasa.gov/Data_and_Derived_Products/GNSS/hourly_30second_data.html** avec une résolution horaire : ce service est utile en pratique pour la correction en post-traitement de signaux GPS acquis avec un récepteur unique (correction du délai ionosphérique notamment en l'absence de base de référence sur site), tel que nous l'avons décrit auparavant en mentionnant IGS.

Le jour courant de la date GPS s'obtient à **sopac.ucsd.edu/convertDate.shtml** : par exemple, le 30 juillet 2018 est le jour 211 de l'année, donc les éphémérides s'obtiennent à **ftp://cddis.gsfc.nasa.gov/gnss/data/hourly/2018/211/**. Le choix de l'heure tiendra évidemment compte du décalage entre heure locale et temps universel, soit 1 ou 2 h en France. **cddis.nasa.gov/Data_and_Derived_Products/GNSS/broadcast_ephemeris_data.html#GPShourly** nous informe que ce sont les fichiers finissant par n qui nous intéressent (*broadcast ephemeris*) pour connaître les paramètres orbitaux des véhicules spatiaux (SV) de la constellation : nous sélectionnons donc le fichier nommé **hour2110.18n.Z** (format **hourDDD0.YYn.Z** avec le jour **DDD** et l'année **YY**) :

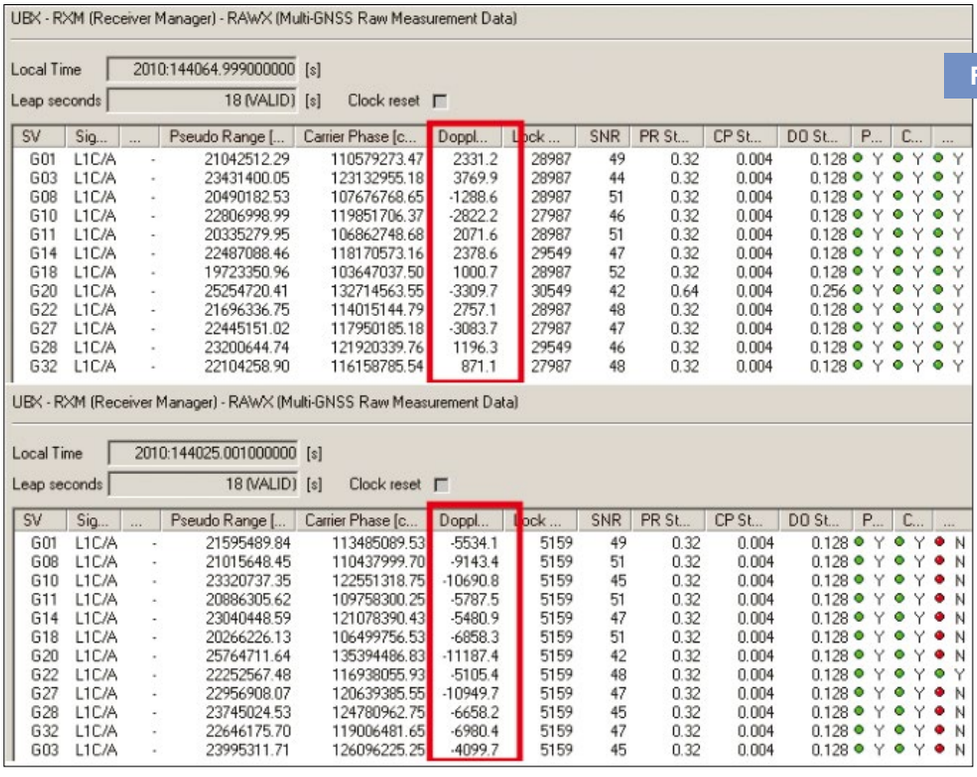
```
./pluto-gps-sim -e hour2110.18n -A -20.0 -t 2018/07/30,10:00:00 -l
48.3621221,-4.8223307,100
Using static location mode.
Gain: -20.0dB
RINEX date = 30-JUL-18 23:30
Start time = 2018/07/30,10:00:00 (2012:122400)
PRN   Az    El     Range     Iono
04  110.0  80.4  20159594.4   1.7
05   33.5   8.9  24730267.9   4.4
09  320.7   5.1  25212521.1   4.5
16  302.7  51.7  21108967.6   2.0
20  144.1   7.2  25065494.6   6.2
21  133.7  64.8  21075459.6   1.9
23  292.6   5.3  25170257.9   4.5
25  120.9   6.6  25194957.8   6.4
26  292.6  82.7  20252112.2   1.7
27  256.8  22.9  23261110.3   3.3
29   64.7  31.3  22678543.7   3.1
31  193.6  33.0  22775272.7   3.0
```

L'outil original, **gps-sdr-sim** dont **pluto-gps-sim** est issu, propose en plus du mode statique un mode dynamique, qui nécessite cependant de sauver un fichier volumineux de coefficients I/Q (2,5 MS/s) précalculés avant exécution, limitant la durée de l'attaque à quelques minutes tout au plus. Ce fichier est généré à partir d'un trajet défini au format NMEA.

DÉMONSTRATION : TÉLÉPHONE MOBILE ET RÉCEPTEUR U-BLOX

La première démonstration de l'efficacité de l'attaque porte sur les téléphones mobiles, outil de géolocalisation le plus couramment utilisé par le grand public actuellement. La figure 4 démontre le résultat de l'attaque sur 3 téléphones : un des téléphones a conservé les coordonnées du site acquises en exploitant les signaux de la constellation GPS (Besançon à 47°N, 6°E), les deux autres se sont fait leurrer par une position erronée choisie arbitrairement au sud de la France à 42,5°N, 2,3°E. Précisons que pour obtenir ce résultat, nous avons désactivé toute assistance de localisation telle que GSM ou WiFi : cette contrainte n'est pas limitante, car le brouillage est excessivement simple à mettre en œuvre par rapport à la complexité du leurrage, et éliminer ces assistances à la localisation ne pose pas de problème technique.

FIGURE 4 *Trois téléphones mobiles sont soumis au signal de leurrage émis par la PlutoSDR : un téléphone Samsung (milieu) et un téléphone Sony (droite) se croient dans le sud de la France, tandis que le Samsung de gauche est resté à Besançon.*



UBX - RXM (Receiver Manager) - RAWX (Multi-GNSS Raw Measurement Data)

Local Time 2010:144064.999000000 [s]
Leap seconds 18 (VALID) [s] Clock reset

SV	Sig...	...	Pseudo Range [...	Carrier Phase [c...	Doppl...	Lock ...	SNR	PR St...	CP St...	DO St...	P...	C...	...
G01	L1C/A	-	21042512.29	110579273.47	2331.2	28987	49	0.32	0.004	0.128	Y	Y	Y
G03	L1C/A	-	23431400.05	123132955.18	3769.9	28987	44	0.32	0.004	0.128	Y	Y	Y
G08	L1C/A	-	20490182.53	107676768.65	-1288.6	28987	51	0.32	0.004	0.128	Y	Y	Y
G10	L1C/A	-	22806998.99	119851706.37	-2822.2	27987	46	0.32	0.004	0.128	Y	Y	Y
G11	L1C/A	-	20335279.95	106862748.68	2071.6	28987	51	0.32	0.004	0.128	Y	Y	Y
G14	L1C/A	-	22487088.46	118170573.16	2378.6	29549	47	0.32	0.004	0.128	Y	Y	Y
G18	L1C/A	-	19723350.96	103647037.50	1000.7	28987	52	0.32	0.004	0.128	Y	Y	Y
G20	L1C/A	-	25254720.41	132714563.55	-3309.7	30549	42	0.64	0.004	0.256	Y	Y	Y
G22	L1C/A	-	21696336.75	114015144.79	2757.1	28987	48	0.32	0.004	0.128	Y	Y	Y
G27	L1C/A	-	22445151.02	117950185.18	-3083.7	27987	47	0.32	0.004	0.128	Y	Y	Y
G28	L1C/A	-	23200644.74	121920339.76	1196.3	29549	46	0.32	0.004	0.128	Y	Y	Y
G32	L1C/A	-	22104258.90	116158785.54	871.1	27987	48	0.32	0.004	0.128	Y	Y	Y

UBX - RXM (Receiver Manager) - RAWX (Multi-GNSS Raw Measurement Data)

Local Time 2010:144025.001000000 [s]
Leap seconds 18 (VALID) [s] Clock reset

SV	Sig...	...	Pseudo Range [...	Carrier Phase [c...	Doppl...	Lock ...	SNR	PR St...	CP St...	DO St...	P...	C...	...
G01	L1C/A	-	21595489.84	113485089.53	-5534.1	5159	49	0.32	0.004	0.128	Y	Y	N
G08	L1C/A	-	21015648.45	110437999.70	-9143.4	5159	51	0.32	0.004	0.128	Y	Y	N
G10	L1C/A	-	23320737.35	122551318.75	-10690.8	5159	45	0.32	0.004	0.128	Y	Y	N
G11	L1C/A	-	20886305.62	109758300.25	-5787.5	5159	51	0.32	0.004	0.128	Y	Y	N
G14	L1C/A	-	23040448.59	121078390.43	-5480.9	5159	47	0.32	0.004	0.128	Y	Y	N
G18	L1C/A	-	20266226.13	106499756.53	-6858.3	5159	51	0.32	0.004	0.128	Y	Y	N
G20	L1C/A	-	25764711.64	135394486.83	-11187.4	5159	42	0.32	0.004	0.128	Y	Y	N
G22	L1C/A	-	22252567.48	116938055.93	-5105.4	5159	48	0.32	0.004	0.128	Y	Y	Y
G27	L1C/A	-	22956908.07	120639385.55	-10949.7	5159	47	0.32	0.004	0.128	Y	Y	N
G28	L1C/A	-	23745024.53	124780962.75	-6658.2	5159	45	0.32	0.004	0.128	Y	Y	N
G32	L1C/A	-	22646175.70	119006481.65	-6980.4	5159	47	0.32	0.004	0.128	Y	Y	N
G03	L1C/A	-	23995311.71	126096225.25	-4099.7	5159	45	0.32	0.004	0.128	Y	Y	N

FIGURE 5

Impact de l'oscillateur local du synthétiseur de signaux sur le décalage Doppler observé par le récepteur U-Blox. Ici, un synthétiseur de fréquence asservi sur un maser à hydrogène cadence la PlutoSDR à la fréquence nominale de 40 MHz (haut) ou à 40 Mhz-200 Hz, soit un décalage de 5 ppm. Nous constatons que dans le premier cas les décalages Doppler sont bien dans l'intervalle autorisé par la mécanique céleste (±5 kHz), alors que dans le second cas les décalages de fréquence sont aberrants.

La même attaque est effectuée sur des récepteurs U-Blox de modèles Neo7M ou NeoM8T avec succès. Ces récepteurs sont intéressants, car en plus d'être utilisés sur de nombreux drones dont ceux commercialisés par DJI, ils fournissent les informations brutes (pseudo-ranges) permettant une analyse détaillée des signaux acquis, avant traitement pour extraire la localisation du récepteur. De ce fait, l'outil d'analyse des trames U-Blox Center fournit de nombreuses informations sur la nature des signaux reçus, dont des caractéristiques d'*anti-spoofing* et *anti-jamming*. Un premier critère de puissance rejette les signaux excessivement puissants qui ne pourraient venir d'un satellite **[12]**.

La figure 5 démontre l'impact d'une variation contrôlée de la fréquence d'horloge cadençant l'émetteur sur le récepteur. Alors que nous décalons l'horloge de l'émetteur de 5 ppm (200 Hz par rapport à la valeur nominale de 40 MHz), le récepteur continue à fournir des informations malgré la détection de dysfonctionnements tel qu'indiqué dans les colonnes de droite nommées PR (*Pseudo-Range*), CP (*Carrier Phase*) et DO (*Doppler Measurement*) : le récepteur U-Blox a bien détecté des valeurs incohérentes du décalage Doppler (DO rouge), mais cela ne l'empêche pas, dans sa configuration par défaut, de transmettre une position erronée.

Une tentative de leurrage similaire à celle démontrée sur téléphone mobile échoue sur le GPS de voitures. Nous attribuons notre échec à leurrer un véhicule à l'utilisation de tels indicateurs d'incohérence du signal reçu, à savoir puissance excessive et irréaliste pour des satellites en orbite, et décalage Doppler incohérent. Le premier point sera résolu par un ajustement de la puissance émise, le second par l'utilisation d'une source de fréquence plus stable que celle fournie d'origine avec la PlutoSDR.

DÉMONSTRATION : GPS DE VOITURE

Cette première expérience de leurrage échoue avec certains modèles de téléphone mobile, mais surtout échoue avec les GPS des voitures. Nous attribuons cet échec à l'écart de l'oscillateur local à la PlutoSDR à sa valeur nominale : même si le Rakon RXO3225M présente d'excellentes performances pour un oscillateur basé sur un résonateur compensé en température, il reste un écart de ±25 ppm à la valeur nominale qu'une « vraie » source GPS ne saurait jamais souffrir. Une horloge rubidium telle que celle équipant les véhicules spatiaux présente au moins une stabilité de quelques ppb au pire, soit au moins mille fois meilleures que cet oscillateur à quartz.

Notre première solution à l'incertitude sur la fréquence de l'oscillateur local consiste à exploiter un synthétiseur générant un signal à 40 MHz asservi sur un maser à hydrogène (voir l'article « Matériel pour la radio logicielle », publié dans *GNU/Linux Magazine n°224*) connu pour être exact. Nous penserons, au cours de cette expérience, à retirer le coefficient d'étalonnage introduit par Analog Devices dans le logiciel contrôlant la PlutoSDR. Ceci peut se faire en se logguant sur la carte avec un émulateur de terminal ou en ssh (login *root*, mot de passe *analog*), puis en exécutant :

```
echo 40000000 > /sys/bus/iio/devices/iio:device1/xo_correction
```

pour annoncer que la fréquence cadençant le circuit est exactement à 40 MHz. Cependant, cette nouvelle définition de la fréquence de l'oscillateur local n'est mémorisée que jusqu'au prochain redémarrage de la carte. Une solution pérenne consiste à définir une nouvelle variable d'environnement de U-Boot non-volatile.

```
fw_setenv xo_correction 40000000
```

FIGURE 6

Haut, droite : montage dans lequel l'oscillateur cadençant la PlutoSDR est remplacé soit par la sortie d'un synthétiseur de fréquence référencé sur un maser à hydrogène (ici inutilisé), soit par un quartz contrôlé en température (OCXO). Bas : deux véhicules – Renault (gauche) et Mercedes (bas, droite) – situés sur le parking de l'ENSMM se croient les roues dans l'eau au large de Brest.

Photo d'illustration.

NOTE

Chaque Pluto est cadencée par un oscillateur à 40 MHz dont la fréquence exacte est calibrée et renseignée dans une zone mémoire non accessible trivialement par l'utilisateur. Dans notre cas, nous désirons expliquer à la PlutoSDR qu'elle sera désormais cadencée par un quartz stable à 10 MHz. Avant de lancer le noyau Linux, U-Boot modifie la valeur par défaut de l'horloge dans le devicetree chargé en mémoire pour appliquer la valeur de calibration. Pour ce faire, U-Boot fait appel au script `adi_loadvals` qui exécute :

```
fdt set /clocks/clock@0 clock-frequency ${ad936x_ext_refclk}
```

Le contenu de la variable `ad936x_ext_refclk` est obtenu par la lecture de la zone dédiée à la calibration et sa valeur est donc écrasée juste avant l'appel à `adi_loadvals`, rendant ainsi impossible à l'utilisateur de la surcharger pour la remplacer par sa propre valeur.

Pour contourner cette limitation et tel que présenté à ez.analog.com/university-program/f/q-a/77922/will-it-be-possible-to-feed-in-a-reference-clock-to-the-adalm-pluto/295481#295481, la solution consiste à modifier le script pour ajouter une nouvelle variable, qui, si elle est présente, sera utilisée à la place de `ad936x_ext_refclk`. Concrètement, le script original de github.com/analogdevicesinc/u-boot-xlnx/blob/pluto/include/configs/zynq-common.h#L271 devient :

```
 if test ! -n $"{ad936x_skip_ext_refclk}"; then if test -n
$"{ad936x_custom_refclk}"; then fdt set /clocks/clock0 clock-
frequency $"{ad936x_custom_refclk}"; elif test -n $"{ad936x_ext_
refclk}"; then fdt set /clocks/clock0 clock-frequency $"{ad936x_
ext_refclk}"; fi; fi;
```

Il devient ensuite possible de définir la variable `ad936x_custom_refclk` avec la valeur choisie :

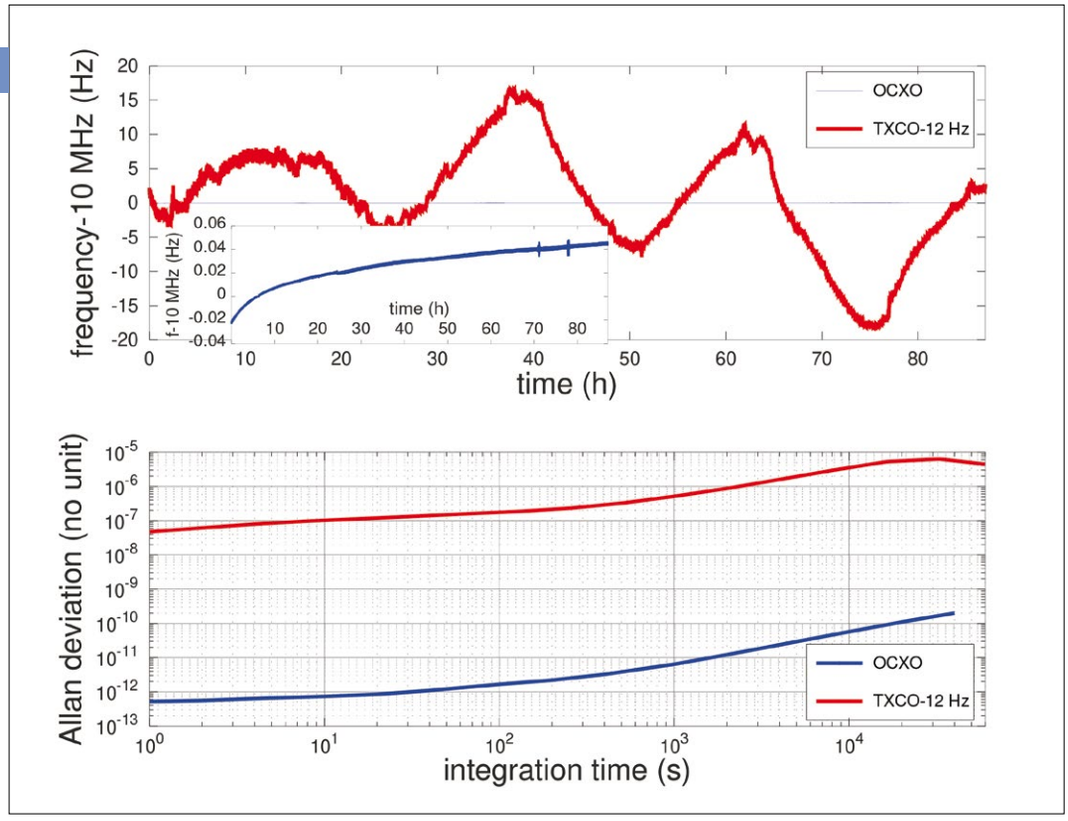
```
fw_setenv ad936x_custom_refclk "<10000000>"
```

Une fois cette modification matérielle effectuée, et en suivant la procédure de la section précédente, les véhicules sont eux aussi rapidement leurrés, pour amener des voitures garées à Besançon sur le parking de l'École Nationale Supérieure de Mécanique et Microtechniques (ENSMM) à croire qu'ils ont les roues dans l'eau au large de Brest (fig. 6). Notre hypothèse est donc la bonne, l'exactitude de l'horloge de la source est la cause du dysfonctionnement de l'attaque sur les GPS de véhicules.

Rares sont cependant les lecteurs ayant accès à un maser à hydrogène, qui de toute façon ne peut être déplacé pour être amené sur le site de l'attaque. Nous pallions donc cette déficience en remplaçant le maser par un oscillateur à quartz de bonne qualité. Alors que les oscillateurs asservis sur des résonateurs compensés en température (*Temperature Controlled Crystal Oscillator* – TCXO) présentent des fluctuations de fréquences de quelques dizaines de ppm avec leur environnement, un oscillateur asservi en température (*Oven Controlled Crystal Oscillator* – OCXO) présente des fluctuations inférieures au ppm. Nous avons récupéré dans un compteur de fréquence (*electronic counter*) Hewlett Packard 5345A défectueux un excellent OCXO HP10811 **[B]**. Cet oscillateur présente une fluctuation relative de fréquence de $5 \cdot 10^{-13}$ à la seconde pour monter à $5 \cdot 10^{-12}$ à 100 secondes et dériver à long terme (fig. 7). Le quartz a été ajusté à mieux que 30 mHz de la fréquence nominale par comparaison avec le maser à hydrogène. Ici encore, en commandant un synthétiseur de fréquence avec cette source, l'attaque sur les véhicules se conclut sans problème, cette fois avec un montage ne nécessitant qu'une centaine de mA sous 24 V pour le chauffage et quelque mA sous 12 V pour l'oscillateur. La PlutoSDR n'a plus qu'à être configurée pour accepter une source à 10 MHz au lieu des 40 MHz nominaux pour s'affranchir du synthétiseur (voir encadré).

FIGURE 7

Haut : évolution au cours du temps de la fréquence du TCXO Rakon initialement fournie avec la PlutoSDR (rouge) et d'un OCXO HP10811. L'insert présente un zoom sur la mesure de l'OCXO sur sa propre échelle. Bas : variance d'Allan sur ce même jeu de données, illustrant le gain en stabilité de 5 ordres de grandeur par le passage du TCXO à l'OCXO. Toutes les mesures sont référencées à un maser à hydrogène : le TCXO est mesuré au moyen d'un compteur de fréquence Agilent 53132A, l'OCXO est caractérisé par un banc Symmetricom TSC5110A.



DÉCALER LE TEMPS

Une application classique de GPS pour le transfert de temps exploite le signal 1 PPS – 1 Pulse Par Seconde – qui représente un signal de synchronisation précis pour asservir des horloges sur la base de temps commune propagée par la constellation de satellites **[13]**.

NOTE

Alors que le transfert de fréquence est un concept relativement abstrait pour le commun des mortels (penser réseau informatique et les Gb/s transmis – comment définir le /s de Gb/s pour deux ordinateurs séparés de plusieurs centaines de kilomètres ?), le transfert de temps est un concept bien concret (« je suis en retard, je ne vais pas arriver à l'heure à mon rendez-vous » – mais comment garantir que l'interlocuteur base la date du rendez-vous sur la même référence ?). Le transfert de temps et de fréquence sont deux activités duales qui ne respectent pas les mêmes contraintes. Une fréquence ou son intégrale la phase, décrit la caractéristique d'un signal périodique : par exemple, une sinusoïde à 10 MHz voit ses propriétés répétées toutes les 100 ns, et il est impossible de distinguer une période de sa voisine 100 ns plus tard. Si le passage à 0 du signal varie un peu dans le temps par rapport à une référence, l'oscillateur peut voir sa fréquence ajustée si nécessaire, mais aucune datation absolue n'est possible. Au contraire, le transfert de temps nécessite de transférer un évènement bref (« maintenant ») – donc intrinsèquement large bande, au contraire du transfert de fréquence qui est à bande étroite – et une datation absolue, et ne doit donc pas se répéter trop rapidement pour laisser le temps de fournir toutes les informations associées à l'impulsion (la date et l'horaire). Le signal 1 PPS (1 Pulse Par Seconde pour sa traduction française) [14, p.247] fournit une telle information : par définition, son front montant est supposé aligné avec l'information de date à transmettre (le début de la seconde), tandis que la durée et donc la position du front descendant ne sont pas normalisés. En parallèle de ce front montant, une information numérique est en général transmise pour informer de la date et l'heure associées à ce front. On se retrouve donc avec une configuration de l'horloge parlante : « au prochain top, il sera XX h ». Les récepteurs GPS ne sont pas les seuls à fournir 1 PPS : White Rabbit, implémentation de PTP (Precision Time Protocol) du CERN, fournit aussi son 1 PPS en parallèle du transfert de fréquence avec son oscillateur à 10 MHz. À titre d'illustration, la figure 8 de droite propose un exemple de mesure pendant un week-end du délai dt entre les 1 PPS issus de deux liens White Rabbit indépendants entre l'ENSMM et l'Observatoire de Besançon. Les fluctuations maximales sont de l'ordre de 200 ps, avec un écart type de l'ordre de 20 ps.

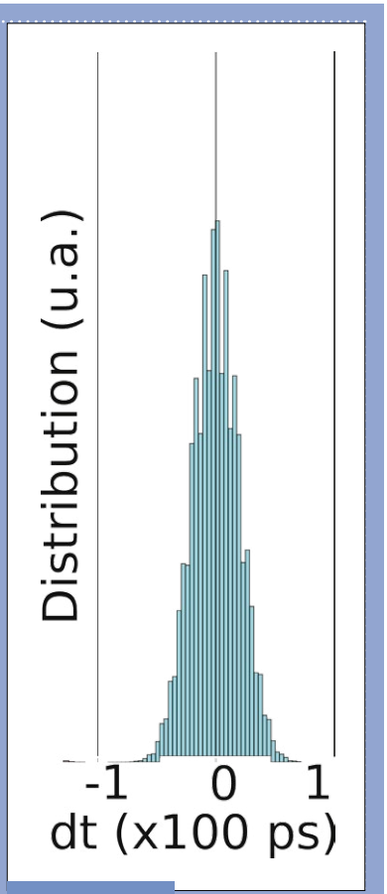


FIGURE 8

Distribution du délai entre deux signaux 1 PPS issus de deux liens White Rabbit (graphique de É. Meyer, Obs. Besançon).

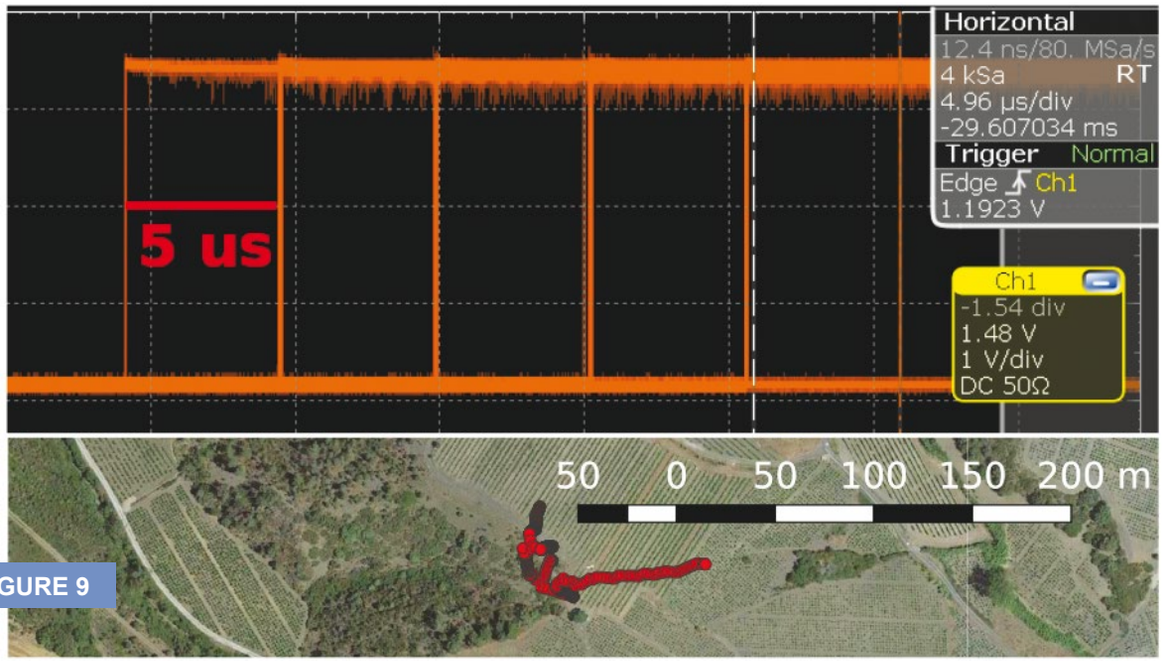


FIGURE 9

Haut : sortie 1 PPS d'un récepteur U-Blox Neo M8T indiquant le temps GPS sur lequel de nombreux oscillateurs (GPSDO) s'asservissent en faisant confiance au signal issu de la constellation de satellites, ici induite en erreur par pas de 5 µs. Bas : impact négligeable de la dérive de temps GPS induite par notre attaque sur l'estimation de la position du récepteur.

GPS est basé sur un transfert de temps, à partir duquel la position au sol est déduite par trilatération des *pseudo-ranges*. Les horloges embarquées dans les satellites ne sont pas exactes, mais dérivent. Plutôt que modifier le comportement des horloges pour les ramener à leur date nominale, il est judicieux de laisser les horloges dériver de façon déterministe et informer l'utilisateur de l'écart de temps entre l'horloge embarquée dans chaque satellite et le temps GPS. Cette information, remise périodiquement à jour, est transmise dans le message de navigation émis par chaque satellite **[9, p.57]**. Que se passe-t-il si nous décalons ce temps d'une valeur connue, par exemple par pas de 5µs dans l'application qui va suivre ?

Cette opération est simple dans une implémentation de radio logicielle : un paramètre du message de navigation à modifier de façon cohérente pour tous les satellites de la constellation, et le temps s'est virtuellement translaté. Comme les pseudo ranges sont calculés par `pluto-gps-sim` en tenant compte de ce décalage temporel pour une position au sol imposée, et puisque de toute façon toutes les horloges sont décalées de la même valeur (qui se compense lors de la trilatération), le récepteur ne verra pas sa position bouger. La figure 9 démontre ce concept en présentant d'une part l'impulsion 1 PPS qui représente le temps GPS sur un récepteur U-Blox Neo M8T, et d'autre part la position de ce même récepteur tel que fournie par ses trames NMEA. Nous constatons que le 1 PPS saute par pas de 5 µs que nous avons introduit volontairement toutes les 2 minutes (pour rappel, le 1 PPS d'un récepteur GPS monofréquence fluctue typiquement de ± 100 ns), mais que la position ne s'est de loin pas déplacée des quelques 7 km que laisserait présager un décalage du temps de 25 µs au bout de 10 minutes d'expérience. Quelques sauts de positions correspondent au temps de convergence des boucles d'asservissement du récepteur qui sont quelque peu surprises de ces sauts soudains de temps. Nous avons vérifié que le signal 1 PPS peut aussi être induit à dériver linéairement ou quadratiquement en jouant non pas sur l'offset de l'horloge de chaque satellite, mais sur sa dérivée première (paramètre AF1) ou seconde (AF2).

Photo d'illustration.

PALLIATIFS

Un numéro spécial récent de Proc. IEEE faisait l'état de l'art sur les attaques sur les systèmes de positionnement par satellite. Une première solution évidente consiste à fusionner multitude de données afin d'identifier une source d'incohérence : ajouter aux signaux issus de multiples constellations de satellites des informations émises au sol telles que WiFi ou téléphonie mobile (GSM, UMTS) réduit le risque, mais ne fait que repousser le problème, puisque ces sources additionnelles sont brouillables ou elles aussi leurrables (penser OpenBSC **[15]**).

Une faiblesse des constellations GNSS tient en leur très faible signal qui est aisément supplanté par des émetteurs au sol : une tendance tient à se positionner au sol par des signaux issus de constellations de satellites en orbite basse (e.g. Iridium NEXT), avec des signaux au sol considérablement plus puissants et cryptés. L'Europe a malheureusement décidé d'abandonner le déploiement d'un réseau très basse fréquence (VLF) d'émetteurs de localisation (eLORAN) qui pallierait aux attaques sur GPS : les États-Unis maintiennent ce réseau en l'étendant au Japon et à la Corée du Sud – tous deux susceptibles d'être brouillés par leur voisin qu'est la Corée du Nord – tandis que l'Arabie Saoudite, la Chine et la Russie (Chayka) maintiennent leur réseau de stations VLF pour conserver une autonomie vis-à-vis des constellations spatiales de positionnement. Leurrer un signal VLF puissant – 360 kW à 100 kHz pour eLORAN – nécessite une infrastructure autrement plus lourde qu'un émetteur de radio logicielle.

Finalement, après avoir mentionné les contraintes physiques d'une constellation de satellites (décalage Doppler respectant les lois de Kepler, distribution des sources dans l'espace imposées par la mécanique céleste), la solution ultime semble tenir dans l'utilisation d'un réseau d'antennes (ou alternativement d'une unique antenne en mouvement) pour identifier la direction d'arrivée des signaux et leur cohérence avec la géométrie de la constellation. Mener une attaque distribuée dans laquelle une multitude d'émetteurs synchronisés en temps et en fréquence pour reproduire la direction d'arrivée de chaque signal semble actuellement inaccessible, et cette solution semble celle favorisée par la majorité des articles de la revue mentionnée au début de cette section **[16][17][18][19][20]**. Cette approche s'accommode parfaitement d'une solution tout logiciel du récepteur (radio logicielle), tel qu'en atteste la participation des auteurs de GNSS-SDR **[21]**.

CONCLUSION

Après avoir rappelé quelques principes de base de fonctionnement de GPS, de la couche physique à la couche logicielle, nous avons démontré l'aisance avec laquelle il est aujourd'hui possible de leurrer GPS, même dans des situations aussi critiques que les systèmes de navigation de voiture. L'objectif est de sensibiliser le lecteur aux dangers associés à une confiance aveugle dans les systèmes de positionnement par satellite, en particulier pour des infrastructures critiques : dans ce cas, des solutions de repli pour pallier un brouillage, voir une détection de leurrage par incohérence des signaux reçus (décalage Doppler en dehors de la gamme physiquement atteignable, puissance de signal excessive) sont nécessaires. Finalement, la solution multi-antennes avec mesure de direction d'arrivée des signaux de la constellation semble la solution la plus robuste pour se prémunir des attaques de leurrage.

REMERCIEMENTS

Cette étude a été motivée par la création du laboratoire commun FASTLAB entre l'Institut FEMTO-ST, l'Observatoire de Besançon et la société Gorgy Timing. L'équipement acquis dans le cadre du Labex FIRST-TF et Equipex OscillateurIMP a fourni les signaux de référence pour qualifier les divers oscillateurs utilisés dans cette présentation, bien que nous insistions sur la capacité de tout amateur éclairé à reproduire ces expériences. Les références bibliographiques qui ne sont pas librement disponibles sur le web ont été obtenues auprès de Library Genesis à **gen.lib.rus.ec**, une ressource incontournable pour nos recherches.

NOTES

[A] Un balun (*balanced-unbalanced*) est un transformateur chargé de convertir le signal non-balancé (distinguant masse et signal) vers un signal balancé pour attaquer les deux brins d'antenne qui sont symétriques.

[B] Cet oscillateur est disponible pour une centaine d'euros sur eBay. Alternativement, une horloge rubidium, disponible pour le même ordre de prix en seconde main telle que la Symmetricom X72, fera certainement l'affaire.

Photo d'illustration.

RÉFÉRENCES

[1] T.Humphreys, « How to fool a GPS », **www.ted.com/talks/todd_humphreys_how_to_fool_a_gps** (2012) puis **news.utexas.edu/2013/07/29/ut-austin-researchers-successfully-spoof-an-80-million-yacht-at-sea**

[2] J.F. Zumberge et G. Gendt, « The demise of selective availability and implications for the international GPS service », Physics and Chemistry of the Earth 26 (6–8), pp. 637–644 (2001), **https://www.gps.gov/systems/gps/modernization/sa/**

[3] « Satellite-derived Time and Position: A Study of Critical Dependencies », **www.gov.uk/government/uploads/system/uploads/attachment_data/file/676675/satellite-derived-time-and-position-blackett-review.pdf**

[4] R.T. Ioannides, T. Pany et G. Gibbons, « Known Vulnerabilities of Global Navigation Satellite Systems, Status, and Potential Mitigation Techniques », Proc. IEEE 104 (6) 1174--1194 (Juin 2016)

[5] K. Zeng & al., « All Your GPS Are Belong To Us: Towards Stealthy Manipulation of Road Navigation Systems », 27th USENIX Security Symposium (2018), disponible à **people.cs.vt.edu/gangwang/sec18-gps.pdf**

[6] L. Huang et Q. Yang, « Low cost GPS simulator: GPS spoofing by SDR », DEFCON 23 (2015), **media.defcon.org/DEF%20CON%2023/DEF%20CON%2023%20presentations/DEFCON-23-Lin-Huang-Qing-Yang-GPS-Spoofing.pdf**

[7] D. Robinson , « Using GPS Spoofing to control time », DEFCON 25 (2017), **www.youtube.com/watch?v=isiuTNh5P34**

[8] J.-M Friedt, G. Cabodevila, « Exploitation de signaux des satellites GPS reçus par récepteur de télévision numérique terrestre DVB-T », *Open Silicium n°15* (2015)

[9] ESA, « GNSS data processing », **gssc.esa.int/navipedia/GNSS_Book/ESA_GNSS-Book_TM-23_Vol_I.pdf** et sa liste d'exercices **gssc.esa.int/navipedia/GNSS_Book/ESA_GNSS-Book_TM-23_Vol_II.pdf**

[10] « Norme RINEX - The Receiver Independent Exchange Format, Version 3.03 », **https://kb.igs.org/hc/en-us/article_attachments/202583897/RINEX_303.pdf**

[11] « Global Positioning System Standard Positioning Service Signal », **www.gps.gov/technical/ps/1995-SPS-signal-specification.pdf** (1995)

[12] A. Thiel et M. Ammann, « Anti-Jamming techniques in u-blox GPS receivers », **www.u-blox.com/sites/default/files/products/documents/u-blox-AntiJamming_WhitePaper_%28GPS-X-09008%29.pdf** (octobre 2009)

[13] W. Lewandowski & al., « Testing Motorola Oncore GPS Receiver and Temperature-Stabilized Antennas for Time Metrology », Proc. 28th Annual Precise Time and Time Interval Systems and Applications Meeting, **https://tycho.usno.navy.mil/ptti/1996papers/Vol%2028_37.pdf** (1996)

[14] C. Audoin et B. Guinot, « The measurement of time – Time, frequency and the atomic clock », Cambridge Univ. Press (2001)

[15] H. Welte, « OpenBSC network-side GSM stack », SSTIC 2010, **www.sstic.org/media/SSTIC2010/SSTIC-actes/Projet_OpenBSC/SSTIC2010-Slides-Projet_OpenBSC-welte.pdf**

[16] I.J. Gupta, I.M. Weiss, et A.W. Morrison,, « Desired Features of Adaptive Antenna Arrays for GNSS Receivers », Proc. IEEE 104 (6) 1195--1206 (juin 2016)

[17] J.L. Volakis, A.J. O'Brien, et C.-C. Chen, « Small and Adaptive Antennas and Arrays for GNSS Applications », Proc. IEEE 104 (6) 1221--1232 (juin 2016)

[18] A. Broumandan, A. Jafarnia-Jahromi, S. Daneshmand et G. Lachapelle, « Overview of Spatial Processing Approaches for GNSS Structural Interference Detection and Mitigation », Proc. IEEE 104 (6) 1246--1257 (juin 2016)

[19] M. Cuntz, A. Konovaltsev et M. Meurer, « Concepts, Development, and Validation of Multiantenna GNSS Receivers for Resilient Navigation », Proc. IEEE 104 (6) 1288--1301 (juin 2016)

[20] M.G. Amin, X. Wang, Y.D. Zhang, F. Ahmad, et E. Aboutanios, « Sparse Arrays and Sampling for Interference Mitigation and DOA Estimation in GNSS », Proc. IEEE 104 (6) 1302--1317 (juin 2016)

[21] C. Fernàndez-Prades, J. Arribas, et P. Closas, « Robust GNSS Receivers by Array Signal Processing: Theory and Implementation », Proc. IEEE 104 (6) 1207--1220 (juin 2016)

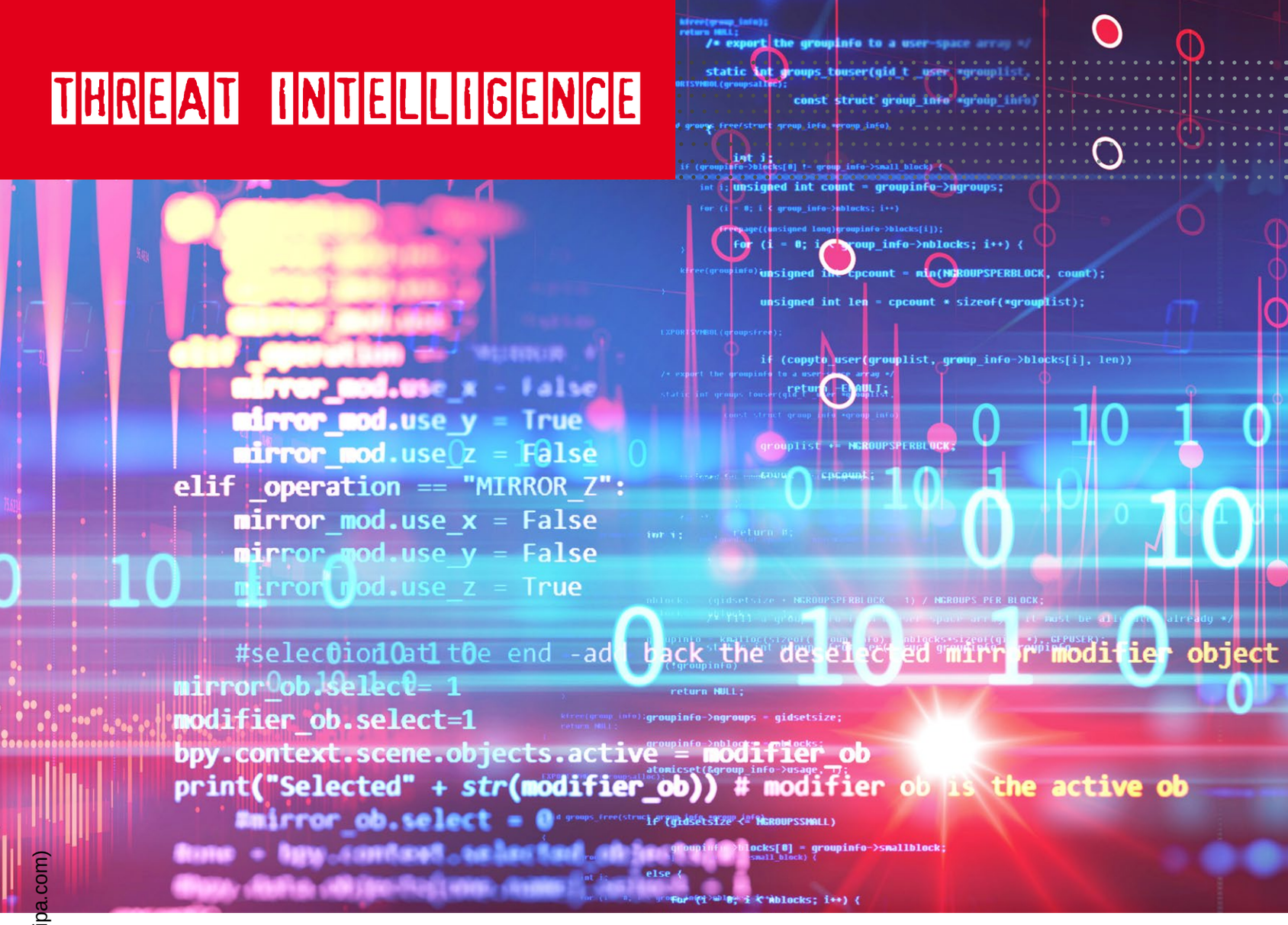
ACTUELLEMENT DISPONIBLE
HACKABLE N°28 !

UTILISEZ LA MÉMOIRE « SECRÈTE » DE VOS ESP8266 !

NE LE MANQUEZ PAS

CHEZ VOTRE MARCHAND DE JOURNAUX ET SUR :

https://www.ed-diamond.com



INTRODUCTION AU FRAMEWORK ATT&CK DU MITRE

Vincent MÉLIN

Le framework ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) du MITRE fait de plus en plus parler de lui dans le monde des modèles de description d'attaques. Focalisé sur la description des tactiques et techniques « post compromission », il offre des clés pour monter ou faire évoluer des programmes de protection, de détection et de réaction. Cet article se propose de présenter ATT&CK ainsi que ses applications possibles dans des contextes opérationnels type CERT/SOC/CSIRT.

Outre son focus sur les activités « post compromission », le framework ATT&CK du MITRE **[MATRIX]**, une organisation américaine à but non lucratif gérant des centres de R&D financés par des fonds fédéraux, se distingue des autres modèles de description d'attaque par une approche descriptive systématique, basée sur une étude continue des attaques et non orientée sur les outils, mais sur les méthodes. À ce titre, on peut le classer dans la catégorie de la threat intelligence tactique.

Cet article présentera le framework ainsi que quelques usages concrets, notamment dans le cadre de la détection et de la réponse à incidents.

ATT&CK ET OUTILS ASSOCIÉS

ATT&CK est une base de connaissance de comportements d'attaque. Il s'appuie sur l'étude d'exemples concrets, issus d'analyses publiques d'attaques, et référence, entre autres, des outils et groupes spécifiques. Il est conçu sur les principes suivants :

- inclure la détection post-compromission à l'arsenal de défense ;
- focus sur les comportements ;
- utiliser un modèle basé sur l'étude de la menace ;
- itérer par conception ;
- développer et tester en environnement réaliste.

L'objectif d'ATT&CK est d'évaluer et d'améliorer les capacités de défense en s'appuyant sur ces connaissances. La matrice ATT&CK peut être vue comme une forme de kill chain détaillée et spécialisée. À noter que les différentes tactiques ne représentent pas une progression temporelle de l'attaque.

La matrice

La matrice ATT&CK est le cœur du framework. Elle présente sous forme visuelle et textuelle les différentes tactiques et techniques liées aux activités de post-compromission. Tout l'intérêt de la matrice est de se focaliser sur les comportements, donc les effets « visibles » des actions des attaquants plutôt que sur les outils qu'ils utilisent. Il est plus simple pour un attaquant de modifier ses outils que de changer ses modes de fonctionnement (« pyramid of pain » **[PAIN]**) ; de plus, les différentes actions laissent plus ou moins de traces sur le système.

Dans la terminologie ATT&CK :

- une « tactique » décrit un objectif (« que cherche faire l'attaquant à cette étape ? ») ;
- une « technique » est une manière d'atteindre l'objectif (« comment atteindre l'objectif ? »).

Onze tactiques aux noms explicites sont listées :

- Accès initial ;
- Exécution (de code, notamment en conjonction avec d'autres tactiques – ex. mouvement latéral) ;
- Persistance ;
- Élévation de privilèges ;
- Évasion (échapper à la détection) ;
- Accès aux données d'authentification ;
- Découverte ; à ne pas confondre avec les activités précédant l'attaque (décrites dans la matrice spécifique « PRE-ATT&CK »). L'objectif de ces techniques est de permettre à l'attaquant de comprendre l'environnement dans lequel il évolue et de s'orienter vers sa ou ses prochaines cibles ;

- Mouvement latéral ;
- Collecte d'informations ;
- Exfiltration ;
- Command & Control.

Sans garantir l'exhaustivité, plus de 200 techniques se répartissent dans ces différentes tactiques.

Chaque technique dispose d'une référence (ex. « T1009 – Binary Padding ») et est documentée selon la même structure :

- une description « fonctionnelle » ;
- des exemples avec des références. Chaque exemple est lié à un groupe d'attaquant (avec un pointeur vers la description de ce groupe) et/ou un outil particulier, encore une fois avec un pointeur vers sa description ;
- une section « mitigation », avec des propositions de moyens de protection contre la technique ;
- une section « détection » indiquant comment la technique peut être détectée.

La liste des techniques prend en compte les fonctionnalités disponibles sous différents OS. Par exemple, la persistance pourra être réalisée via :

- Linux/macOS - T1156 « .bash_profile and .bashrc » ;
- Windows - T1103 « AppInit DLLs » ;
- macOS – T1157 « dylib Hijacking ».

Les outils associés

» Groupes et logiciels

Deux autres sections complètent la matrice :

- « Groupes » ;
- « Logiciels ».

La section « Groupes » détaille des groupes d'attaquants connus et documentés. Chaque groupe (avec ses alias connus) est associé à une référence (ex. APT1, G0006) est succinctement décrit, les techniques et logiciels qu'il est connu pour utiliser sont listés.

Côté « logiciel », chacun est lié à une ou des techniques (ex. Cachedump, S0119, est lié à la technique T1003 « Credential Dumping ») et à une liste de groupes.

Dans les deux cas, des références externes sont encore une fois fournies.

Photo d'illustration.

On voit donc que les différents types « d'objets » bénéficient de références croisées permettant de les mettre dans un contexte plus global.

ATT&CK Navigator

Le MITRE fournit une interface interactive pour la matrice : le Navigator **[NAV]** qui permet de visualiser la matrice et de sélectionner les techniques.

CAR et CARET

CAR **[CAR]**, pour *Cyber Analytics Repository*, est une base de connaissances destinée à aider à la découverte de traces d'exécution des techniques décrites dans la matrice. Un « analytic » est une manière de détecter des comportements adverses décrits par ATT&CK qui combine l'identification des comportements à détecter, des sources de données et des capteurs nécessaires à la détection et une logique de détection.

Le MITRE identifie plusieurs types « d'analytics » en fonction du type d'activité que la méthode est conçue pour détecter (***TTP*** **[TTP]**, ***Attribution***, ***Posture/Hygiène***, ***Situational Awareness***, ***Forensics***, ***Anomaly***, ***Statistical***, ***Investigative***, ***Malware***, ***Event Characterization***).

Dans les faits, la liste des analytics (une quarantaine) contient majoritairement des analytics de type « situational awareness » et « TTP » applicables à des environnements Windows. Le premier type donne des informations sur l'état général du système, pouvant être utilisé pour suivre les évolutions de cet état dans le temps (ex. « CAR-2013-01-002: Autorun Differences). Le second fait ce que son nom indique (ex. « CAR-2013-03-001: Reg.exe called from Command Shell »).

Chaque méthode (« analytics ») est :

- nommée (ex. « CAR-2013-01-003: SMB Events Monitoring ») ;
- typée ;
- décrite fonctionnellement ;
- associée à un type d'observable (ex. fichier pcap) et à une liste de données en sortie ;
- associée à une liste de techniques qu'il est possible de détecter en l'appliquant ;
- outillée par un pseudo code permettant de la mettre en œuvre.

Chaque méthode donnera ensuite lieu à l'analyse des données en sortie pour essayer de détecter des signes des techniques associées tout en sachant que tous les évènements remontés ne seront pas des attaques.

Exemple de pseudo code (CAR-2013-05-002 « Suspicious Run Locations », type TTP) :

```
processes = search Process:Create
suspicious_locations = filter process where (
   image_path == "*:\RECYCLER\*" or
   image_path == "*:\SystemVolumeInformation\*" or
   image_path == "%windir%\Tasks\*" or
   image_path == "%systemroot%\debug\*"
)
output suspicious_locations
```

CARET (pour *CAR Exploration Tool* – **[CARET]**) est un outil de visualisation de la base de connaissances. Il suffit de cliquer sur le nom d'un « analytics » pour que les techniques associées soient identifiées dans la matrice ATT&CK.

» Émulation d'attaques : caldera

Afin d'automatiser l'usage de la matrice pour émuler des attaques réalistes, le MITRE fournit l'outil en python Caldera, présenté entre autres à Black Hat Europe en 2017 **[BHE]**.

Caldera **[CGH]** permet d'automatiser des scénarios d'attaque basés sur des modèles. Chaque action est associée à une « pré-condition » (exigence pour que la technique puisse être appliquée) et à une « conséquence » (état après exécution de la technique). L'automate est censé pouvoir évoluer dynamiquement en fonction des préconditions remplies et mises à jour au fur et à mesure de l'exécution du scénario.

À noter un autre projet du même genre développé par Uber : Metta **[MGH]**.

» Pre-Att&ck et Mobile Profile

Le MITRE fourni deux autres bases de description d'attaques, l'une orientée « pré compromission » (PRE-ATT&CK) et l'autre orientée mobile (ATT&CK Mobile Profile).

PRE-ATT&CK **[PRE]** comporte essentiellement des activités difficilement détectables par les équipes opérationnelles (reconnaissance, collecte d'informations, identification des faiblesses...) en particulier parce que la plupart d'entre elles n'impliquent pas d'interaction technique directe avec la cible (ex. social engineering et OSINT en passant par les réseaux sociaux).

Le Mobile Profile **[MOB]** a une portée explicite avec 3 sous-parties : « obtain device access » (pré compromission), « use device access » (qui reprend les 11 tactiques de la matrice ATT&CK, orientées action sur l'équipement) et « network based effects » (attaques de mobile par le réseau, par exemple en redirigeant les appels/SMS par l'exploitation du protocole SS7 ou en effectuant un déni de service par brouillage du signal).

QUELQUES EXEMPLES D'UTILISATION AU SEIN D'UN CERT ET/OU D'UN SOC

De par sa structure, son approche systématique basée sur l'expérience et l'enrichissement continu et sa portée opérationnelle, la matrice ATT&CK présente de nombreux intérêts pour les équipes en charge de la détection et du traitement des incidents de sécurité.

Formation et montée en compétences

Bien qu'il ne soit pas simple, quand on fait partie d'un CERT ou d'un SOC, de consacrer le temps nécessaire à l'analyse des rapports d'attaques publiés en continu et en nombre grandissant, ATT&CK est un recueil de techniques que tout analyste se devrait de connaître. Le MITRE a fait un travail de synthèse qui permet de disposer d'un résumé des tactiques et techniques avec des pointeurs vers les ressources permettant d'aller plus dans le détail.

Photo d'illustration.

ATT&CK peut donc servir à faire monter les équipes en compétences et à partager un langage commun. Dans l'idéal, un analyste pourra plus facilement associer un évènement à une tactique et y reconnaître les signes d'exécution des techniques associées.

L'enjeu d'une telle démarche est son maintien dans le temps, la limite d'une approche basée sur l'existant et le connu étant que de nouvelles techniques peuvent apparaître.

La matrice étant ouverte, il est également possible de se l'approprier et de l'étendre en fonction des besoins et de l'expérience du CERT/SOC.

Amélioration de la supervision

» Conception et implémentation de scénarios

Une partie des techniques peut être vue comme une source « sur étagère » de scénarios fonctionnels de détection pour un SOC. Les descriptions de techniques peuvent être étudiées une par une et la section « détection » déclinée en règles opérationnelles. La couverture des techniques identifiées comme utilisables servira à construire la feuille de route du SOC en plus d'autres référentiels tels que celui de l'ETSI **[ETSI]**.

En plus des techniques elles-mêmes, les analytics de type « TTP » seront idéaux pour commencer. Les techniques étant par définition relativement génériques, le passage de la technique au scénario dans un outil de détection/protection nécessitera une phase d'étude pour identifier des spécificités observables.

Prenons par exemple le cas T1170 « mshta ». La section « détection » indique qu'il faut mettre en place du *process tracking* et analyser les arguments avec lesquels `mshta.exe` est appelé, les comparer à un historique connu et y chercher des anomalies. C'est en soi un vaste programme. À part si l'exécution de fichier hta n'est pas commune dans l'environnement observé auquel cas, chaque exécution pourrait être une anomalie à traiter. Dans le cas contraire, la logique pour séparer le plus automatiquement possible le « a priori normal » du « a priori « suspect » reste à déterminer.

Le principe des analytics est justement de donner des outils d'identification des techniques.

Par exemple, au lieu d'alerter pour toutes les invocations de powershell, l'analytic « CAR-2014-04-003: Powershell Execution » recherche les exécutions de powershell dont le parent n'est pas `explorer.exe`. On pourrait ajouter des alertes sur des exécutions suspectes de scripts en recherchant l'usage d'options telles que `-e` (encodage base64) ou `-ep bypass` dans la ligne de commandes.

La logique de l'exemple « Suspicious Run Locations » est simple à implémenter dans un SIEM (comparer un chemin d'exécution à une liste prédéfinie), mais nécessite de disposer des logs de « process tracking » des systèmes Windows à surveiller (EventID 4688).

Dans d'autres cas, notamment pour les analytics à base de PCAP (exemple : usage malveillant des protocoles SMB ou RPC), les moyens de surveillance réseau seront mis à profit.

Les logs standards de Windows devront parfois être augmentés par l'utilisation de Sysmon **[SYSMON]** comme par exemple « CAR-2013-07-001: Suspicious Arguments » (EventID 1) qui nécessite d'avoir accès aux arguments passés et pas seulement au chemin de l'exécutable.

```
process = search Process:Create
port_fwd = filter process where (command_line match "-R .* -pw")
scp = filter process where (command_line match "-pw .* .* .*@.*"
mimikatz = filter process where (command_line match "sekurlsa")
rar = filter process where (command_line match " -hp ")
archive = filter process where (command_line match ".* a .*")
ip_addr = filter process where (command_line match \d{1,3}\.\d{1,3}\.
\d{1,3}\.\d{1,3})

output port_fwd, scp, mimikatz, rar, archive, ip_addr
```

D'autres analytics comme ceux de type « anomalies » sont utilisables. Il sera cependant nécessaire de construire des bases de référence. L'analytics « CAR-2013-09-005: Service Outlier Executables » est un exemple typique (comparaison de liste des services déclarés à une liste « historique » de services) :

```
processes = search Process:Create
services = filter processes where (parent_image_path == "C:\Windows\
System32\services.exe")
historic_services = filter services (where timestamp < now - 1 day AND
timestamp > now - 1 day)
current_services = filter services (where timestamp >= now - 1 day)
new_services = historic_services - current_services
output new_services
```

L'étude de la section « logiciels » viendra compléter la feuille de route. Une démarche telle que celle menée par le CERT japonais **[CERTJP]** pourrait être généralisée pour être en mesure de détecter différents outils non pas sur leur « forme », mais par rapport aux comportements exhibés. A minima, un état des lieux de la couverture des logiciels listés par le MITRE par les outils de détection/protection en place (antimalware, IDS/IPS...) est à réaliser.

Le « navigator » pourra également être utilisé pour concevoir des règles de corrélation tactique afin de fiabiliser les alertes « unitaires » et de modéliser des séquences progressives d'une attaque. Le but d'une telle approche serait de déclencher une alerte « corrélée », de préférence avant la phase de réalisation des objectifs (ex. exfiltration ou destruction de données sensibles).


Photo d'illustration.

La nomenclature peut également être utilisée pour caractériser les scénarios du SOC afin de donner du contexte aux évènements et alertes avec l'identifiant des techniques.

» Évaluer la couverture de détection

ATT&CK peut être envisagé dans le cadre d'exercices Red Team / Blue Team pour évaluer les capacités du SOC à détecter les incidents. Le MITRE présente dans « Finding Cyber Threats with ATT&CK Based Analytics » **[WP]** des exemples de ce type d'usage, plutôt dans le cadre de la création des « analytics ».

Les deux équipes pourront modéliser l'attaque sur la base des techniques et comparer leurs modélisations. L'objectif de la Blue Team sera d'identifier l'ensemble des actions de la Red Team. L'étape suivante sera la capitalisation des résultats et la mise à jour de la roadmap de couverture.

Au-delà de l'approche « humaine », les scénarios de détection pourront être testés plus ou moins régulièrement en utilisant Caldera ou Metta. À l'extrême, le SOC pourra décider d'inclure leur usage dans les cahiers de test / validation de scénarios (à voir la charge de développement induite).

Ces outils d'émulation d'attaque pourront également être mis à profit pour tester des solutions de sécurité « sur étagère » de manière standardisée et automatisée.

» Limites

Il y a quelques limites à la déclinaison directe des techniques en scénarios à implémenter par le SOC :

- la plupart des activités décrites dans la matrice ne sont pas détectables par des solutions périmétriques ou à base de signatures. L'application de la matrice dans les activités de détection nécessitera d'avoir une couverture importante du SI en termes de collecte d'informations (moyens et contenu). Le SOC pourra proposer des évolutions dans le choix ou la configuration des solutions de sécurité déployées, mais ces évolutions ne sont pas toujours simples à obtenir surtout si le SOC n'a pas la main sur ces solutions. Il conviendra donc d'inscrire la démarche dans le temps ;
- le temps et les efforts nécessaires à une couverture exhaustive de la matrice ne sont pas à négliger d'autant que, comme évoqué dans le point précédent, les moyens techniques ne seront pas nécessairement à la main du SOC. De plus, certaines techniques peuvent être réalisées d'un grand nombre de façons ;

- comme pour toute supervision comportementale, le risque de faux-positif est non négligeable. Beaucoup d'alertes nécessiteront un travail d'analyse alors que les équipes des SOC sont déjà surchargées d'alertes. Ce type de détection peut également avoir un impact important sur les performances du SIEM. Le SOC pourra tester des produits spécialisés à base de machine learning pour créer et maintenir automatiquement des « baselines » et faire de la détection d'anomalies (« UEBA » - profiling comportemental, analyse par groupe de pairs, clustering & classification). Il faudra garder à l'esprit que ces produits sont assez nouveaux, que ce sont souvent des boîtes noires et qu'il peut être difficile d'interpréter l'absence d'alarmes.

Nouvelles méthodes de détection : le threat hunting

Pour compenser partiellement les limites évoquées ci-dessus, la détection « temps réel » et les travaux d'analyse pourront être complétés par des phases de recherche active de traces de compromission (*threat hunting*).

L'usage du framework ATT&CK revient souvent dans les discussions autour du *threat hunting*.

L'objectif du présent article n'est pas de rentrer dans les détails de cette démarche. De nombreux articles sur le sujet existent d'ores et déjà, par exemple **[ENDGAME].** L'ensemble des travaux CAR/CARET est directement applicable.

Comme pour la supervision sécurité, la disponibilité des données et les capacités d'analyse sont des facteurs clé de réussite. Et comme pour les scénarios du SOC, l'application de l'ensemble des « analytics » ne pourra pas être réalisée en une seule campagne et le CAR servira de roadmap à l'équipe de *threat hunting*. Au titre de la capitalisation, l'équipe de *threat hunting* pourra également développer de nouveaux analytics dont certains pourront être transformés en nouveaux scénarios de détection dans le SIEM.

Réponse à incidents

Une autre manière d'utiliser le framework consiste à appliquer sa structure et sa nomenclature à l'analyse, l'évaluation et la description d'un incident en cours.

» En cours d'investigation

Les analystes peuvent utiliser la matrice pour appliquer une démarche systématique au processus d'analyse de l'incident.

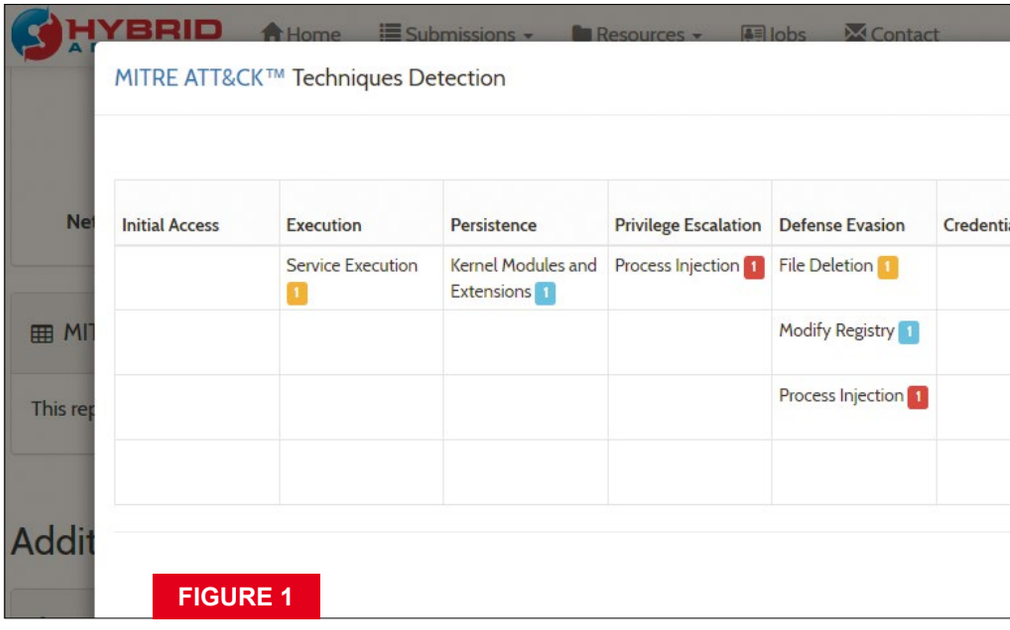


FIGURE 1

Le site Hybrid Analysis utilise cette approche dans ses rapports d'exécution. La capture d'écran montre le positionnement des différents indicateurs comportementaux d'un malware par rapport aux tactiques ATT&CK.

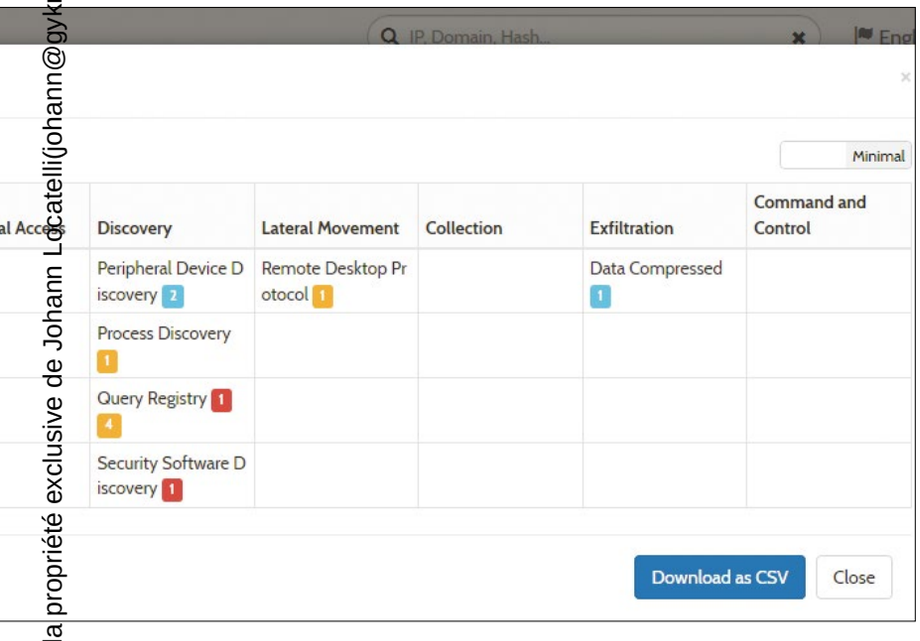
Disposer d'un *playbook* voire d'outils automatisés (type EDR) pour rechercher des signes d'exécution de chacune des techniques rendra le processus plus formel.

Dans le cadre d'une boucle OODA (*Observe, Orient, Decide and Act* - **[OODA]**), l'équipe d'investigation pourra positionner les actions identifiées dans la matrice ou dans le « navigator » afin d'estimer à quelle étape se trouve l'attaquant par rapport aux traces trouvées et quelles pourraient être ses prochaines actions.

Les personnes en charge de l'analyse des outils rencontrés dans le cadre de l'incident pourront caractériser les comportements observés par rapport aux tactiques ATT&CK.

Ces analyses caractérisées aideront les personnes en charge de la réponse à incident dans la compréhension des actions et des capacités de l'attaquant.

La connaissance des « modus operandi » des différents groupes décrits peut également alimenter le processus d'investigation, tout comme la connaissance des outils permet de rechercher des traces de manière plus exhaustive. Il conviendra, afin de ne pas introduire de biais, de ne pas utiliser cette connaissance comme axe de travail principal, mais plutôt pour contextualiser les faits remontés par le processus d'investigation.



Enfin, comme pour le *threat hunting*, la base de connaissances CAR, adaptée aux outils internes et enrichie par les traitements ad-hoc viendra compléter le processus d'investigation, dans la mesure où les données nécessaires sont disponibles (notamment pour tirer parti des analytics de type « Situational Awareness »).

» Dans le rapport d'incident

L'usage du framework permettra d'avoir un langage commun au sein du SOC/CERT/CSIRT et facilitera les échanges avec l'ensemble des parties prenantes.

Ainsi, le rapport d'incident décrira les phases de l'intrusion, les techniques utilisées et les artefacts relevés. L'analyse de chacun des systèmes concernés pourra être présentée de façon standardisée et la chronologie de l'incident pourra être annotée avec les tactiques identifiées.

Le tableau, page suivante, propose l'application d'ATT&CK au scénario Red Team issu du dernier rapport MTREND de FireEye **[MTREND]**.

» Capitalisation et « lessons learned »

La phase de capitalisation est essentielle dans le processus de réponse à incident pour améliorer dans le temps la posture de défense de l'organisation.

Les équipes opérationnelles peuvent s'approprier les bases ATT&CK et CAR et les faire évoluer par rapport aux résultats du traitement interne des

	Étapes dans le temps							
Initial Access	Phishing with link to malicious HTA **T1189** Drive-by compromise							
Execution		Exécution fichier HTA **T1170** MSHTA	Payload exécutée par Certutil.exe **T1218** Signed Binary Proxy Execution		**T1072** Third-party Software (Symantec symerr.exe) Exécution			
Persistence					**T1084** WMI Event Subscription			
Privilege escalation								
Defense Evasion		Code obfusqué **T1140** Deobfuscate/Decode Files or Information			**T1073** DLL Side-Loading (DLL malveillante cclib.dll par symerr.exe)			
Credentiel Access						Recherche de comptes avec un « userPassword » en clair **T1003** Credential Dumping	Une fois un compte trouvé, connexion et utilisation de Mimikatz pour récupérer des comptes admin de domaine **T1003** Credential Dumping	
Discovery						Recherche de comptes avec un « userPassword » en clair **T1087** Account Discovery		
Lateral Movement								
Collection								
Exfiltration								
Command and Control				**T1041** Commonly used port **T1172** Domain Fronting **T1032** Standard Cryptographic Protocol				

FIGURE 2

	Connexion avec les comptes admin de domaine trouvés **T1078** Valid Accounts					
	Usage de WMI pour lancer le binaire signé MSBuild.exe et exécuter des commandes à distance **T1218** Signed Binary Proxy Execution					
		Brute force des soft token **T1110** Brute force Obtention du PIN RSA via un keylogger **T1056** Input Capture				
				Découverte des routes vers les BDD **T1018** Remove System Discovery		
	Usage de WMI pour lancer le binaire signé MSBuild.exe et exécuter des commandes à distance **T1218** Signed Binary Proxy Execution **T1047** WMI		Accès au jump server concentrant l'accès aux BDD **T1021** Remote Service		Accès aux serveurs de BDD **T1021** Remote Service	
						Identification des BDD stockant des PII via un script **T1119** Automated Collection

incidents. Des techniques vues spécifiquement dans l'environnement protégé seront ajoutées à ATT&CK et les « analytics » développés de manière « ad-hoc » lors d'incident (idéalement supportés par des outils) seront reversés à la base CAR interne, prêts à être utilisés à la fois par le SOC/CERT/CSIRT et les « hunters ». Il est en effet important que des contrôles conçus pendant un incident passent en mode récurrent dans l'environnement pour éviter qu'une même situation se reproduise. La base interne sera mise à jour avec les logiciels rencontrés et analysés lors de l'incident (permettant d'aller plus loin qu'une capitalisation à base d'IoC).

CONCLUSION

Cet article n'a fait qu'effleurer les usages possibles du framework ATT&CK qui, par son approche systématique, apparaît comme un bon outil à utiliser dans la structuration des activités de sécurité opérationnelles qu'elles soient proactives, détectives ou réactives.

Son utilisation permettra de développer à la fois un référentiel et un langage commun, mais aussi d'alimenter une feuille de route et des indicateurs de maturité ou encore des méthodes de capitalisation.

Son déploiement effectif (et complet) nécessite néanmoins de disposer à la fois de temps et de nombreuses sources de données, parfois très proches du système d'exploitation (par exemple les appels d'API). Il conviendra d'aborder l'architecture sécurité de l'entité concernée dans son ensemble pour en tirer parti.

L'usage de l'outil (et non l'outil lui-même) doit s'inscrire dans le temps, sa valeur venant de la capitalisation opérationnelle qui en sera faite. C'est dans ce sens que le MITRE a organisé la conférence ATT&CKcon en octobre 2018 **[CON]**.

Merci à FM, PAC et SLA pour leur relecture attentive et les commentaires pertinents.

RÉFÉRENCES

[MATRIX] https://attack.mitre.org/wiki/Main_Page

[NAV] https://mitre.github.io/attack-navigator/enterprise/

[CAR] https://car.mitre.org/

[TTP] http://ryanstillions.blogspot.com/2014/04/on-ttps.html

[CARET] https://car.mitre.org/caret/#/

[PRE] https://attack.mitre.org/pre-attack/index.php/Main_Page

[MOB] https://attack.mitre.org/mobile/index.php/Main_Page

[ETSI] https://www.etsi.org/technologies-clusters/technologies/information-security-indicators

[WP] https://www.mitre.org/sites/default/files/publications/16-3713-finding-cyber-threats%20with%20att%26ck-based-analytics.pdf

[ENDGAME] https://www.endgame.com/resource/white-paper/endgame-guide-threat-hunting-practitioners-edition

[SYSMON] https://car.mitre.org/wiki/Sysmon_(3.2)

[CERTJP] https://www.jpcert.or.jp/english/pub/sr/20170612ac-ir_research_en.pdf

[OODA] https://fr.wikipedia.org/wiki/Boucle_OODA

[MTREND] https://www.fireeye.com/content/dam/collateral/en/mtrends-2018.pdf

[PAIN] http://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html

[BHE] https://www.blackhat.com/docs/eu-17/materials/eu-17-Miller-CALDERA-Automating-Adversary-Emulation.pdf

[CGH] https://github.com/mitre/caldera

[MGH] https://github.com/uber-common/metta

[CON] https://www.mitre.org/attackcon

ACTUELLEMENT DISPONIBLE

LINUX PRATIQUE HORS-SÉRIE N°44

NE LE MANQUEZ PAS

CHEZ VOTRE MARCHAND DE JOURNAUX ET SUR :

https://www.ed-diamond.com

MACHINE LEARNING POUR LES SYSTÈMES DE DÉTECTION : RECOMMANDATIONS ET SOLUTIONS AVEC SECUML

Anaël BEAUGNON – ANSSI – anael.beaugnon@ssi.gouv.fr

Le machine learning est souvent présenté comme une solution miracle pour les systèmes de détection. Dans cet article, j'identifie les pièges à éviter et je propose des solutions pratiques, disponibles dans SecuML [SECUML], pour construire des modèles de détection performants avec du machine learning.

Photo d'illustration.

Le machine learning est une technique attractive pour automatiser la création de règles de détection et renforcer les capacités de détection des méthodes traditionnelles. Dans cet article, je commence par expliquer les grandes étapes nécessaires au déploiement d'un modèle de détection. Ensuite, j'identifie trois écueils majeurs et propose des solutions pratiques pour les éviter. Ces solutions ont l'avantage d'être génériques, elles peuvent être utilisées sur tout type de données, et sont disponibles en open source dans un outil : SecuML **[SECUML]**.

> **NOTE**
> **Dans les figures 1 et 2, pages suivantes, et tout au long de cet article, je prends comme exemple les fichiers PDF, mais les techniques de machine learning sont génériques, et peuvent donc être appliquées à n'importe quel type de données.**

MACHINE LEARNING POUR LA DÉTECTION D'INTRUSION

Les algorithmes de machine learning génèrent automatiquement des règles de détection à partir d'un jeu de données annotées (deux étiquettes : malveillant ou bénin). Ils cherchent les points communs des éléments ayant la même étiquette, et les caractéristiques discriminantes entre les deux étiquettes.

Dans cette partie, je présente les grandes étapes de construction d'un modèle de détection (voir figure 1, page suivante), et comment il est ensuite utilisé pour générer des alertes (voir figure 2, page suivante).

Construction d'un modèle de détection

» Définition de la cible de détection

Avant de se lancer dans l'apprentissage d'un modèle, il faut définir la cible de détection, c'est-à-dire les phénomènes anormaux devant générer une alerte.

Le machine learning n'est pas une solution magique qui va permettre de construire un unique modèle capable de détecter tous les comportements malveillants. Il faut prendre garde à ne pas avoir une cible de détection trop large qui empêcherait le machine learning d'apprendre des règles de détection pertinentes.

Il faut créer au minimum un modèle de détection pour chaque type de données (ex. : fichiers PDF, documents Office, applications Android, données réseau comme des NetFlow). Mais parfois il est plus judicieux de construire plusieurs modèles pour un même type de données. Par exemple, dans le cas des fichiers PDF on peut apprendre un modèle pour les fichiers exploitant une vulnérabilité du format, et un autre pour ceux embarquant du JavaScript malveillant.

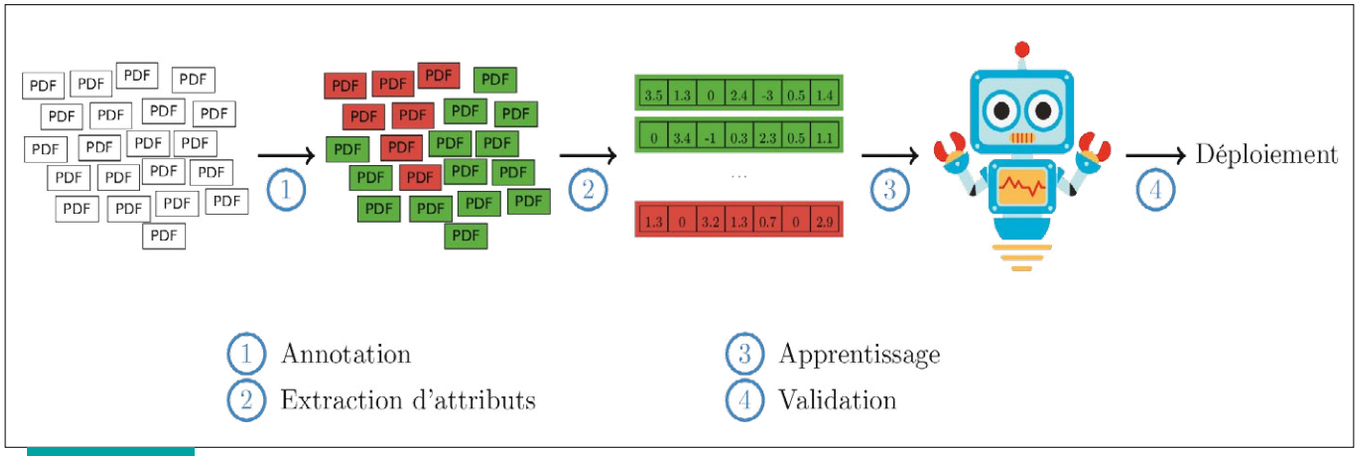


Image du robot sous licence (CC BY-SA 4.0). Auteur : Deepa Avudiappan.

FIGURE 1 *Étapes de génération d'un modèle de détection avec du machine learning.*

» Annotation et extraction des attributs

Les algorithmes de machine learning ne prennent pas en entrée des données brutes, mais des données annotées (malveillant ou bénin) en accord avec la cible de détection, et représentées sous un format standard : des vecteurs d'attributs (étapes 1 et 2 de la figure 1). Les attributs (*features* en anglais) sont des valeurs numériques utilisées par le machine learning pour construire des règles de détection. Par exemple, pour les fichiers PDF on peut considérer la présence de JavaScript ou de balises *OpenAction*. Pour des données NetFlows, on peut considérer le nombre d'octets envoyés ou reçus globalement et sur certains ports d'intérêt ou le nombre total d'adresses IP contactées. Des connaissances métier sont nécessaires pour déterminer les attributs pertinents : plus les attributs extraits sont discriminants, plus le modèle de détection sera efficace.

Les données d'apprentissage ont un rôle central en machine learning. Si elles sont biaisées, c'est-à-dire non représentatives de l'environnement de déploiement ou en désaccord avec la cible de détection, alors le modèle résultant aura une performance médiocre. Dans la partie 2, je détaille cette notion de biais d'apprentissage et j'apporte des solutions pour les éviter.

» Apprentissage du modèle

Une fois qu'on a recueilli un jeu de données d'apprentissage, on peut construire un modèle. De nombreuses bibliothèques de machine learning (ex. : scikit-learn, TensorFlow, Weka, Mahout) permettent d'apprendre des modèles sans se préoccuper des détails mathématiques, et proposent en général un nombre considérable d'algorithmes. Dans la partie 3, j'explique comment choisir un type de modèles adapté aux systèmes de détection.

» Validation du modèle avant le déploiement

Avant d'envisager le déploiement d'un modèle, il faut le valider, c'est-à-dire évaluer son adéquation avec la cible de détection visée. Dans la partie 4, j'explique comment bien valider un modèle pour éviter les mauvaises surprises lors de la mise en production.

Génération des alertes

» Extraction d'attributs

Pour savoir si une alerte doit être générée pour une donnée, il faut d'abord la représenter sous forme d'un vecteur d'attributs, avec la même méthode que celle utilisée dans la phase d'apprentissage.

Photo d'illustration.

En pratique, les systèmes de détection ont un parseur pour chaque type de données manipulé. Si on souhaite utiliser du machine learning, il faut étendre ces parseurs avec un module d'extraction d'attributs qui calcule des attributs numériques à partir de la représentation interne des données.

Application du modèle

Une fois que la donnée est sous forme d'un vecteur numérique, on peut appliquer le modèle de détection. En général, la sortie n'est pas binaire (malveillant ou bénin), mais une probabilité de malveillance. Une alerte est donc générée seulement si cette probabilité est supérieure à un seuil fixé par le responsable du système de détection en fonction du taux de détection souhaité ou du taux de fausses alarmes toléré. Cette probabilité permet de classer les alertes et d'identifier celles que les opérateurs de sécurité doivent analyser en priorité.

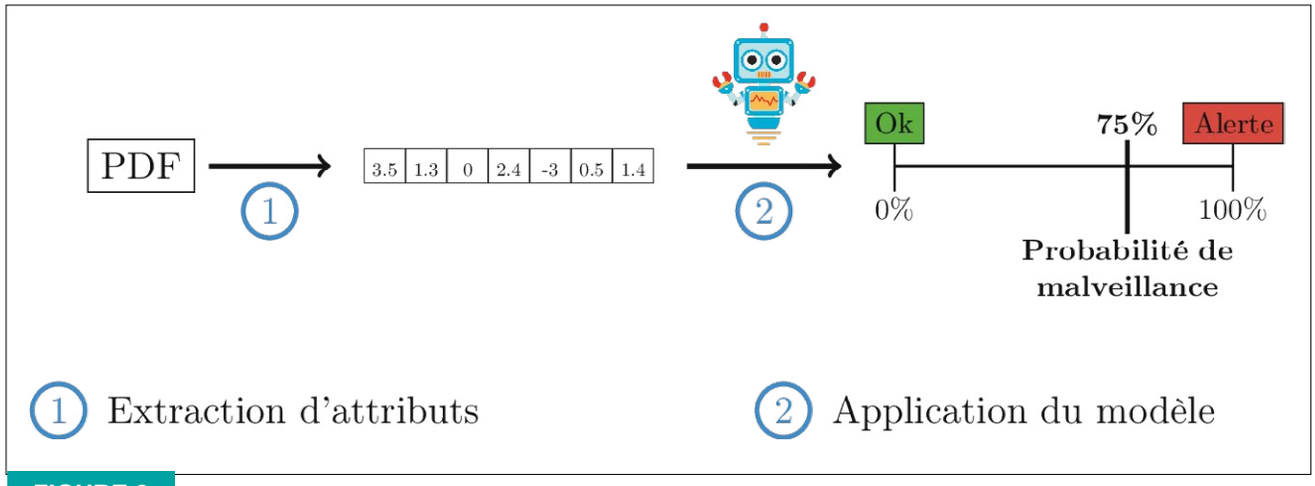


Image du robot sous licence (CC BY-SA 4.0). Auteur : Deepa Avudiappan.

FIGURE 2 *Génération des alertes avec un modèle de machine learning.*

ATTENTION AUX BIAIS D'APPRENTISSAGE !

Biais d'apprentissage

Ce problème a été médiatisé avec la dénonciation de modèles de machine learning racistes ou sexistes **[1, 2]**. Les articles de presse parlent souvent de « biais algorithmiques » alors que ce ne sont pas les algorithmes de machine learning qui sont en cause, mais les données d'apprentissage. Le modèle est le reflet direct des données d'apprentissage : si elles sont biaisées, le modèle le sera aussi.

Dans le cadre des systèmes de détection, les données d'apprentissage doivent être représentatives de l'environnement de déploiement et en accord avec la cible de détection. Dans le cas contraire, les données d'apprentissage sont biaisées et le modèle résultant sera médiocre (faible taux de détection et/ou fort taux de fausses alarmes).

Certains jeux de données sont publics et largement utilisés dans des publications académiques, mais ils sont souvent inadaptés à l'apprentissage d'un modèle destiné à être déployé dans un système de détection opérationnel. En effet, ils deviennent rapidement obsolètes et ils sont souvent non représentatifs des données observées en pratique.

Annoter un jeu de données in situ avec ILAB

Une solution pour éviter les biais d'apprentissage est d'annoter des données provenant directement de l'environnement de déploiement. Afin de réduire la charge de travail liée aux annotations manuelles, nous avons conçu et implémenté un système d'annotations nommé ILAB **[ILAB]**, disponible dans SecuML **[SECUML]**. ILAB maximise la performance de détection du modèle tout en minimisant le nombre d'annotations manuelles : une stratégie d'apprentissage actif sélectionne les données les plus intéressantes à annoter. Nos expériences utilisateur, menées avec quatre experts en sécurité, ont montré qu'ILAB est un système d'annotations adapté à leurs besoins.

En somme, les jeux de données publics constituent des données d'apprentissage bon marché, mais il faut les utiliser avec précaution. Il faut garder à l'esprit qu'ils peuvent être biaisés et donc engendrer des modèles médiocres. Une solution pour s'assurer de la qualité des données d'apprentissage est d'annoter des données issues de l'environnement de déploiement avec ILAB.

ÉVITER LES MODÈLES « BOÎTE NOIRE »

L'importance de l'interprétation

L'interprétation des modèles est primordiale dans le cadre des systèmes de détection. Le responsable du système de détection souhaite comprendre le fonctionnement des méthodes de détection déployées, et les opérateurs de sécurité ont besoin de comprendre pourquoi une alerte a été générée pour la traiter correctement. Nous verrons aussi dans la partie 4 que la transparence d'un modèle facilite grandement sa validation.

Il n'y a pas que les réseaux de neurones...

Les réseaux de neurones sont tellement populaires, notamment grâce à leurs résultats exceptionnels en reconnaissance d'images, que l'amalgame entre deep learning et machine learning est souvent fait. Cependant, ce ne sont pas les modèles les mieux adaptés aux systèmes de détection. Ce sont des boîtes noires : il est complexe de comprendre les règles de détection apprises à partir des données.

VOUS L'AVEZ RATÉ ?
VOUS AVEZ UNE DEUXIÈME CHANCE !

GNU/LINUX MAGAZINE HORS-SÉRIE n°96

À NOUVEAU DISPONIBLE

CHEZ VOTRE MARCHAND DE JOURNAUX ET SUR :

https://www.ed-diamond.com

Modèles interprétables

Les modèles linéaires (ex. : régression logistique, SVM) sont facilement interprétables : ils associent un coefficient à chacun des attributs (voir figure 3). Plus la valeur absolue du coefficient est élevée, plus l'attribut influe sur la prise de décision du modèle de détection. Les attributs associés à des coefficients négatifs (resp. positifs) caractérisent les données bénignes (resp. malveillantes).

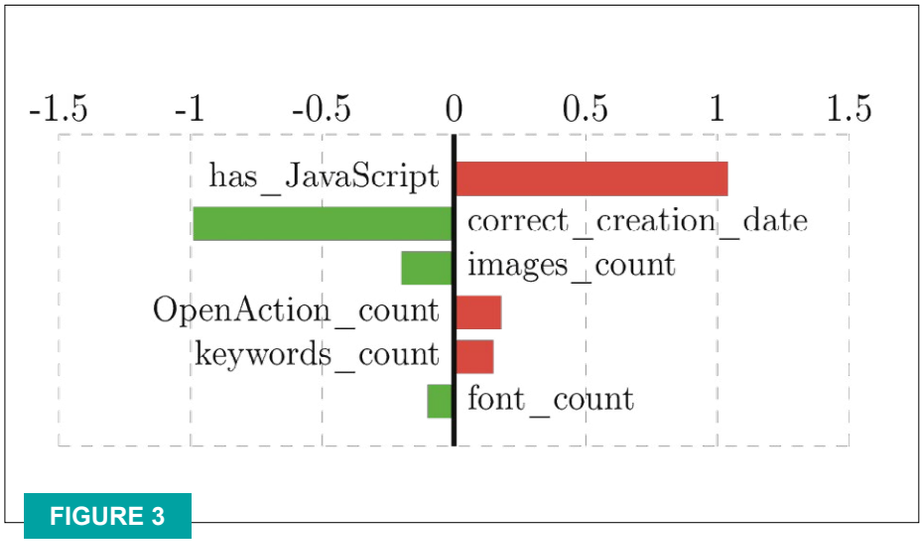


FIGURE 3

Attributs les plus influents d'un modèle de détection de fichiers PDF malveillants (régression logistique apprise sur le jeu de données Contagio **[CONTAGIO]** *avec les attributs présentés dans l'article de recherche* **[3]***).*

En général, les experts en sécurité apprécient les modèles linéaires, car ils peuvent faire une analogie avec les systèmes experts. Un modèle linéaire est un système expert où les coefficients associés à chaque attribut ont été choisis automatiquement à partir de données annotées. Ainsi, grâce au machine learning les coefficients des attributs peuvent automatiquement être mis à jour à partir des vrais et faux positifs analysés par les opérateurs de sécurité.

En pratique, je conseille de toujours commencer par apprendre un modèle linéaire qui a le grand avantage d'être facilement interprétable. Dans le cas où un modèle linéaire n'est pas suffisamment complexe (les performances du modèle sont médiocres même sur les données d'apprentissage), on peut alors se tourner vers des modèles plus complexes (ex. : *random forest*, *gradient boosting*, réseaux de neurones).

Photo d'illustration.

LA VALIDATION NE DOIT PAS SE LIMITER À DES CHIFFRES

Étudier le fonctionnement d'un modèle

La phase de validation permet de décider si un modèle de détection peut être déployé, mais elle doit aussi indiquer des pistes d'amélioration en cas de dysfonctionnement. Elle ne doit donc pas se limiter à l'étude d'indicateurs de performance tels que les taux de détection ou de fausses alarmes. Il est nécessaire d'analyser plus finement le comportement du modèle.

» Attributs les plus influents

Si le modèle choisi est interprétable, on peut analyser ses attributs les plus influents pour diagnostiquer d'éventuels biais d'apprentissage et évaluer la robustesse du modèle face à des tentatives de contournement.

Si nous reprenons l'exemple de la figure 3, deux attributs sont prédominants : `has_JavaScript` et `correct_creation_date`. L'analyse des valeurs de ces attributs sur les données d'apprentissage, le jeu de données Contagio **[CONTAGIO]**, nous montre que la très grande majorité des fichiers PDF malveillants comportent du JavaScript et leur date de création est souvent non renseignée ou invalide, contrairement aux fichiers bénins. En somme, les données d'apprentissage sont stéréotypées, et le modèle résultant est donc niais : il aura tendance à générer une alerte dès qu'un fichier PDF comporte du JavaScript ou a une date de création non renseignée ou invalide.

Il est aussi intéressant d'évaluer l'effort nécessaire à un attaquant pour modifier la valeur des attributs les plus influents, pour contourner la détection, tout en maintenant la charge malveillante. L'attribut `has_JavaScript` ne peut pas être facilement modifié si la charge malveillante est contenue dans un objet JavaScript. En revanche, l'attaquant peut facilement spécifier la date de création pour réduire la probabilité de malveillance prédite.

» Prédictions

L'analyse de certaines données, notamment celles qui sont proches de la frontière de décision (probabilité de malveillance proche de 50%) et les erreurs de prédiction, peut indiquer des pistes d'amélioration. Analyser ces données permet de détecter d'éventuelles erreurs d'annotation et aide à trouver de nouveaux attributs pertinents.

Diagnostiquer un modèle avec DIADEM

DIADEM **[DIADEM]**, disponible dans SecuML **[SECUML]**, facilite l'apprentissage et la validation de modèles de détection. DIADEM affiche évidemment les indicateurs de performance classiques, mais il permet aussi d'étudier les attributs les plus influents du modèle et d'examiner des prédictions individuellement.

CONCLUSION

Déployer des modèles de machine learning dans les systèmes de détection, en parallèle des méthodes traditionnelles, est une bonne façon de renforcer leurs capacités de détection. Afin de favoriser leur déploiement, nous mettons à disposition SecuML **[SECUML]**. Si vous souhaitez plus de détails techniques, vous pouvez vous référer à ma thèse *Expert-in-the-Loop Supervised Learning for Computer Security Detection Systems* **[THESE]**. ▌

RÉFÉRENCES

[1] « Amazon a abandonné une IA de recrutement sexiste », Génération Nouvelles Technologies, 2018

[2] « Racisme, sexisme : on entraîne trop souvent des IA avec des photos d'hommes blancs », Numerama, 2018

[3] Charles Smutz et al. « Malicious PDF detection using metadata and structural features », ACSAC 2012

[CONTAGIO] **http://contagiodump.blogspot.com/**

[DIADEM] « DIADEM : DIAgnosis of DEtection Models » : **https://anssifr.github.io/SecuML/_build/html/SecuML.DIADEM.html**

[ILAB] « ILAB : Interactive LABeling » : **https://anssi-fr.github.io/SecuML/_build/html/SecuML.ILAB.html**

[SECUML] **https://github.com/ANSSI-FR/SecuML**

[THESE] A. Beaugnon, « Expert-in-the-loop Supervised Learning for Computer Security Detection Systems », thèse de doctorat, 2018

digital.security

COnnected DEvices EXploitation

Formations de hacking hardware et software des objets connectés

CODEX01

4 JOURS
FORMATION EN ANGLAIS

- S'interfacer et communiquer avec un objet connecté
- Déterminer la surface d'attaque d'un objet connecté
- Extraire les micro-logiciels de différentes façons
- Analyser les micro-logiciels
- Identifier des vulnérabilités dans un micro-logiciel et les exploiter
- Conserver un accès résistant aux mises à jour de l'objet compromis

CODEX02 AVANCÉ

5 JOURS
FORMATION EN ANGLAIS

- Contourner les protections contre l'extraction de micro-logiciel
- Obtenir un accès privilégié au système d'un objet connecté
- Analyser un micro-logiciel d'un système ne reposant pas sur un système d'exploitation
- Identifier et exploiter des vulnérabilités applicatives sur architecture ARM
- Réaliser des attaques par canaux auxiliaires afin de contourner des restrictions
- Identifier, analyser et exploiter de multiples protocoles de communications

Les formations seront dispensées par **Damien Cauquil** (**@virtualabs**), expert sécurité spécialisé dans la sécurité des objets connectés depuis plus de 5 ans. Il est par ailleurs l'auteur de plusieurs outils de référence publiés lors de conférences comme DEF CON ou le Chaos Communicaton Congress et a trouvé et communiqué diverses vulnérabilités dans des objets connectés ainsi que dans plusieurs protocoles de communication.

- 10 % pour les 10 premiers inscrits avec le code 7mq9s3

Inscriptions par mail à formations@digital.security